**Q. *What are the types of Exceptions? Explain the hierarchy of Java Exception classes?***

Exception is an error event that can happen during the execution of a program and disrupts its normal flow.

**Types of Java Exceptions**

**1. Checked Exception**: The classes which directly inherit Throwable class except RuntimeException and Error are known as checked exceptions e.g. IOException, SQLException etc. Checked exceptions are checked at compile-time.  
**2. Unchecked Exception**: The classes which inherit RuntimeException are known as unchecked exceptions e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.  
**3. Error**: Error is irrecoverable e.g. OutOfMemoryError, VirtualMachineError, AssertionError etc.

**Hierarchy of Java Exception classes**  
The java.lang.Throwable class is the root class of Java Exception hierarchy which is inherited by two subclasses: Exception and Error.

[](https://github.com/learning-zone/java-interview-questions/blob/master/assets/exception.png)

Example:

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.InputStream;

public class CustomExceptionExample {

public static void main(String[] args) throws MyException {

try {

processFile("file.txt");

} catch (MyException e) {

processErrorCodes(e);

}

}

private static void processErrorCodes(MyException e) throws MyException {

switch(e.getErrorCode()){

case "BAD\_FILE\_TYPE":

System.out.println("Bad File Type, notify user");

throw e;

case "FILE\_NOT\_FOUND\_EXCEPTION":

System.out.println("File Not Found, notify user");

throw e;

case "FILE\_CLOSE\_EXCEPTION":

System.out.println("File Close failed, just log it.");

break;

default:

System.out.println("Unknown exception occured," +e.getMessage());

e.printStackTrace();

}

}

private static void processFile(String file) throws MyException {

InputStream fis = null;

try {

fis = new FileInputStream(file);

} catch (FileNotFoundException e) {

throw new MyException(e.getMessage(),"FILE\_NOT\_FOUND\_EXCEPTION");

} finally {

try {

if(fis !=null) fis.close();

} catch (IOException e) {

throw new MyException(e.getMessage(),"FILE\_CLOSE\_EXCEPTION");

}

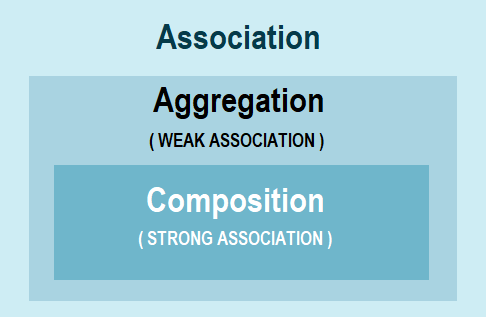
}

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is the difference between aggregation and composition?***

[](https://github.com/learning-zone/java-interview-questions/blob/master/assets/aggregation.png)

**Aggregation**: We call aggregation those relationships whose **objects have an independent lifecycle, but there is ownership**, and child objects cannot belong to another parent object.

Example: Since Organization has Person as employees, the relationship between them is Aggregation. Here is how they look like in terms of Java classes

public class Organization {

private List employees;

}

public class Person {

private String name;

}

**Composition**: We use the term composition to refer to relationships whose objects **don’t have an independent lifecycle**, and if the parent object is deleted, all child objects will also be deleted.

Example: Since Engine is-part-of Car, the relationship between them is Composition. Here is how they are implemented between Java classes.

public class Car {

//final will make sure engine is initialized

private final Engine engine;

public Car(){

engine = new Engine();

}

}

class Engine {

private String type;

}

|  |  |
| --- | --- |
| **Aggregation** | **Composition** |
| Aggregation is a weak Association. | Composition is a strong Association. |
| Class can exist independently without owner. | Class can not meaningfully exist without owner. |
| Have their own Life Time. | Life Time depends on the Owner. |
| A uses B. | A owns B. |
| Child is not owned by 1 owner. | Child can have only 1 owner. |
| Has-A relationship. A has B. | Part-Of relationship. B is part of A. |
| Denoted by a empty diamond in UML. | Denoted by a filled diamond in UML. |
| We do not use "final" keyword for Aggregation. | "final" keyword is used to represent Composition. |
| Examples: - Car has a Driver. - A Human uses Clothes. - A Company is an aggregation of People. - A Text Editor uses a File. - Mobile has a SIM Card. | Examples: - Engine is a part of Car. - A Human owns the Heart. - A Company is a composition of Accounts. - A Text Editor owns a Buffer. - IMEI Number is a part of a Mobile. |

*Note: "final" keyword is used in Composition to make sure child variable is initialized.*

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is difference between Heap and Stack Memory in java?***

**Java Heap Space**

Java Heap space is used by java runtime to allocate memory to Objects and JRE classes. Whenever we create any object, it’s always created in the Heap space.

Garbage Collection runs on the heap memory to free the memory used by objects that doesn’t have any reference. Any object created in the heap space has global access and can be referenced from anywhere of the application.

**Java Stack Memory**

Stack in java is a section of memory which contains methods, local variables and reference variables. Local variables are created in the stack.

Stack memory is always referenced in LIFO (Last-In-First-Out) order. Whenever a method is invoked, a new block is created in the stack memory for the method to hold local primitive values and reference to other objects in the method.

As soon as method ends, the block becomes unused and become available for next method. Stack memory size is very less compared to Heap memory.

**Difference**

| **Parameter** | **Stack Memory** | **Heap Space** |
| --- | --- | --- |
| Application | Stack is used in parts, one at a time during execution of a thread | The entire application uses Heap space during runtime |
| Size | Stack has size limits depending upon OS and is usually smaller then Heap | There is no size limit on Heap |
| Storage | Stores only primitive variables and references to objects that are created in Heap Space | All the newly created objects are stored here |
| Order | It is accessed using Last-in First-out (LIFO) memory allocation system | This memory is accessed via complex memory management techniques that include Young Generation, Old or Tenured Generation, and Permanent Generation. |
| Life | Stack memory only exists as long as the current method is running | Heap space exists as long as the application runs |
| Efficiency | Comparatively much faster to allocate when compared to heap | Slower to allocate when compared to stack |
| Allocation/Deallocation | This Memory is automatically allocated and deallocated when a method is called and returned respectively | Heap space is allocated when new objects are created and deallocated by Gargabe Collector when they are no longer referenced |

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is JVM and is it platform independent?***

Java Virtual Machine (JVM) is a specification that provides runtime environment in which java bytecode(.class files) can be executed. The JVM is the platform. The JVM acts as a "virtual" machine or processor. Java's platform independence consists mostly of its Java Virtual Machine (JVM). JVM makes this possible because it is aware of the specific instruction lengths and other particularities of the platform (Operating System).

The JVM is not platform independent. Java Virtual Machine (JVM) provides the environment to execute the java file(. Class file). So at the end it's depends on kernel and kernel is differ from OS (Operating System) to OS. The JVM is used to both translate the bytecode into the machine language for a particular computer and actually execute the corresponding machine-language instructions as well.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is JIT compiler in Java?***

The Just-In-Time (JIT) compiler is a component of the runtime environment that improves the performance of Java applications by compiling bytecodes to native machine code at run time.

Java programs consists of classes, which contain platform-neutral bytecodes that can be interpreted by a JVM on many different computer architectures. At run time, the JVM loads the class files, determines the semantics of each individual bytecode, and performs the appropriate computation. The additional processor and memory usage during interpretation means that a Java application performs more slowly than a native application. The JIT compiler helps improve the performance of Java programs by compiling bytecodes into native machine code at run time. The JIT compiler is enabled by default. When a method has been compiled, the JVM calls the compiled code of that method directly instead of interpreting it.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is Classloader in Java? What are different types of classloaders?***

The **Java ClassLoader** is a part of the Java Runtime Environment that dynamically loads Java classes into the Java Virtual Machine. Java code is compiled into class file by javac compiler and JVM executes Java program, by executing byte codes written in class file. ClassLoader is responsible for loading class files from file system, network or any other source.

**Types of ClassLoader**

**a) Bootstrap Class Loader**: It loads standard JDK class files from rt.jar and other core classes. It loads class files from jre/lib/rt.jar. For example, java.lang package class.

**b) Extensions Class Loader**: It loads classes from the JDK extensions directly usually JAVA\_HOME/lib/ext directory or any other directory as java.ext.dirs.

**c) System Class Loader**: It loads application specific classes from the CLASSPATH environment variable. It can be set while invoking program using -cp or classpath command line options.

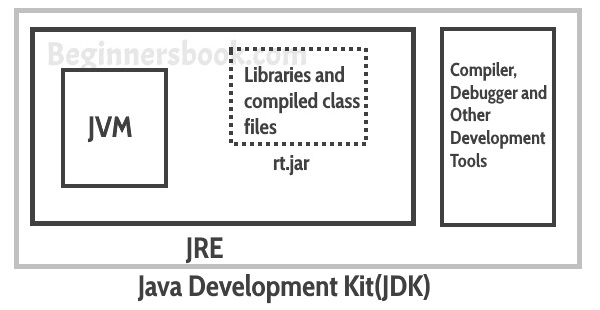
[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *Java Compiler is stored in JDK, JRE or JVM?***

**JDK**: Java Development Kit is the core component of Java Environment and provides all the tools, executables and binaries required to compile, debug and execute a Java Program.

**JVM**: JVM is responsible for converting Byte code to the machine specific code. JVM is also platform dependent and provides core java functions like memory management, garbage collection, security etc. JVM is customizable and we can use java options to customize it, for example allocating minimum and maximum memory to JVM. JVM is called virtual because it provides an interface that does not depend on the underlying operating system and machine hardware.

**JRE**: Java Runtime Environment provides a platform to execute java programs. JRE consists of JVM and java binaries and other classes to execute any program successfully.

[](https://github.com/learning-zone/java-interview-questions/blob/master/assets/jdk.jpg)

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is the difference between factory and abstract factory pattern?***

The Factory Method is usually categorised by a switch statement where each case returns a different class, using the same root interface so that the calling code never needs to make decisions about the implementation.

For example credit card validator factory which returns a different validator for each card type.

public ICardValidator GetCardValidator (string cardType)

{

switch (cardType.ToLower())

{

case "visa":

return new VisaCardValidator();

case "mastercard":

case "ecmc":

return new MastercardValidator();

default:

throw new CreditCardTypeException("Do not recognise this type");

}

}

Abstract Factory patterns work around a super-factory which creates other factories. This factory is also called as factory of factories. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

In Abstract Factory pattern an interface is responsible for creating a factory of related objects without explicitly specifying their classes. Each generated factory can give the objects as per the Factory pattern.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What are the methods used to implement for key Object in HashMap?***

**1. equals()** and **2. hashcode()** Class inherits methods from the following classes in terms of HashMap

* java.util.AbstractMap
* java.util.Object
* java.util.Map

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is difference between the Inner Class and Sub Class?***

Nested Inner class can access any private instance variable of outer class. Like any other instance variable, we can have access modifier private, protected, public and default modifier.

class Outer {

class Inner {

public void show() {

System.out.println("In a nested class method");

}

}

}

class Main {

public static void main(String[] args) {

Outer.Inner in = new Outer().new Inner();

in.show();

}

}

A subclass is class which inherits a method or methods from a superclass.

class Car {

//...

}

class HybridCar extends Car {

//...

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *Can we import same package/class two times? Will the JVM load the package twice at runtime?***

We can import the same package or same class multiple times. The JVM will internally load the class only once no matter how many times import the same class.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *Distinguish between static loading and dynamic class loading?***

**Static Class Loading**: Creating objects and instance using new keyword is known as static class loading. The retrieval of class definition and instantiation of the object is done at compile time.

class TestClass {

public static void main(String args[]) {

TestClass tc = new TestClass();

}

}

**Dynamic Class Loading**: Loading classes use Class.forName() method. Dynamic class loading is done when the name of the class is not known at compile time.

Class.forName (String className);

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is the difference between transient and volatile variable in Java?***

**Transient**: The transient modifier tells the Java object serialization subsystem to exclude the field when serializing an instance of the class. When the object is then deserialized, the field will be initialized to the default value; i.e. null for a reference type, and zero or false for a primitive type.

public transient int limit = 55; // will not persist

public int b; // will persist

**Volatile**: The volatile modifier tells the JVM that writes to the field should always be synchronously flushed to memory, and that reads of the field should always read from memory. This means that fields marked as volatile can be safely accessed and updated in a multi-thread application without using native or standard library-based synchronization.

public class MyRunnable implements Runnable {

private volatile boolean active;

public void run() {

active = true;

while (active) {

}

}

public void stop() {

active = false;

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *How many types of memory areas are allocated by JVM?***

JVM is a program which takes Java bytecode and converts the byte code (line by line) into machine understandable code. JVM perform some particular types of operations:

* Loading of code
* Verification of code
* Executing the code
* It provide run-time environment to the users

**Types of Memory areas allocated by the JVM:**

**1. Classloader**: Classloader is a subsystem of JVM that is used to load class files.  
**2. Class(Method) Area**: Class(Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.  
**3. Heap**: It is the runtime data area in which objects are allocated.  
**4. Stack**: Java Stack stores frames.It holds local variables and partial results, and plays a part in method invocation and return. Each thread has a private JVM stack, created at the same time as thread.  
**5. Program Counter Register**: PC (program counter) register. It contains the address of the Java virtual machine instruction currently being executed.  
**6. Native Method Stack**: It contains all the native methods used in the application.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What will be the initial value of an object reference which is defined as an instance variable?***

The object references are all initialized to null in Java. However in order to do anything useful with these references, It must set to a valid object, else you will get NullPointerExceptions everywhere you try to use such default initialized references.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *How can constructor chaining be done using this keyword?***

Java constructor chaining is a method of calling one constructor with the help of another while considering the present object. It can be done in 2 ways –

* **Within same class**: It can be done using this() keyword for constructors in the same class.
* **From base class**: By using super() keyword to call a constructor from the base class.

// Java program to illustrate Constructor Chaining

// within same class Using this() keyword

class Temp

{

// default constructor 1

// default constructor will call another constructor

// using this keyword from same class

Temp() {

// calls constructor 2

this(5);

System.out.println("The Default constructor");

}

// parameterized constructor 2

Temp(int x) {

// calls constructor 3

this(10, 20);

System.out.println(x);

}

// parameterized constructor 3

Temp(int x, int y) {

System.out.println(10 + 20);

}

public static void main(String args[]) {

// invokes default constructor first

new Temp();

}

}

Ouput:

30

10

The Default constructor

// Java program to illustrate Constructor Chaining to

// other class using super() keyword

class Base

{

String name;

// constructor 1

Base() {

this("");

System.out.println("No-argument constructor of base class");

}

// constructor 2

Base(String name) {

this.name = name;

System.out.println("Calling parameterized constructor of base");

}

}

class Derived extends Base

{

// constructor 3

Derived() {

System.out.println("No-argument constructor of derived");

}

// parameterized constructor 4

Derived(String name) {

// invokes base class constructor 2

super(name);

System.out.println("Calling parameterized constructor of derived");

}

public static void main(String args[]) {

// calls parameterized constructor 4

Derived obj = new Derived("test");

// Calls No-argument constructor

// Derived obj = new Derived();

}

}

Output:

Calling parameterized constructor of base

Calling parameterized constructor of derived

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *Can you declare the main method as final?***

Yes. We can declare main method as final. But, In inheritance concept we cannot declare main method as final in parent class. It give compile time error. The main method has to be public because it has to be called by JVM which is outside the scope of the package and hence would need the access specifier-public.

public class Test {

public final static void main(String[] args) throws Exception {

System.out.println("This is Test Class");

}

}

class Child extends Test {

public static void main(String[] args) throws Exception {

System.out.println("This is Child Class");

}

}

Output

Cannot override the final method from Test.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is the difference between the final method and abstract method?***

Final method is a method that is marked as final, i.e. it cannot be overridden anymore. Just like final class cannot be inherited anymore.

Abstract method, on the other hand, is an empty method that is ought to be overridden by the inherited class. Without overriding, you will quickly get compilation error.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is the difference between compile-time polymorphism and runtime polymorphism?***

There are two types of polymorphism in java:

1. Static Polymorphism also known as compile time polymorphism
2. Dynamic Polymorphism also known as runtime polymorphism

**Example of static Polymorphism**

Method overloading is one of the way java supports static polymorphism. Here we have two definitions of the same method add() which add method would be called is determined by the parameter list at the compile time. That is the reason this is also known as compile time polymorphism.

class SimpleCalculator

{

int add(int a, int b) {

return a + b;

}

int add(int a, int b, int c) {

return a + b + c;

}

}

public class Demo

{

public static void main(String args[]) {

SimpleCalculator obj = new SimpleCalculator();

System.out.println(obj.add(10, 20));

System.out.println(obj.add(10, 20, 30));

}

}

Output:

30

60

**Runtime Polymorphism (or Dynamic polymorphism)**

It is also known as Dynamic Method Dispatch. Dynamic polymorphism is a process in which a call to an overridden method is resolved at runtime, thats why it is called runtime polymorphism.

class ABC {

public void myMethod() {

System.out.println("Overridden Method");

}

}

public class XYZ extends ABC {

public void myMethod() {

System.out.println("Overriding Method");

}

public static void main(String args[]) {

ABC obj = new XYZ();

obj.myMethod();

}

}

Output:

Overriding Method

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *Can you achieve Runtime Polymorphism by data members?***

No, we cannot achieve runtime polymorphism by data members. Method is overridden not the data members, so runtime polymorphism can not be achieved by data members.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *Can you have virtual functions in Java?***

In Java, all non-static methods are by default **virtual functions**. Only methods marked with the keyword final, which cannot be overridden, along with private methods, which are not inherited, are non-virtual.

**Virtual function with Interface**

/\*\*

\* The function applyBrakes() is virtual because

\* functions in interfaces are designed to be overridden.

\*\*/

interface Bicycle {

void applyBrakes();

}

class ACMEBicycle implements Bicycle {

public void applyBrakes(){ //Here we implement applyBrakes()

System.out.println("Brakes applied"); //function

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is covariant return type?***

It is possible to have different return type for a overriding method in child class, but child’s return type should be sub-type of parent’s return type. Overriding method becomes variant with respect to return type. The covariant return type specifies that the return type may vary in the same direction as the subclass.

class SuperClass {

SuperClass get() {

System.out.println("SuperClass");

return this;

}

}

public class Tester extends SuperClass {

Tester get() {

System.out.println("SubClass");

return this;

}

public static void main(String[] args) {

SuperClass tester = new Tester();

tester.get();

}

}

Output:

Subclass

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is the difference between abstraction and encapsulation?***

* Abstraction solves the problem at design level while Encapsulation solves it implementation level.
* In Java, Abstraction is supported using interface and abstract class while Encapsulation is supported using access modifiers e.g. public, private and protected.
* Abstraction is about hiding unwanted details while giving out most essential details, while Encapsulation means hiding the code and data into a single unit e.g. class or method to protect inner working of an object from outside world.

|  |  |
| --- | --- |
| **Abstraction** | **Encapsulation** |
| Abstraction is a process of hiding the implementation details and showing only functionality to the user. | Encapsulation is a process of wrapping code and data together into a single unit |
| Abstraction lets you focus on what the object does instead of how it does it. | Encapsulation provides you the control over the data and keeping it safe from outside misuse. |
| Abstraction solves the problem in the Design Level. | Encapsulation solves the problem in the Implementation Level. |
| Abstraction is implemented by using Interfaces and Abstract Classes. | Encapsulation is implemented by using Access Modifiers (private, default, protected, public) |
| Abstraction means hiding implementation complexities by using interfaces and abstract class. | Encapsulation means hiding data by using setters and getters. |

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *Can there be an abstract method without an abstract class?***

Yes. because methods in an interface are also abstract. so the interface can be use to declare abstract method.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *Can we use private or protected member variables in an interface?***

The java compiler adds public and abstract keywords before the interface method and **public, static and final keyword** before data members automatically

public interface Test {

public string name1;

private String email;

protected pass;

}

as you have declare variable in test interface with private and protected it will give error. if you do not specify the modifier the compiler will add public static final automatically.

public interface Test {

public static final string name1;

public static final String email;

public static final pass;

}

* interfaces cannot be instantiated that is why the variable are **static**
* interface are used to achieve the 100% abstraction there for the variable are **final**
* An interface provide a way for the client to interact with the object. If variables were not public, the clients would not have access to them. that is why variable are **public**

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *When can an object reference be cast to a Java interface reference?***

An interface reference can point to any object of a class that implements this interface

interface Foo {

void display();

}

public class TestFoo implements Foo {

void display() {

System.out.println("Hello World");

}

public static void main(String[] args) {

Foo foo = new TestFoo();

foo.display();

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *Give the hierarchy of InputStream and OutputStream classes?***

A stream can be defined as a sequence of data. There are two kinds of Streams −

* **InPutStream** − The InputStream is used to read data from a source.
* **OutPutStream** − The OutputStream is used for writing data to a destination.

**Byte Streams**  
Java byte streams are used to perform input and output of 8-bit bytes. Though there are many classes related to byte streams but the most frequently used classes are, FileInputStream and FileOutputStream.

import java.io.\*;

public class CopyFile {

public static void main(String args[]) throws IOException {

FileInputStream in = null;

FileOutputStream out = null;

try {

in = new FileInputStream("input.txt");

out = new FileOutputStream("output.txt");

int c;

while ((c = in.read()) != -1) {

out.write(c);

}

} finally {

if (in != null) {

in.close();

}

if (out != null) {

out.close();

}

}

}

}

**Character Streams**  
Java Byte streams are used to perform input and output of 8-bit bytes, whereas Java Character streams are used to perform input and output for 16-bit unicode. Though there are many classes related to character streams but the most frequently used classes are, FileReader and FileWriter.

import java.io.\*;

public class CopyFile {

public static void main(String args[]) throws IOException {

FileReader in = null;

FileWriter out = null;

try {

in = new FileReader("input.txt");

out = new FileWriter("output.txt");

int c;

while ((c = in.read()) != -1) {

out.write(c);

}

} finally {

if (in != null) {

in.close();

}

if (out != null) {

out.close();

}

}

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *Can you access non static variable in static context?***

No, non-static variable cannot be referenced in a static context directly one needs to use object.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is the purpose of the Runtime class and System class?***

**Runtime Class**: The purpose of the Runtime class is to provide access to the Java runtime system. The runtime information like memory availability, invoking the garbage collector, etc.

**System Class**: The purpose of the System class is to provide access to system resources. It contains accessibility to standard input, standart output, error output streams, current time in millis, terminating the application, etc.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What are assertions in Java?***

An assertion allows testing the correctness of any assumptions that have been made in the program. Assertion is achieved using the assert statement in Java. While executing assertion, it is believed to be true. If it fails, JVM throws an error named AssertionError. It is mainly used for testing purposes during development.

The assert statement is used with a Boolean expression and can be written in two different ways.

// First way

assert expression;

// Second way

assert expression1 : expression2;

Example:

public class Example {

public static void main(String[] args) {

int age = 14;

assert age <= 18 : "Cannot Vote";

System.out.println("The voter's age is " + age);

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *Can we have multiple public classes in a java source file?***

A Java source file can have only one class declared as **public**, we cannot put two or more public classes together in a **.java** file. This is because of the restriction that the file name should be same as the name of the public class with **.java** extension. If we want to multiple classes under consideration are to be declared as public, we have to store them in separate source files and attach the package statement as the first statement in those source files.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is the difference between abstract class and interface?***

Abstract class and interface both are used to achieve abstraction where we can declare the abstract methods. Abstract class and interface both can't be instantiated.

| **Sl.No** | **Abstract Class** | **Interface** |
| --- | --- | --- |
| 01. | Abstract class can have abstract and non-abstract methods. | Interface can have only abstract methods. Since Java 8, it can have default and static methods also. |
| 02. | Abstract class doesn't support multiple inheritance. | Interface supports multiple inheritance. |
| 03. | Abstract class can have final, non-final, static and non-static variables. | Interface has only static and final variables. |
| 04. | Abstract class can provide the implementation of interface. | Interface can't provide the implementation of abstract class. |
| 05. | The abstract keyword is used to declare abstract class. | The interface keyword is used to declare interface. |
| 06. | An abstract class can extend another Java class and implement multiple Java interfaces. | An interface can extend another Java interface only. |
| 07. | An abstract class can be extended using keyword "extends". | An interface can be implemented using keyword "implements". |
| 08. | A Java abstract class can have class members like private, protected, etc. | Members of a Java interface are public by default. |

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What are Wrapper classes?***

The wrapper class in Java provides the mechanism to convert primitive into object and object into primitive.

**Use of Wrapper classes in Java**

* **Change the value in Method**: Java supports only call by value. So, if we pass a primitive value, it will not change the original value. But, if we convert the primitive value in an object, it will change the original value.
* **Serialization**: We need to convert the objects into streams to perform the serialization. If we have a primitive value, we can convert it in objects through the wrapper classes.
* **Synchronization**: Java synchronization works with objects in Multithreading.
* **java.util package**: The java.util package provides the utility classes to deal with objects.
* **Collection Framework**: Java collection framework works with objects only. All classes of the collection framework (ArrayList, LinkedList, Vector, HashSet, LinkedHashSet, TreeSet, PriorityQueue, ArrayDeque, etc.) deal with objects only.

| **Sl.No** | **Primitive Type** | **Wrapper class** |
| --- | --- | --- |
| 01. | boolean | Boolean |
| 02. | char | Character |
| 03. | byte | Byte |
| 04. | short | Short |
| 05. | int | Integer |
| 06. | long | Long |
| 07. | float | Float |
| 08. | double | Double |

Example: Primitive to Wrapper

//Java program to convert primitive into objects

//Autoboxing example of int to Integer

class WrapperExample {

public static void main(String args[]){

//Converting int into Integer

int a=20;

Integer i = Integer.valueOf(a);//converting int into Integer explicitly

Integer j = a; //autoboxing, now compiler will write Integer.valueOf(a) internally

System.out.println(a+" "+i+" "+j);

}

}

Output

20 20 20

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is Java Reflection API?***

Java Reflection is the process of analyzing and modifying all the capabilities of a class at runtime. Reflection API in Java is used to manipulate class and its members which include fields, methods, constructor, etc. at runtime. The **java.lang.Class** class provides many methods that can be used to get metadata, examine and change the run time behavior of a class.

There are 3 ways to get the instance of Class class. They are as follows:

* forName() method of Class class
* getClass() method of Object class
* the .class syntax

**1. forName() method of Class class**

* is used to load the class dynamically.
* returns the instance of Class class.
* It should be used if you know the fully qualified name of class.This cannot be used for primitive types.

class Simple{}

class Test {

public static void main(String args[]) {

Class c = Class.forName("Simple");

System.out.println(c.getName());

}

}

Output

Simple

**2. getClass() method of Object class**

It returns the instance of Class class. It should be used if you know the type. Moreover, it can be used with primitives.

class Simple{}

class Test {

void printName(Object obj) {

Class c=obj.getClass();

System.out.println(c.getName());

}

public static void main(String args[]) {

Simple s=new Simple();

Test t=new Test();

t.printName(s);

}

}

Output

Simple

**3. The .class syntax**

If a type is available but there is no instance then it is possible to obtain a Class by appending ".class" to the name of the type.It can be used for primitive data type also.

class Test {

public static void main(String args[]) {

Class c = boolean.class;

System.out.println(c.getName());

Class c2 = Test.class;

System.out.println(c2.getName());

}

}

Output

boolean

Test

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is the default value of the local variables?***

There is no default value for local variables, so local variables should be declared and an initial value should be assigned before the first use.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *How many types of constructors are used in Java?***

In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory.

**Types of Java Constructors**

* Default Constructor (or) no-arg Constructor
* Parameterized Constructor

Example: Default Constructor (or) no-arg constructor

public class Car

{

Car() {

System.out.println("Default Constructor of Car class called");

}

public static void main(String args[]) {

//Calling the default constructor

Car c = new Car();

}

}

Output

Default Constructor of Car class called

Example: Parameterized Constructor

public class Car

{

String carColor;

Car(String carColor) {

this.carColor = carColor;

}

public void disp() {

System.out.println("Color of the Car is : "+carColor);

}

public static void main(String args[]) {

//Calling the parameterized constructor

Car c = new Car("Blue");

c.disp();

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What are the restrictions that are applied to the Java static methods?***

If a method is declared as static, it is a member of a class rather than belonging to the object of the class. It can be called without creating an object of the class. A static method also has the power to access static data members of the class.

* There are a few restrictions imposed on a static method
* The static method cannot use non-static data member or invoke non-static method directly.
* The this and super cannot be used in static context.
* The static method can access only static type data (static type instance variable).
* There is no need to create an object of the class to invoke the static method.
* A static method cannot be overridden in a subclass

class Parent {

static void display() {

System.out.println("Super class");

}

}

public class Example extends Parent {

void display() // trying to override display() {

System.out.println("Sub class");

}

public static void main(String[] args) {

Parent obj = new Example();

obj.display();

}

}

This generates a compile time error. The output is as follows −

Example.java:10: error: display() in Example cannot override display() in Parent

void display() // trying to override display()

^

overridden method is static

1 error

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is the final variable, final class, and final blank variable?***

**Final Variable**: final variables are nothing but constants. We cannot change the value of a final variable once it is initialized.

class Demo {

final int MAX\_VALUE = 99;

void myMethod() {

MAX\_VALUE = 101;

}

public static void main(String args[]) {

Demo obj = new Demo();

obj.myMethod();

}

}

Output

Exception in thread "main" java.lang.Error: Unresolved compilation problem:

The final field Demo.MAX\_VALUE cannot be assigned

at beginnersbook.com.Demo.myMethod(Details.java:6)

at beginnersbook.com.Demo.main(Details.java:10)

**Blank final variable**: A final variable that is not initialized at the time of declaration is known as blank final variable. We must initialize the blank final variable in constructor of the class otherwise it will throw a compilation error (Error: variable MAX\_VALUE might not have been initialized).

class Demo {

//Blank final variable

final int MAX\_VALUE;

Demo() {

//It must be initialized in constructor

MAX\_VALUE = 100;

}

void myMethod() {

System.out.println(MAX\_VALUE);

}

public static void main(String args[]) {

Demo obj = new Demo();

obj.myMethod();

}

}

Output

100

**Final Method**: A final method cannot be overridden. Which means even though a sub class can call the final method of parent class without any issues but it cannot override it.

class XYZ {

final void demo() {

System.out.println("XYZ Class Method");

}

}

class ABC extends XYZ {

void demo() {

System.out.println("ABC Class Method");

}

public static void main(String args[]) {

ABC obj= new ABC();

obj.demo();

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is the static import?***

The static import feature of Java 5 facilitate the java programmer to access any static member of a class directly. There is no need to qualify it by the class name.

import static java.lang.System.\*;

class StaticImportExample {

public static void main(String args[]) {

out.println("Hello");//Now no need of System.out

out.println("Java");

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *Name some classes present in java.util.regex package?***

**Java Regex**: The Java Regex or Regular Expression is an API to define a pattern for searching or manipulating strings.

**java.util.regex package**

* MatchResult interface
* Matcher class
* Pattern class
* PatternSyntaxException class

import java.util.regex.\*;

public class RegexExample {

public static void main(String args[]) {

//1st way

Pattern p = Pattern.compile(".s");//. represents single character

Matcher m = p.matcher("as");

boolean b = m.matches();

//2nd way

boolean b2 = Pattern.compile(".s").matcher("as").matches();

//3rd way

boolean b3 = Pattern.matches(".s", "as");

System.out.println(b + " " + b2 + " " + b3);

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *How will you invoke any external process in Java?***

We can invoke the external process in Java using **exec()** method of **Runtime Class**.

class ExternalProcessExample

{

public static void main(String[] args) {

try {

// Command to create an external process

String command = "C:\Program Files (x86)"+

"\Google\Chrome\Application\chrome.exe";

// Running the above command

Runtime run = Runtime.getRuntime();

Process proc = run.exec(command);

} catch (IOException e) {

e.printStackTrace();

}

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is the purpose of using BufferedInputStream and BufferedOutputStream classes?***

BufferedInputStream and BufferedOutputStream class is used for buffering an input and output stream while reading and writing, respectively. It internally uses buffer to store data. It adds more efficiency than to write data directly into a stream. So, it makes the performance fast.

**BufferedInputStreamExample.java**

import java.io.BufferedInputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

public class BufferedInputStreamExample {

public static void main(String[] args) {

File file = new File("file.txt");

FileInputStream fileInputStream = null;

BufferedInputStream bufferedInputStream = null;

try {

fileInputStream = new FileInputStream(file);

bufferedInputStream = new BufferedInputStream(fileInputStream);

// Create buffer

byte[] buffer = new byte[1024];

int bytesRead = 0;

while ((bytesRead = bufferedInputStream.read(buffer)) != -1) {

System.out.println(new String(buffer, 0, bytesRead));

}

} catch (IOException e) {

e.printStackTrace();

} finally {

try {

if (fileInputStream != null) {

fileInputStream.close();

}

if (bufferedInputStream != null) {

bufferedInputStream.close();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

Output

This is an example of reading data from file

**BufferedOutputStreamExample.java**

import java.io.BufferedOutputStream;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

public class BufferedOutputStreamExample {

public static void main(String[] args) {

File file = new File("outfile.txt");

FileOutputStream fileOutputStream=null;

BufferedOutputStream bufferedOutputStream=null;

try {

fileOutputStream = new FileOutputStream(file);

bufferedOutputStream = new BufferedOutputStream(fileOutputStream);

bufferedOutputStream.write("This is an example of writing data to a file".getBytes());

bufferedOutputStream.flush();

} catch (IOException e) {

e.printStackTrace();

} finally {

try {

if(fileOutputStream != null) {

fileOutputStream.close();

}

if(bufferedOutputStream != null) {

bufferedOutputStream.close();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

Output

This is an example of writing data to a file

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *How to set the Permissions to a file in Java?***

Java 7 has introduced PosixFilePermission Enum and **java.nio.file.Files** includes a method setPosixFilePermissions(Path path, Set<PosixFilePermission> perms) that can be used to set file permissions easily.

import java.io.File;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.nio.file.attribute.PosixFilePermission;

import java.util.HashSet;

import java.util.Set;

public class FilePermissions {

public static void main(String[] args) throws IOException {

File file = new File("/Users/file.txt");

//change permission to 777 for all the users

//no option for group and others

file.setExecutable(true, false);

file.setReadable(true, false);

file.setWritable(true, false);

//using PosixFilePermission to set file permissions 777

Set<PosixFilePermission> perms = new HashSet<PosixFilePermission>();

//add owners permission

perms.add(PosixFilePermission.OWNER\_READ);

perms.add(PosixFilePermission.OWNER\_WRITE);

perms.add(PosixFilePermission.OWNER\_EXECUTE);

//add group permissions

perms.add(PosixFilePermission.GROUP\_READ);

perms.add(PosixFilePermission.GROUP\_WRITE);

perms.add(PosixFilePermission.GROUP\_EXECUTE);

//add others permissions

perms.add(PosixFilePermission.OTHERS\_READ);

perms.add(PosixFilePermission.OTHERS\_WRITE);

perms.add(PosixFilePermission.OTHERS\_EXECUTE);

Files.setPosixFilePermissions(Paths.get("/Users/run.sh"), perms);

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *In Java, How many ways you can take input from the console?***

In Java, there are three different ways for reading input from the user in the command line environment(console).

**1. Using Buffered Reader Class**: This method is used by wrapping the System.in (standard input stream) in an InputStreamReader which is wrapped in a BufferedReader, we can read input from the user in the command line.

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Test

{

public static void main(String[] args) throws IOException {

//Enter data using BufferReader

BufferedReader reader =

new BufferedReader(new InputStreamReader(System.in));

// Reading data using readLine

String name = reader.readLine();

// Printing the read line

System.out.println(name);

}

}

**2. Using Scanner Class**: The main purpose of the Scanner class is to parse primitive types and strings using regular expressions, however it is also can be used to read input from the user in the command line.

import java.util.Scanner;

class GetInputFromUser

{

public static void main(String args[]) {

// Using Scanner for Getting Input from User

Scanner in = new Scanner(System.in);

String s = in.nextLine();

System.out.println("You entered string "+s);

int a = in.nextInt();

System.out.println("You entered integer "+a);

float b = in.nextFloat();

System.out.println("You entered float "+b);

}

}

**3. Using Console Class**: It has been becoming a preferred way for reading user’s input from the command line. In addition, it can be used for reading password-like input without echoing the characters entered by the user; the format string syntax can also be used (like System.out.printf()).

public class Sample

{

public static void main(String[] args) {

// Using Console to input data from user

String name = System.console().readLine();

System.out.println(name);

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *How can you avoid serialization in child class if the base class is implementing the Serializable interface?***

If superClass has implemented Serializable that means subclass is also Serializable (as subclass always inherits all features from its parent class), for avoiding Serialization in sub-class we can define **writeObject()** method and throw **NotSerializableException()** from there as done below.

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.NotSerializableException;

import java.io.ObjectOutput;

import java.io.ObjectOutputStream;

import java.io.OutputStream;

import java.io.Serializable;

class Super implements Serializable {

private static final long serialVersionUID = 1L;

}

class Sub extends Super {

private static final long serialVersionUID = 1L;

private Integer id;

public Sub(Integer id) {

this.id = id;

}

@Override

public String toString() {

return "Employee [id=" + id + "]";

}

/\*

\* define how Serialization process will write objects.

\*/

private void writeObject(ObjectOutputStream os) throws NotSerializableException {

throw new NotSerializableException("This class cannot be Serialized");

}

}

public class SerializeDeserialize {

public static void main(String[] args) {

Sub object1 = new Sub(8);

try {

OutputStream fout = new FileOutputStream("ser.txt");

ObjectOutput oout = new ObjectOutputStream(fout);

System.out.println("Serialization process has started, serializing objects...");

oout.writeObject(object1);

fout.close();

oout.close();

System.out.println("Object Serialization completed.");

} catch (IOException e) {

e.printStackTrace();

}

}

}

Output

Serialization process has started, serializing objects...

java.io.NotSerializableException: This class cannot be Serialized

at SerDeser11throwNotSerExc.Sub.writeObject(SerializeConstructorCheck.java:35)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)

at java.lang.reflect.Method.invoke(Unknown Source)

at java.io.ObjectStreamClass.invokeWriteObject(Unknown Source)

at java.io.ObjectOutputStream.writeSerialData(Unknown Source)

at java.io.ObjectOutputStream.writeOrdinaryObject(Unknown Source)

at java.io.ObjectOutputStream.writeObject0(Unknown Source)

at java.io.ObjectOutputStream.writeObject(Unknown Source)

at SerDeser11throwNotSerExc.SerializeConstructorCheck.main(SerializeConstructorCheck.java:51)

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is the difference between Serializable and Externalizable interface?***

| **Sl.No** | **SERIALIZABLE** | **EXTERNALIZABLE** |
| --- | --- | --- |
| 01. | Serializable is a marker interface i.e. does not contain any method. | Externalizable interface contains two methods writeExternal() and readExternal() which implementing classes MUST override. |
| 02. | Serializable interface pass the responsibility of serialization to JVM and it’s default algorithm. | Externalizable provides control of serialization logic to programmer – to write custom logic. |
| 03. | Mostly, default serialization is easy to implement, but has higher performance cost. | Serialization done using Externalizable, add more responsibility to programmer but often result in better performance. |
| 04. | It’s hard to analyze and modify class structure because any change may break the serialization. | It’s more easy to analyze and modify class structure because of complete control over serialization logic. |
| 05. | Default serialization does not call any class constructor. | A public no-arg constructor is required while using Externalizable interface. |

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What are the ways to instantiate the Class class?***

**1. Using new keyword**

MyObject object = new MyObject();

**2. Using Class.forName()**

MyObject object = (MyObject) Class.forName("subin.rnd.MyObject").newInstance();

**3. Using clone()**

MyObject anotherObject = new MyObject();

MyObject object = (MyObject) anotherObject.clone();

**4. Using object deserialization**

ObjectInputStream inStream = new ObjectInputStream(anInputStream );

MyObject object = (MyObject) inStream.readObject();

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is the purpose of using javap?***

The javap command displays information about the fields, constructors and methods present in a class file. The javap command (also known as the Java Disassembler) disassembles one or more class files.

class Simple {

public static void main(String args[]) {

System.out.println("Hello World");

}

}

cmd> javap Simple.class

Output

Compiled from ".java"

class Simple {

Simple();

public static void main(java.lang.String[]);

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What are autoboxing and unboxing? When does it occur?***

The automatic conversion of primitive data types into its equivalent Wrapper type is known as boxing and opposite operation is known as unboxing.

Example: Autoboxing

class BoxingExample1 {

public static void main(String args[]) {

int a = 50;

Integer a2 = new Integer(a); //Boxing

Integer a3 = 5; //Boxing

System.out.println(a2+" "+a3);

}

}

Example: Unboxing

class UnboxingExample1 {

public static void main(String args[]) {

Integer i = new Integer(50);

int a = i;

System.out.println(a);

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is a native method?***

A native method is a Java method (either an instance method or a class method) whose implementation is also written in another programming language such as C/C++. Moreover, a method marked as native cannot have a body and should end with a semicolon:

**Main.java**

public class Main {

public native int intMethod(int i);

public static void main(String[] args) {

System.loadLibrary("Main");

System.out.println(new Main().intMethod(2));

}

}

**Main.c**

#include <jni.h>

#include "Main.h"

JNIEXPORT jint JNICALL Java\_Main\_intMethod(

JNIEnv \*env, jobject obj, jint i) {

return i \* i;

}

**Compile and run**

javac Main.java

javah -jni Main

gcc -shared -fpic -o libMain.so -I${JAVA\_HOME}/include \

-I${JAVA\_HOME}/include/linux Main.c

java -Djava.library.path=. Main

Output

4

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is immutable object? Can you write immutable object?***

Immutable objects are objects that don't change. A Java immutable object must have all its fields be internal, private final fields. It must not implement any setters. It needs a constructor that takes a value for every single field.

**Creating an Immutable Object**

* Do not add any setter method
* Declare all fields final and private
* If a field is a mutable object create defensive copies of it for getter methods
* If a mutable object passed to the constructor must be assigned to a field create a defensive copy of it
* Don't allow subclasses to override methods.

public class DateContainer {

private final Date date;

public DateContainer() {

this.date = new Date();

}

public Date getDate() {

return new Date(date.getTime());

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *The difference between Inheritance and Composition?***

Though both Inheritance and Composition provides code reusablility, main difference between Composition and Inheritance in Java is that Composition allows reuse of code without extending it but for Inheritance you must extend the class for any reuse of code or functionality. Inheritance is an **"is-a"** relationship. Composition is a **"has-a"**.

Example: Inheritance

class Fruit {

//...

}

class Apple extends Fruit {

//...

}

Example: Composition

class Fruit {

//...

}

class Apple {

private Fruit fruit = new Fruit();

//...

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *The difference between DOM and SAX parser in Java?***

DOM and SAX parser are extensively used to read and parse XML file in java and have their own set of advantage and disadvantage.

|  | **DOM (Document Object Model)** | **Parser SAX (Simple API for XML) Parser** |
| --- | --- | --- |
| Abbreviation | DOM stands for Document Object Model | SAX stands for Simple API for XML Parsing |
| type | Load entire memory and keep in tree structure | event based parser |
| size of Document | good for smaller size | good to choose for larger size of file. |
| Load | Load entire document in memory | does not load entire document. |
| suitable | better suitable for smaller and efficient memory | SAX is suitable for larger XML doc |

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is the difference between creating String as new() and literal?***

When you create String object using new() operator, it always create a new object in heap memory. On the other hand, if you create object using String literal syntax e.g. "Java", it may return an existing object from String pool (a cache of String object in Perm gen space, which is now moved to heap space in recent Java release), if it's already exists. Otherwise it will create a new string object and put in string pool for future re-use.

String a = "abc";

String b = "abc";

System.out.println(a == b); // true

String c = new String("abc");

String d = new String("abc");

System.out.println(c == d); // false

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *How can we create an immutable class in Java?***

Immutable class means that once an object is created, we cannot change its content. In Java, all the wrapper classes (like Integer, Boolean, Byte, Short) and String class is immutable.

**Rules to create immutable classes**

* The class must be declared as final (So that child classes can’t be created)
* Data members in the class must be declared as final (So that we can’t change the value of it after object creation)
* A parameterized constructor
* Getter method for all the variables in it
* No setters(To not have the option to change the value of the instance variable)

public final class Employee {

final String pancardNumber;

public Employee(String pancardNumber) {

this.pancardNumber = pancardNumber;

}

public String getPancardNumber() {

return pancardNumber;

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is difference between String, StringBuffer and StringBuilder?***

**Mutability Difference:** String is **immutable**, if you try to alter their values, another object gets created, whereas StringBuffer and StringBuilder are **mutable** so they can change their values.

**Thread-Safety Difference:** The difference between StringBuffer and StringBuilder is that StringBuffer is thread-safe. So when the application needs to be run only in a single thread then it is better to use StringBuilder. StringBuilder is more efficient than StringBuffer.

Example: StringBuffer

public class BufferTest{

public static void main(String[] args){

StringBuffer buffer=new StringBuffer("Hello");

buffer.append(" World");

System.out.println(buffer);

}

}

Example: StringBuilder

public class BuilderTest{

public static void main(String[] args){

StringBuilder builder=new StringBuilder("Hello");

builder.append(" World");

System.out.println(builder);

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is a Memory Leak? How can a memory leak appear in garbage collected language?***

The standard definition of a memory leak is a scenario that occurs when **objects are no longer being used by the application, but the Garbage Collector is unable to remove them from working memory** – because they’re still being referenced. As a result, the application consumes more and more resources – which eventually leads to a fatal OutOfMemoryError.

Some tools that do memory management to identifies useless objects or memeory leaks like:

* [HP OpenView](https://support.hpe.com/hpsc/doc/public/display?docId=emr_na-c00990822&docLocale=en_US)
* [HP JMETER](https://h20392.www2.hpe.com/portal/swdepot/displayProductInfo.do?productNumber=HPJMETER)
* [JProbe](http://www.javaperformancetuning.com/tools/jprobe/index.shtml)
* [IBM Tivoli](https://www.ibm.com/support/knowledgecenter/en/SSTFXA_6.3.0/com.ibm.itm.doc_6.3/install/itm_over.htm)

// Java Program to illustrate memory leaks

import java.util.Vector;

public class MemoryLeaksDemo

{

public static void main(String[] args) {

Vector v = new Vector(214444);

Vector v1 = new Vector(214744444);

Vector v2 = new Vector(214444);

System.out.println("Memory Leaks Example");

}

}

Output

Exception in thread "main" java.lang.OutOfMemoryError: Java heap space exceed

**Types of Memory Leaks in Java**

* Memory Leak through static Fields
* Unclosed Resources/connections
* Adding Objects With no hashCode() and equals() Into a HashSet
* Inner Classes that Reference Outer Classes
* Through finalize() Methods
* Calling String.intern() on Long String

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *Why String is popular HashMap key in Java?***

Since String is immutable, its hashcode is cached at the time of creation and it doesn’t need to be calculated again. This makes it a great candidate for key in a Map and its processing is fast than other HashMap key objects. This is why String is mostly used Object as HashMap keys.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is difference between Error and Exception?***

| **BASIS FOR COMPARISON** | **ERROR** | **EXCEPTION** |
| --- | --- | --- |
| Basic | An error is caused due to lack of system resources. | An exception is caused because of the code. |
| Recovery | An error is irrecoverable. | An exception is recoverable. |
| Keywords | There is no means to handle an error by the program code. | Exceptions are handled using three keywords "try", "catch", and "throw". |
| Consequences | As the error is detected the program will terminated abnormally. | As an exception is detected, it is thrown and caught by the "throw" and "catch" keywords correspondingly. |
| Types | Errors are classified as unchecked type. | Exceptions are classified as checked or unchecked type. |
| Package | In Java, errors are defined "java.lang.Error" package. | In Java, an exceptions are defined in"java.lang.Exception". |
| Example | OutOfMemory, StackOverFlow. | Checked Exceptions: NoSuchMethod, ClassNotFound.Unchecked Exceptions: NullPointer, IndexOutOfBounds. |

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *Explain about Exception Propagation?***

An exception is first thrown from the top of the stack and if it is not caught, it drops down the call stack to the previous method, If not caught there, the exception again drops down to the previous method, and so on until they are caught or until they reach the very bottom of the call stack. This is called exception propagation.

class TestExceptionPropagation {

void m() {

int data = 50/0;

}

void n() {

m();

}

void p() {

try {

n();

} catch(Exception e) {

System.out.println("exception handled");

}

}

public static void main(String args[]) {

TestExceptionPropagation obj = new TestExceptionPropagation();

obj.p();

System.out.println("Normal Flow...");

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What are different scenarios causing "Exception in thread main"?***

Some of the common main thread exception are as follows:

* **Exception in thread main java.lang.UnsupportedClassVersionError**: This exception comes when your java class is compiled from another JDK version and you are trying to run it from another java version.
* **Exception in thread main java.lang.NoClassDefFoundError**: There are two variants of this exception. The first one is where you provide the class full name with .class extension. The second scenario is when Class is not found.
* **Exception in thread main java.lang.NoSuchMethodError: main**: This exception comes when you are trying to run a class that doesn’t have main method.
* **Exception in thread "main" java.lang.ArithmeticException**: Whenever any exception is thrown from main method, it prints the exception is console. The first part explains that exception is thrown from main method, second part prints the exception class name and then after a colon, it prints the exception message.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What are the differences between throw and throws?***

**Throw** keyword is used in the method body to throw an exception, while **throws** is used in method signature to declare the exceptions that can occur in the statements present in the method.

**Throw Example**

public class ThrowExample {

void checkAge(int age) {

if(age < 18)

throw new ArithmeticException("Not Eligible for voting");

else

System.out.println("Eligible for voting");

}

public static void main(String args[]) {

ThrowExample obj = new ThrowExample();

obj.checkAge(13);

System.out.println("End Of Program");

}

}

Output

Exception in thread "main" java.lang.ArithmeticException:

Not Eligible for voting

at Example1.checkAge(Example1.java:4)

at Example1.main(Example1.java:10)

**Throws Example**

public class ThrowsExample {

int division(int a, int b) throws ArithmeticException {

int t = a/b;

return t;

}

public static void main(String args[]) {

ThrowsExample obj = new ThrowsExample();

try {

System.out.println(obj.division(15,0));

}

catch(ArithmeticException e) {

System.out.println("You shouldn't divide number by zero");

}

}

}

Output

You shouldn't divide number by zero

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *The difference between Serial and Parallel Garbage Collector?***

**Serial Garbage Collector**

Serial garbage collector works by holding all the application threads. It is designed for the single-threaded environments. It uses just a single thread for garbage collection. The way it works by freezing all the application threads while doing garbage collection may not be suitable for a server environment. It is best suited for simple command-line programs.

Turn on the -XX:+UseSerialGC JVM argument to use the serial garbage collector.

**Parallel Garbage Collector**

Parallel garbage collector is also called as throughput collector. It is the default garbage collector of the JVM. Unlike serial garbage collector, this uses multiple threads for garbage collection. Similar to serial garbage collector this also freezes all the application threads while performing garbage collection.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is difference between WeakReference and SoftReference in Java?***

In Java there are four types of references differentiated on the way by which they are garbage collected.

* Strong References
* Weak References
* Soft References
* Phantom References

**Strong References**: This is the default type/class of Reference Object. Any object which has an active strong reference are not eligible for garbage collection. The object is garbage collected only when the variable which was strongly referenced points to null.

MyClass obj = new MyClass();

**Weak References**: Weak Reference Objects are not the default type/class of Reference Object and they should be explicitly specified while using them.

//Java Code to illustrate Weak reference

import java.lang.ref.WeakReference;

class MainClass

{

public void message() {

System.out.println("Weak References Example");

}

}

public class Example

{

public static void main(String[] args) {

// Strong Reference

MainClass g = new MainClass();

g.message();

// Creating Weak Reference to MainClass-type object to which 'g'

// is also pointing.

WeakReference<MainClass> weakref = new WeakReference<MainClass>(g);

g = null;

g = weakref.get();

g.message();

}

}

**Soft References**: In Soft reference, even if the object is free for garbage collection then also its not garbage collected, until JVM is in need of memory badly.The objects gets cleared from the memory when JVM runs out of memory.To create such references java.lang.ref.SoftReference class is used.

//Java Code to illustrate Weak reference

import java.lang.ref.SoftReference;

class MainClass

{

public void message() {

System.out.println("Weak References Example");

}

}

public class Example

{

public static void main(String[] args) {

// Strong Reference

MainClass g = new MainClass();

g.message();

// Creating Weak Reference to MainClass-type object to which 'g'

// is also pointing.

SoftReference<MainClass> softref = new SoftReference<MainClass>(g);

g = null;

g = softref.get();

g.message();

}

}

**Phantom References**: The objects which are being referenced by phantom references are eligible for garbage collection. But, before removing them from the memory, JVM puts them in a queue called ‘reference queue’ . They are put in a reference queue after calling finalize() method on them.To create such references java.lang.ref.PhantomReference class is used.

//Java Code to illustrate Weak reference

import java.lang.ref.\*;

class MainClass

{

public void message() {

System.out.println("Phantom References Example");

}

}

public class Example

{

public static void main(String[] args) {

// Strong Reference

MainClass g = new MainClass();

g.message();

// Creating Phantom Reference to MainClass-type object to which 'g'

// is also pointing.

PhantomReference<MainClass> phantomRef = null;

phantomRef = new PhantomReference<MainClass>(g,refQueue);

g = null;

g = phantomRef.get();

g.message();

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is a compile time constant in Java? What is the risk of using it?***

If a primitive type or a string is defined as a constant and the value is known at compile time, the compiler replaces the constant name everywhere in the code with its value. This is called a compile-time constant.

**Compile time constant must be:**

* declared final
* primitive or String
* initialized within declaration
* initialized with constant expression

They are replaced with actual values at compile time because compiler know their value up-front and also knows that it cannot be changed during run-time.

private final int x = 10;

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *How bootstrap class loader works in java?***

Bootstrap **ClassLoader** is repsonsible for loading standard JDK classs files from **rt.jar** and it is parent of all class loaders in java. There are three types of built-in ClassLoader in Java:

**1. Bootstrap Class Loader** – It loads JDK internal classes, typically loads rt.jar and other core classes for example java.lang.\* package classes

**2. Extensions Class Loader** – It loads classes from the JDK extensions directory, usually $JAVA\_HOME/lib/ext directory.

**3. System Class Loader** – It loads classes from the current classpath that can be set while invoking a program using -cp or -classpath command line options.

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\* Java program to demonstrate How ClassLoader works in Java

\*

\*\*/

public class ClassLoaderTest {

public static void main(String args[]) {

try {

//printing ClassLoader of this class

System.out.println("ClassLoader : "+ ClassLoaderTest.class.getClassLoader());

//trying to explicitly load this class again using Extension class loader

Class.forName("Explicitly load class", true

, ClassLoaderTest.class.getClassLoader().getParent());

} catch (ClassNotFoundException ex) {

Logger.getLogger(ClassLoaderTest.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *Why string is immutable in java?***

The string is Immutable in Java because String objects are cached in String pool. Since cached String literals are shared between multiple clients there is always a risk, where one client's action would affect all another client.

Since string is immutable it can safely share between many threads and avoid any synchronization issues in java.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is Java String Pool?***

String Pool in java is a pool of Strings stored in Java Heap Memory. String pool helps in saving a lot of space for Java Runtime although it takes more time to create the String.

When we use double quotes to create a String, it first looks for String with the same value in the String pool, if found it just returns the reference else it creates a new String in the pool and then returns the reference. However using **new** operator, we force String class to create a new String object in heap space.

/\*\*

\* Java program to illustrate String Pool

\*

\*\*/

public class StringPool {

public static void main(String[] args) {

String s1 = "Java";

String s2 = "Java";

String s3 = new String("Java");

System.out.println("s1 == s2 :" +(s1==s2)); // true

System.out.println("s1 == s3 :" +(s1==s3)); // false

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *How Garbage collector algorithm works?***

Garbage collection works on **Mark** and **Sweep** algorithm. In Mark phase it detects all the unreachable objects and Sweep phase it reclaim the heap space used by the garbage objects and make the space available again to the program.

There are methods like System.gc() and Runtime.gc() which is used to send request of Garbage collection to JVM but it’s not guaranteed that garbage collection will happen. If there is no memory space for creating a new object in Heap Java Virtual Machine throws OutOfMemoryError or java.lang.OutOfMemoryError heap space

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *How to create marker interface?***

An interface with no methods is known as marker or tagged interface. It provides some useful information to JVM/compiler so that JVM/compiler performs some special operations on it. It is used for better readability of code. Example: **Serializable, Clonnable** etc.

Syntax:

public interface Interface\_Name {

}

Example:

/\*\*

\* Java program to illustrate Maker Interface

\*

\*\*/

interface Marker { }

class A implements Marker {

//do some task

}

class Main {

public static void main(String[] args) {

A obj = new A();

if (obj instanceOf Marker){

// do some task

}

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *How serialization works in java?***

Serialization is a mechanism of converting the state of an object into a byte stream. Deserialization is the reverse process where the byte stream is used to recreate the actual Java object in memory. This mechanism is used to persist the object.

Example:

/\*\*

\* Serialization and Deserialization

\* example of a Java object

\*

\*\*/

import java.io.\*;

class Employee implements Serializable {

private static final long serialversionUID =

129348938L;

transient int a;

static int b;

String name;

int age;

// Default constructor

public Employee(String name, int age, int a, int b) {

this.name = name;

this.age = age;

this.a = a;

this.b = b;

}

}

public class SerialExample {

public static void printdata(Employee object1) {

System.out.println("name = " + object1.name);

System.out.println("age = " + object1.age);

System.out.println("a = " + object1.a);

System.out.println("b = " + object1.b);

}

public static void main(String[] args) {

Employee object = new Employee("ab", 20, 2, 1000);

String filename = "shubham.txt";

// Serialization

try {

// Saving of object in a file

FileOutputStream file = new FileOutputStream(filename);

ObjectOutputStream out = new ObjectOutputStream(file);

// Method for serialization of object

out.writeObject(object);

out.close();

file.close();

System.out.println("Object has been serialized\n"

+ "Data before Deserialization.");

printdata(object);

// value of static variable changed

object.b = 2000;

}

catch (IOException ex) {

System.out.println("IOException is caught");

}

object = null;

// Deserialization

try {

// Reading the object from a file

FileInputStream file = new FileInputStream(filename);

ObjectInputStream in = new ObjectInputStream(file);

// Method for deserialization of object

object = (Employee)in.readObject();

in.close();

file.close();

System.out.println("Object has been deserialized\n"

+ "Data after Deserialization.");

printdata(object);

System.out.println("z = " + object1.z);

}

catch (IOException ex) {

System.out.println("IOException is caught");

}

catch (ClassNotFoundException ex) {

System.out.println("ClassNotFoundException is caught");

}

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What are the various ways to load a class in Java?***

**a). Creating a reference**:

SomeClass someInstance = null;

**b). Using Class.forName(String)**:

Class.forName("SomeClass");

**c). Using SystemClassLoader()**:

ClassLoader.getSystemClassLoader().loadClass("SomeClass");

**d). Using Overloaded Class.forName()**:

Class.forName(String name, boolean initialize, ClassLoader loader);

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *Java Program to Implement Singly Linked List?***

The singly linked list is a linear data structure in which each element of the list contains a pointer which points to the next element in the list. Each element in the singly linked list is called a node. Each node has two components: data and a pointer next which points to the next node in the list.

Example:

public class SinglyLinkedList {

// Represent a node of the singly linked list

class Node{

int data;

Node next;

public Node(int data) {

this.data = data;

this.next = null;

}

}

// Represent the head and tail of the singly linked list

public Node head = null;

public Node tail = null;

// addNode() will add a new node to the list

public void addNode(int data) {

// Create a new node

Node newNode = new Node(data);

// Checks if the list is empty

if(head == null) {

// If list is empty, both head and tail will point to new node

head = newNode;

tail = newNode;

}

else {

// newNode will be added after tail such that tail's next will point to newNode

tail.next = newNode;

// newNode will become new tail of the list

tail = newNode;

}

}

// display() will display all the nodes present in the list

public void display() {

// Node current will point to head

Node current = head;

if(head == null) {

System.out.println("List is empty");

return;

}

System.out.println("Nodes of singly linked list: ");

while(current != null) {

// Prints each node by incrementing pointer

System.out.print(current.data + " ");

current = current.next;

}

System.out.println();

}

public static void main(String[] args) {

SinglyLinkedList sList = new SinglyLinkedList();

// Add nodes to the list

sList.addNode(10);

sList.addNode(20);

sList.addNode(30);

sList.addNode(40);

// Displays the nodes present in the list

sList.display();

}

}

**Output:**

Nodes of singly linked list:

10 20 30 40

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *While overriding a method can you throw another exception or broader exception?***

If a method declares to throw a given exception, the overriding method in a subclass can only declare to throw that exception or its subclass. This is because of polymorphism.

Example:

class A {

public void message() throws IOException {..}

}

class B extends A {

@Override

public void message() throws SocketException {..} // allowed

@Override

public void message() throws SQLException {..} // NOT allowed

public static void main(String args[]) {

A a = new B();

try {

a.message();

} catch (IOException ex) {

// forced to catch this by the compiler

}

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is checked, unchecked exception and errors?***

**1. Checked Exception**:

* These are the classes that extend **Throwable** except **RuntimeException** and **Error**.
* They are also known as compile time exceptions because they are checked at **compile time**, meaning the compiler forces us to either handle them with try/catch or indicate in the function signature that it **throws** them and forcing us to deal with them in the caller.
* They are programmatically recoverable problems which are caused by unexpected conditions outside the control of the code (e.g. database down, file I/O error, wrong input, etc).
* Example: **IOException, SQLException** etc.

import java.io.\*;

class Main {

public static void main(String[] args) {

FileReader file = new FileReader("C:\\assets\\file.txt");

BufferedReader fileInput = new BufferedReader(file);

for (int counter = 0; counter < 3; counter++)

System.out.println(fileInput.readLine());

fileInput.close();

}

}

output:

Exception in thread "main" java.lang.RuntimeException: Uncompilable source code -

unreported exception java.io.FileNotFoundException; must be caught or declared to be

thrown

at Main.main(Main.java:5)

After adding IOException

import java.io.\*;

class Main {

public static void main(String[] args) throws IOException {

FileReader file = new FileReader("C:\\assets\\file.txt");

BufferedReader fileInput = new BufferedReader(file);

for (int counter = 0; counter < 3; counter++)

System.out.println(fileInput.readLine());

fileInput.close();

}

}

output:

Output: First three lines of file “C:\assets\file.txt”

**2. Unchecked Exception**:

* The classes that extend **RuntimeException** are known as unchecked exceptions.
* Unchecked exceptions are not checked at compile-time, but rather at **runtime**, hence the name.
* They are also programmatically recoverable problems but unlike checked exception they are caused by faults in code flow or configuration.
* Example: **ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException** etc.

class Main {

public static void main(String args[]) {

int x = 0;

int y = 10;

int z = y/x;

}

}

Output:

Exception in thread "main" java.lang.ArithmeticException: / by zero

at Main.main(Main.java:5)

Java Result: 1

**3. Error**:

**Error** refers to an irrecoverable situation that is not being handled by a **try/catch**.  
Example: **OutOfMemoryError, VirtualMachineError, AssertionError** etc.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is difference between ClassNotFoundException and NoClassDefFoundError?***

ClassNotFoundException and NoClassDefFoundError occur when a particular class is not found at runtime. However, they occur at different scenarios.

ClassNotFoundException is an exception that occurs when you try to load a class at run time using Class.forName() or loadClass() methods and mentioned classes are not found in the classpath.

NoClassDefFoundError is an error that occurs when a particular class is present at compile time, but was missing at run time.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What do we mean by weak reference?***

In Java there are four types of references differentiated on the way by which they are garbage collected.

1. Strong Reference
2. Weak Reference
3. Soft Reference
4. Phantom Reference

**1. Strong Reference**: This is the default type/class of Reference Object. Any object which has an active strong reference are not eligible for garbage collection. The object is garbage collected only when the variable which was strongly referenced points to null.

StrongReferenceClass obj = new StrongReferenceClass();

Here obj object is strong reference to newly created instance of MyClass, currently obj is active object so can't be garbage collected.

**2. Weak Reference**: A weakly referenced object is cleared by the Garbage Collector when it’s weakly reachable. Weak reachability means that an object has neither strong nor soft references pointing to it. The object can be reached only by traversing a weak reference. To create such references java.lang.ref.WeakReference class is used.

/\*\*

\* Java Code to illustrate Weak reference

\*

\*\*/

import java.lang.ref.WeakReference;

class WeakReferenceExample {

public void message() {

System.out.println("Weak Reference Example!");

}

}

public class MainClass {

public static void main(String[] args) {

// Strong Reference

WeakReferenceExample obj = new WeakReferenceExample();

obj.message();

// Creating Weak Reference to WeakReferenceExample-type object to which 'obj'

// is also pointing.

WeakReference<WeakReferenceExample> weakref = new WeakReference<WeakReferenceExample>(obj);

obj = null; // is available for garbage collection.

obj = weakref.get();

obj.message();

}

}

Output

Weak Reference Example!

Weak Reference Example!

**3. Soft Reference**: In Soft reference, even if the object is free for garbage collection then also its not garbage collected, until JVM is in need of memory badly.The objects gets cleared from the memory when JVM runs out of memory.To create such references java.lang.ref.SoftReference class is used.

/\*\*

\* Java Code to illustrate Soft reference

\*

\*\*/

import java.lang.ref.SoftReference;

class SoftReferenceExample {

public void message() {

System.out.println("Soft Reference Example!");

}

}

public class MainClass {

public static void main(String[] args) {

// Soft Reference

SoftReferenceExample obj = new SoftReferenceExample();

obj.message();

// Creating Soft Reference to SoftReferenceExample-type object to which 'obj'

// is also pointing.

SoftReference<SoftReferenceExample> softref = new SoftReference<SoftReferenceExample>(obj);

obj = null; // is available for garbage collection.

obj = softref.get();

obj.message();

}

}

Output

Soft Reference Example!

Soft Reference Example!

**4. Phantom Reference**: The objects which are being referenced by phantom references are eligible for garbage collection. But, before removing them from the memory, JVM puts them in a queue called **reference queue**. They are put in a reference queue after calling finalize() method on them. To create such references java.lang.ref.PhantomReference class is used.

/\*\*

\* Code to illustrate Phantom reference

\*

\*\*/

import java.lang.ref.\*;

class PhantomReferenceExample {

public void message() {

System.out.println("Phantom Reference Example!");

}

}

public class MainClass {

public static void main(String[] args) {

//Strong Reference

PhantomReferenceExample obj = new PhantomReferenceExample();

obj.message();

//Creating reference queue

ReferenceQueue<PhantomReferenceExample> refQueue = new ReferenceQueue<PhantomReferenceExample>();

//Creating Phantom Reference to PhantomReferenceExample-type object to which 'obj'

//is also pointing.

PhantomReference<PhantomReferenceExample> phantomRef = null;

phantomRef = new PhantomReference<PhantomReferenceExample>(obj, refQueue);

obj = null;

obj = phantomRef.get(); //It always returns null

obj.message(); //It shows NullPointerException

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What do you mean Run time Polymorphism?***

Polymorphism in Java is a concept by which we can perform a single action in different ways.  
There are two types of polymorphism in java:

* **Static Polymorphism** also known as compile time polymorphism
* **Dynamic Polymorphism** also known as runtime polymorphism

Example: Static Polymorphism

class SimpleCalculator {

int add(int a, int b) {

return a + b;

}

int add(int a, int b, int c) {

return a + b + c;

}

}

public class MainClass

{

public static void main(String args[]) {

SimpleCalculator obj = new SimpleCalculator();

System.out.println(obj.add(10, 20));

System.out.println(obj.add(10, 20, 30));

}

}

Output

30

60

Example: Runtime polymorphism

class ABC {

public void myMethod() {

System.out.println("Overridden Method");

}

}

public class XYZ extends ABC {

public void myMethod() {

System.out.println("Overriding Method");

}

public static void main(String args[]) {

ABC obj = new XYZ();

obj.myMethod();

}

}

Output

Overriding Method

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *If I do not have Explicit constructor in parent class and having in child class, while calling the child constructor jvm automatically calls Implicit Constructor of parent class?***

If the subclass constructor does not specify which superclass constructor to invoke then the compiler will automatically call the accessible no-args constructor in the superclass.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What are the different types of JDBC Driver?***

JDBC Driver is a software component that enables java application to interact with the database.  
There are 4 types of JDBC drivers:

1. **JDBC-ODBC bridge driver**: The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. This is now discouraged because of thin driver.
2. **Native-API driver**: The Native API driver uses the client-side libraries of the database. The driver converts JDBC method calls into native calls of the database API. It is not written entirely in java.
3. **Network Protocol driver**: The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. It is fully written in java.
4. **Thin driver**: The thin driver converts JDBC calls directly into the vendor-specific database protocol. That is why it is known as thin driver. It is fully written in Java language.

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *How Encapsulation concept implemented in JAVA?***

Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as data hiding.

To achieve encapsulation in Java −

* Declare the variables of a class as private.
* Provide public setter and getter methods to modify and view the variables values.

Example:

public class EncapClass {

private String name;

public String getName() {

return name;

}

public void setName(String newName) {

name = newName;

}

}

public class MainClass {

public static void main(String args[]) {

EncapClass obj = new EncapClass();

obj.setName("Pradeep Kumar");

System.out.print("Name : " + obj.getName());

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *Do you know Generics? How did you used in your coding?***

Generics allows type (Integer, String, … etc and user defined types) to be a parameter to methods, classes and interfaces. For example, classes like HashSet, ArrayList, HashMap, etc use generics very well.

**Advantages**

* **Type-safety**: We can hold only a single type of objects in generics. It doesn't allow to store other objects.
* **Type Casting**: There is no need to typecast the object.
* **Compile-Time Checking**: It is checked at compile time so problem will not occur at runtime.

Example:

/\*\*

\* A Simple Java program to show multiple

\* type parameters in Java Generics

\*

\* We use < > to specify Parameter type

\*

\*\*/

class GenericClass<T, U> {

T obj1; // An object of type T

U obj2; // An object of type U

// constructor

GenericClass(T obj1, U obj2) {

this.obj1 = obj1;

this.obj2 = obj2;

}

// To print objects of T and U

public void print() {

System.out.println(obj1);

System.out.println(obj2);

}

}

// Driver class to test above

class MainClass {

public static void main (String[] args) {

GenericClass <String, Integer> obj =

new GenericClass<String, Integer>("Generic Class Example !", 100);

obj.print();

}

}

Output:

Generic Class Example !

100

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is difference between String, StringBuilder and StringBuffer?***

String is immutable, if you try to alter their values, another object gets created, whereas StringBuffer and StringBuilder are mutable so they can change their values.

The difference between StringBuffer and StringBuilder is that StringBuffer is thread-safe. So when the application needs to be run only in a single thread then it is better to use StringBuilder. StringBuilder is more efficient than StringBuffer.

**Situations**:

* If your string is not going to change use a String class because a String object is immutable.
* If your string can change (example: lots of logic and operations in the construction of the string) and will only be accessed from a single thread, using a StringBuilder is good enough.
* If your string can change, and will be accessed from multiple threads, use a StringBuffer because StringBuffer is synchronous so you have thread-safety.

Example:

class StringExample {

// Concatenates to String

public static void concat1(String s1) {

s1 = s1 + "World";

}

// Concatenates to StringBuilder

public static void concat2(StringBuilder s2) {

s2.append("World");

}

// Concatenates to StringBuffer

public static void concat3(StringBuffer s3) {

s3.append("World");

}

public static void main(String[] args) {

String s1 = "Hello";

concat1(s1); // s1 is not changed

System.out.println("String: " + s1);

StringBuilder s2 = new StringBuilder("Hello");

concat2(s2); // s2 is changed

System.out.println("StringBuilder: " + s2);

StringBuffer s3 = new StringBuffer("Hello");

concat3(s3); // s3 is changed

System.out.println("StringBuffer: " + s3);

}

}

Output

String: Hello

StringBuilder: World

StringBuffer: World

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *How can we create a object of a class without using new operator?***

Different ways to create an object in Java

* **Using new Keyword**

class ObjectCreationExample{

String Owner;

}

public class MainClass {

public static void main(String[] args) {

// Here we are creating Object of JBT using new keyword

ObjectCreationExample obj = new ObjectCreationExample();

}

}

* **Using New Instance (Reflection)**

class CreateObjectClass {

static int j = 10;

CreateObjectClass() {

i = j++;

}

int i;

@Override

public String toString() {

return "Value of i :" + i;

}

}

class MainClass {

public static void main(String[] args) {

try {

Class cls = Class.forName("CreateObjectClass");

CreateObjectClass obj = (CreateObjectClass) cls.newInstance();

CreateObjectClass obj1 = (CreateObjectClass) cls.newInstance();

System.out.println(obj);

System.out.println(obj1);

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (InstantiationException e) {

e.printStackTrace();

} catch (IllegalAccessException e) {

e.printStackTrace();

}

}

}

* **Using Clone**

class CreateObjectWithClone implements Cloneable {

@Override

protected Object clone() throws CloneNotSupportedException {

return super.clone();

}

int i;

static int j = 10;

CreateObjectWithClone() {

i = j++;

}

@Override

public String toString() {

return "Value of i :" + i;

}

}

class MainClass {

public static void main(String[] args) {

CreateObjectWithClone obj1 = new CreateObjectWithClone();

System.out.println(obj1);

try {

CreateObjectWithClone obj2 = (CreateObjectWithClone) obj1.clone();

System.out.println(obj2);

} catch (CloneNotSupportedException e) {

e.printStackTrace();

}

}

}

* **Using ClassLoader**

class CreateObjectWithClassLoader {

static int j = 10;

CreateObjectWithClassLoader() {

i = j++;

}

int i;

@Override

public String toString() {

return "Value of i :" + i;

}

}

public class MainClass {

public static void main(String[] args) {

CreateObjectWithClassLoader obj = null;

try {

obj = (CreateObjectWithClassLoader) new MainClass().getClass()

.getClassLoader().loadClass("CreateObjectWithClassLoader").newInstance();

// Fully qualified classname should be used.

} catch (InstantiationException e) {

e.printStackTrace();

} catch (IllegalAccessException e) {

e.printStackTrace();

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

System.out.println(obj);

}

}

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What code coverage tools are you using for your project?***

* [Cobertura](https://cobertura.github.io/cobertura/)

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *Scenario of browser’s browsing history, where you need to store the browsing history, what data structure will you use.?***

* use stack

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *Scenario where in we have to download a big file by clicking on a link, how will you make sure that connections is reliable throughout?***

* use persistent MQueues

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What are methods of Object Class?***

The Object class is the parent class of all the classes in java by default.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public final Class getClass() | returns the Class class object of this object. The Class class can further be used to get the metadata of this class. |
| public int hashCode() | returns the hashcode number for this object. |
| public boolean equals(Object obj) | compares the given object to this object. |
| protected Object clone() throws CloneNotSupportedException | creates and returns the exact copy (clone) of this object. |
| public String toString() | returns the string representation of this object. |
| public final void notify() | wakes up single thread, waiting on this object's monitor. |
| public final void notifyAll() | wakes up all the threads, waiting on this object's monitor. |
| public final void wait(long timeout)throws InterruptedException | causes the current thread to wait for the specified milliseconds, until another thread notifies (invokes notify() or notifyAll() method). |
| public final void wait(long timeout,int nanos)throws InterruptedException | causes the current thread to wait for the specified milliseconds and nanoseconds, until another thread notifies (invokes notify() or notifyAll() method). |
| public final void wait()throws InterruptedException | causes the current thread to wait, until another thread notifies (invokes notify() or notifyAll() method). |
| protected void finalize()throws Throwable | is invoked by the garbage collector before object is being garbage collected. |

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**Q. *What is copyonwritearraylist in java?***

**Q. *How do you test static method?***

**Q. *How to do you test a method for an exception using JUnit?***

**Q. *Which unit testing libraries you have used for testing Java programs?***

**Q. *What is the difference between @Before and @BeforeClass annotation?***

**Q. *Can you explain Liskov Substitution principle?***

**Q. *Give me an example of design pattern which is based upon open closed principle?***

**Q. *What is Law of Demeter violation? Why it matters?***

**Q. *What is differences between External Iteration and Internal Iteration?***

*ToDo*

[**↥ back to top**](https://github.com/learning-zone/java-interview-questions)

**About**

1000+ Java Interview Questions

[learning-zone.github.io/java-interview-questions/](https://learning-zone.github.io/java-interview-questions/)

**Topics**

[java](https://github.com/topics/java) [design-pattern](https://github.com/topics/design-pattern) [jsp](https://github.com/topics/jsp) [jdbc](https://github.com/topics/jdbc) [multithreading](https://github.com/topics/multithreading) [hibernate](https://github.com/topics/hibernate) [collections](https://github.com/topics/collections) [servlet](https://github.com/topics/servlet) [java8](https://github.com/topics/java8) [java-programs](https://github.com/topics/java-programs)

**Resources**

[Readme](https://github.com/learning-zone/java-interview-questions#readme)

[**Releases**](https://github.com/learning-zone/java-interview-questions/releases)

No releases published

[**Packages**](https://github.com/users/learning-zone/packages?repo_name=java-interview-questions)

No packages published

[**Contributors**3](https://github.com/learning-zone/java-interview-questions/graphs/contributors)

* [**learning-zone** Pradeep Kumar](https://github.com/learning-zone)
* [**qilingit** Qilin Zhang](https://github.com/qilingit)
* [**pradeepkumar2** Pradeep Kumar](https://github.com/pradeepkumar2)

**Languages**

* [Java100.0%](https://github.com/learning-zone/java-interview-questions/search?l=java)
* © 2021 GitHub, Inc.
* [Terms](https://docs.github.com/en/github/site-policy/github-terms-of-service)
* [Privacy](https://docs.github.com/en/github/site-policy/github-privacy-statement)
* [Security](https://github.com/security)
* [Status](https://www.githubstatus.com/)
* [Docs](https://docs.github.com/)
* [Contact GitHub](https://support.github.com/)
* [Pricing](https://github.com/pricing)
* [API](https://docs.github.com/)
* [Training](https://services.github.com/)
* [Blog](https://github.blog/)
* [About](https://github.com/about)

Top of Form



Bottom of Form

* [Knowledge Base](https://www.javacodegeeks.com/)
* [Resources](https://www.javacodegeeks.com/resources)
* [About](https://www.javacodegeeks.com/about)
* [Deals](https://deals.javacodegeeks.com/)
* [Login](https://www.javacodegeeks.com/login?redirect_to=/java-interview-questions.html)
* [Register](https://www.javacodegeeks.com/register?redirect_to=/java-interview-questions.html)



* [Android](https://www.javacodegeeks.com/category/android)
* [Java](https://www.javacodegeeks.com/category/java)
* [JVM Languages](https://www.javacodegeeks.com/category/jvm-languages)
* [Soft Dev](https://www.javacodegeeks.com/category/software-development)
* [Web Development](https://www.javacodegeeks.com/category/web-development)
* [Agile](https://www.javacodegeeks.com/category/agile)
* [Career](https://www.javacodegeeks.com/category/career)
* [Comms](https://www.javacodegeeks.com/category/communications)
* [DevOps](https://www.javacodegeeks.com/category/devops)
* [Meta JCG](https://www.javacodegeeks.com/category/meta-jcg)



[Home](https://www.javacodegeeks.com/) » [Java](https://www.javacodegeeks.com/category/java) » [Core Java](https://www.javacodegeeks.com/category/java/core-java) » 150 Java Interview Questions and Answers – The ULTIMATE List (PDF Download)

About Sotirios-Efstathios Maneas



Sotirios-Efstathios (Stathis) Maneas is a postgraduate student at the Department of Informatics and Telecommunications of The National and Kapodistrian University of Athens. His main interests include distributed systems, web crawling, model checking, operating systems, programming languages and web applications.

[https://www.javacodegeeks.com/wp-content/themes/jarida/images/author_site.png](https://www.javacodegeeks.com/)

150 Java Interview Questions and Answers – The ULTIMATE List (PDF Download)

Posted by: [Sotirios-Efstathios Maneas](https://www.javacodegeeks.com/author/stathis-maneas) in [Core Java](https://www.javacodegeeks.com/category/java/core-java) April 7th, 2014 [154 Comments](https://www.javacodegeeks.com/java-interview-questions.html#comments) 47940 Views

Our Java Interview Questions and Answers collection is all about different types of questions that can be used in a Java interview, in order for the employer to test your skills in Java and object-oriented programming in general.

In the following sections we will discuss Java Interview Questions about object-oriented programming and its characteristics, general questions regarding Java and its functionality, collections in Java, garbage collectors, exception handling, Java applets, Swing, JDBC, Remote Method Invocation (RMI), Servlets and JSP.

Let’s go…!

Table Of Contents

[A.Object Oriented Programming (OOP)](https://www.javacodegeeks.com/java-interview-questions.html#1)

[B.General Questions about Java](https://www.javacodegeeks.com/java-interview-questions.html#2)

[C.Java Threads](https://www.javacodegeeks.com/java-interview-questions.html#3)

[D.Java Collections](https://www.javacodegeeks.com/java-interview-questions.html#4)

[E.Garbage Collectors](https://www.javacodegeeks.com/java-interview-questions.html#5)

[F.Exception Handling](https://www.javacodegeeks.com/java-interview-questions.html#6)

[G.Java Applets](https://www.javacodegeeks.com/java-interview-questions.html#7)

[H.Swing](https://www.javacodegeeks.com/java-interview-questions.html#8)

[I.JDBC](https://www.javacodegeeks.com/java-interview-questions.html#9)

[J.Remote Method Invocation (RMI)](https://www.javacodegeeks.com/java-interview-questions.html#10)

[K.Servlets](https://www.javacodegeeks.com/java-interview-questions.html#11)

[L.JSP](https://www.javacodegeeks.com/java-interview-questions.html#12)

A. Object Oriented Programming (OOP)

1. What is Java?

Java is a computer programming language that is concurrent, class-based and object-oriented. The advantages of object-oriented software development are shown below:

* Modular development of code, which leads to easy maintenance and modification.
* Reusability of code.
* Improved reliability and flexibility of code.
* Increased understanding of code.

2. What are the concepts of OOP?

Object Oriented Programming (OOP) includes:

* Abstraction
* Encapsulation
* Polymorphism
* Inheritance
* Predefined types must be objects
* User defined types must be objects
* Operations must be performed by sending messages to objects

3. Mention some features of Java

Some of the features which play important role in the popularity of java are as follows:

* Object-Oriented
* Platform independent
* High Performance
* Multithreaded
* Portable
* Secure

Sample code for Helloworld in java is shown below:

*Hello World*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | public class Helloworld{    public static void main(String args[])  {      System.out.println("Hello World");  }    } |

4. Is Java 100% Object-oriented?

Not 100%. Java does not satisfy all the OOP conditions (predefined types must be objects) because it uses eight primitive data types(Boolean, byte, char, int, float, double, long, short) which are not objects.

5. What is Abstraction?

[Abstraction](https://www.javacodegeeks.com/2014/07/abstraction-in-java.html) is the process of separating ideas from specific instances and thus, develop classes in terms of their own functionality, instead of their implementation details. Java supports the creation and existence of abstract classes that expose interfaces, without including the actual implementation of all methods. The abstraction technique aims to separate the implementation details of a class from its behavior.

Abstract class Person is presented below. It has an abstract method getName.

*Abstract Class Person*

|  |  |
| --- | --- |
| 1  2  3  4 | public abstract class Person  {      public abstract String getName();  } |

Employee class extends the Abstract class Person. The method getName returns the name attribute of the employee.

*Employee Class*

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | public class Employee extends Person  {      private String name;        public Employee(String name)      {        this.name = name;      }      public String getName()      {         return this.name;      }      public static void main (String args[])      {          Employee employee = new Employee("John Wilson");            System.out.println("Employee's Name "+ employee.getName());            Person person = new Employee("Thomas Smith");            System.out.println("Employee-Person's Name "+ person.getName());          }  } |

6. What is Encapsulation?

[Encapsulation](https://www.javacodegeeks.com/2013/04/the-three-greatest-paragraphs-ever-written-on-encapsulation.html) provides objects with the ability to hide their internal characteristics and behavior. Each object provides a number of methods, which can be accessed by other objects and change its internal data. In Java, there are three access modifiers: public, private and protected. Each modifier imposes different access rights to other classes, either in the same or in external packages. Some of the advantages of using encapsulation are listed below:

* The internal state of every object is protected by hiding its attributes.
* It increases usability and maintenance of code, because the behavior of an object can be independently changed or extended.
* It improves modularity by preventing objects to interact with each other, in an undesired way.

You can refer to our tutorial[here](http://examples.javacodegeeks.com/java-basics/encapsulation-in-java/) for more details and examples on encapsulation.

A sample class Student which has attributes Id and Name is shown as an example for encapsulation.

*Student Class*

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | public class Student{   private int id;   private String name;     public void setId(int id)   {     this.id = id;   }     public void setName(String name)   {     this.name = name;   }     public int getId()   {     return this.id;   }     public String getName()   {     return this.name;   }    public static void main(String args[])  {    Student student=new Student();    student.setId(1034);    student.setName("David Smith");      System.out.println("Student id "+ student.getId());    System.out.println("Student name "+ student.getName());    }    } |

7. What are the differences between Abstraction and Encapsulation?

Abstraction and encapsulation are complementary concepts. On the one hand, abstraction focuses on the behavior of an object. On the other hand, encapsulation focuses on the implementation of an object’s behavior. Encapsulation is usually achieved by hiding information about the internal state of an object and thus, can be seen as a strategy used in order to provide abstraction.

8. What is Polymorphism?

[Polymorphism](https://www.javacodegeeks.com/2013/04/polymorphism-and-inheritance-are-independent-of-each-other.html) is the ability of programming languages to present the same interface for differing underlying data types. A polymorphic type is a type whose operations can also be applied to values of some other type.

You can see the example below where Vehicle interface has the method increaseVelocity. Truck, Train and Aeroplane implement the Vehicle Interface and the method increases the velocity to the appropriate velocity related to the vehicle type.

Polymorphism

9. What are the types of Polymorphism?

There are two types of Polymorphism in Java:

* Compile-time polymorphism (Static binding) – Method overloading
* Runtime polymorphism (Dynamic binding) – Method overriding

We can perform polymorphism by Method Overloading and Method Overriding.

|  |  |
| --- | --- |
| **Compile Time** | **Runtime** |
| Methods of a class have the same name. Each method has a different number of parameters . It can have parameters with different types and order. | the subclass has method with the name as of a superclass method. It has the number of paramers, type of parameters and the return type as of a superclass method. |
| Method Overloading is to add to the method behavior. It can be extending to the method’s behavior. | Method Overriding is to modify the method’s behavior . |
| Overloaded methods will not have same signature. | Overridden methods will have exactly the same signature. |
| Inheritance is not need in this case. | Inheritance is reuqired. |

Sample code for overloading method subtract of Calculator Class is shown below:

*Calculator Class*

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17 | public class Calculator {    public int subtract(int a, int b)  {     return a-b;  }  public double subtract( double a, double b)  {   return a-b;  }    public static void main(String args[])  {    Calculator calculator = new Calculator();    System.out.println("Difference of 150 and 12 is " +calculator.subtract(150,12));    System.out.println("Difference of 15.5 and 15.4 is " +calculator.subtract(15.50,15.40));  }} |

Method overriding is shown below in Shape class. Shape has a method getArea.

*Shape Class*

|  |  |
| --- | --- |
| 1  2  3  4 | public class Shape  {    public void getArea(){System.out.println("Shape Area");}  } |

Rectangle class overrides getArea method and the implementation of the method is specific to Rectangle. Override annotation is used to indicate to the compiler that the method is overridden. Readability of the code is improved using the annotation.

*Rectangle Class*

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22 | public class Rectangle extends Shape{      @Override    public void getArea()    {      System.out.println("Rectangle Area");      }          public static void main(String args[])    {      Shape shape = new Shape();        shape.getArea();        Rectangle rectangle = new Rectangle();        rectangle.getArea();    }  } |

10. What is Inheritance?

[Inheritance](https://www.javacodegeeks.com/2013/08/multiple-inheritance-in-java-and-composition-vs-inheritance.html) provides an object with the ability to acquire the fields and methods of another class, called base class. Inheritance provides reusability of code and can be used to add additional features to an existing class, without modifying it.

Sample class Mammal is shown below which has a constructor.

*Mammal Class*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | public class Mammal{     public Mammal()   {     System.out.println("Mammal created");   }    } |

Man class extends Mammal which has a default constructor. The sample code is shown below.

*Man class*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | public class Man extends Mammal{     public Man()   {     System.out.println("Man is created");   }  } |

Inheritance is tested by creating an instance of Man using default constructor. The sample code is shown to demonstrate the inheritance.

*TestInheritance Class*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | public class TestInheritance{    public static void main(String args[])  {     Man man = new Man();   }  } |

11. What is composition?

[Composition](https://www.javacodegeeks.com/2018/09/composition-in-java-jep-draft.html) is exactly like Aggregation except that the lifetime of the ‘part’ is controlled by the ‘whole’. This control may be direct or transitive. That is, the ‘whole’ may take direct responsibility for creating or destroying the ‘part’, or it may accept an already created part, and later pass it on to some other whole that assumes responsibility for it.

Sample class Car is shown below to demonstrate Composition of tires, doors, windows and steering.

*Car class*

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | public class Car  {    private Tire[] tires;      private Door[] doors;      private Steering steering;      private Window[] windows;  }    class Tire  {    }    class Door  {    }    class Steering  {    }    class Window  {    } |

12. What is an association?

Association represents the ability of one instance to send a message to another instance. This is typically implemented with a pointer or reference instance variable, although it might also be implemented as a method argument or the creation of a local variable.

13. What is aggregation?

Aggregation is the typical whole/part relationship. This is exactly the same as an association with the exception that instances cannot have cyclic aggregation relationships.

Sample class Person is shown below to demonstrate Aggregation relationship with Address.

*Person class*

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19 | public class Person  {    private Address address;    }    class Address  {    private String city;      private String state;      private String country;      private String line1;      private String line2;    } |

B.General Questions about Java

14. What is JVM?

A Java virtual machine (JVM) is a process[virtual machine](https://www.javacodegeeks.com/2013/12/part-1-of-3-synopsis-of-articles-videos-on-performance-tuning-jvm-gc-in-java-mechanical-sympathy-et-al.html) that can execute Java[bytecode](https://www.javacodegeeks.com/2013/12/mastering-java-bytecode.html). Each Java source file is compiled into a bytecode file, which is executed by the JVM.

15. Why is Java called the Platform Independent Programming Language?

Java was designed to allow application programs to be built that could be run on any platform, without having to be rewritten or recompiled by the programmer for each separate platform. A Java virtual machine makes this possible because it is aware of the specific instruction lengths and other particularities of the underlying hardware platform.

16. What is the Difference between JDK and JRE?

The Java Runtime Environment (JRE) is basically the Java Virtual Machine (JVM) where your Java programs are being executed. It also includes browser plugins for applet execution. The Java Development Kit (JDK) is the full-featured Software Development Kit for Java, including the JRE, the compilers and tools (like[JavaDoc](https://docs.oracle.com/javase/7/docs/technotes/tools/windows/javadoc.html), and[Java Debugger](https://docs.oracle.com/javase/7/docs/technotes/tools/windows/jdb.html)), in order for a user to develop, compile and execute Java applications.

|  |  |
| --- | --- |
| **JDK** | **JRE** |
| JDK stands for the term : Java Development Kit. | JRE stands for the term: Java Runtime Environment. |
| JDK is the tool for compilng, documenting and packaging Java software. | JRE is a runtime environment. JavaByte code gets executed in the environment. |
| JDK has JRE and development tools. | JRE is a JVM implementation |

17. What does the static keyword mean?

The static keyword denotes that a member variable or method can be accessed, without requiring an instantiation of the class to which it belongs.

Sample static method which is shown below:

*Static method*

|  |  |
| --- | --- |
| 1  2  3  4  5 | static void printGreeting()      {          } |

18. Can you override private or static method in Java?

A user cannot override[static methods in Java](https://www.javacodegeeks.com/2012/05/java-static-methods-can-be-code-smell.html), because method overriding is based upon dynamic binding at runtime and static methods are statically bound at compile time. A static method is not associated with any instance of a class so the concept is not applicable.

19. Can you access the non-static variable in static context?

A static variable in Java belongs to its class and its value remains the same for all its instances. A static variable is initialized when the class is loaded by the JVM. If your code tries to access a non-static variable, without any instance, the compiler will complain, because those variables are not created yet and they are not associated with any instance.

20. What are the Data Types supported by Java?

The eight primitive data types supported by the Java programming language are:

* byte
* short
* int
* long
* float
* double
* boolean
* char

21. What is Autoboxing and Unboxing?

Autoboxing is the[automatic conversion made by the Java compiler](https://www.javacodegeeks.com/2013/07/java-generics-tutorial-example-class-interface-methods-wildcards-and-much-more.html) between the primitive types and their corresponding object wrapper classes. For example, the compiler converts an int to an[Integer](https://docs.oracle.com/javase/7/docs/api/java/lang/Integer.html?is-external=true), a double to a[Double](https://docs.oracle.com/javase/7/docs/api/java/lang/Double.html), and so on. If the conversion goes the other way, this operation is called unboxing.

22. What is Function Overriding and Overloading in Java?

Method overloading in Java occurs when two or more methods in the same class have the exact same name, but different parameters. On the other hand, method overriding is defined as the case when a child class redefines the same method as a parent class. Overridden methods must have the same name, argument list, and return type. The overriding method may not limit the access of the method it overrides.

23. What is a Constructor?

A constructor gets invoked when a new object is created. Every class[has a constructor](https://www.javacodegeeks.com/2014/01/which-is-better-option-cloning-or-copy-constructors.html). In case the programmer does not provide a constructor for a class, the Java compiler (Javac) creates a default constructor for that class.

A default constructor in java is shown in the example below:

*Default Constructor*

|  |  |
| --- | --- |
| 1  2  3  4 | public Man()  {    System.out.println("Man is created");  } |

Constructor which takes a parameter is shown in the sample below:

*Constructor*

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | private String name;        public Employee(String name)      {        this.name = name;      } |

24. What is Constructor Overloading?

The constructor overloading is similar to method overloading in Java. Different constructors can be created for a single class. Each constructor must have its own unique parameter list.

25. What is Copy-Constructor?

Finally, Java does support copy constructors like C++, but the difference lies in the fact that Java does not create a default copy constructor if you do not write your own.

Copy constructor for Employee class is shown below:

*Copy Constructor*

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15 | public class Employee extends Person  {      private String name;        public Employee(String name)      {        this.name = name;      }        public Employee(Employee emp)      {        this.name = emp.name;      }    } |

26. Does Java support multiple inheritance?

No, Java does not support multiple inheritance. Each class is able to extend only on one class but is able to implement more than one interfaces.

Multiple Inheritance

27. What is the difference between an Interface and an Abstract class?

Java provides and supports the creation of both the[abstract classes](http://examples.javacodegeeks.com/java-basics/java-abstract-class-example/) and interfaces. Both implementations share some common characteristics, but they differ in the following features:

* All methods in an interface are implicitly abstract. On the other hand, an abstract class may contain both abstract and non-abstract methods.
* A class may implement a number of Interfaces but can extend only one abstract class.
* In order for a class to implement an interface, it must implement all its declared methods. However, a class may not implement all declared methods of an abstract class. Though, in this case, the sub-class must also be declared as abstract.
* Abstract classes can implement interfaces without even providing the implementation of interface methods.
* Variables declared in a Java interface is by default final. An abstract class may contain non-final variables.
* Members of a Java interface are public by default. A member of an abstract class can either be private, protected or public.
* An interface is absolutely abstract and cannot be instantiated. An abstract class also cannot be instantiated but can be invoked if it contains the main method.

Also, check out the[Abstract class and Interface differences for JDK 8](https://www.javacodegeeks.com/2014/04/abstract-class-versus-interface-in-the-jdk-8-era.html).

|  |  |
| --- | --- |
| **Interface** | **Abstract Class** |
| An interface has the method signatures. It does not have any implementation. | Abstract class has the abstract methods and details to be overridden. |
| A Class can implement multiple interfaces | In this case, a class can extend just one abstract class |
| Interface has all abstract methods. | Non abstract methods can be there in an abstract class. |
| Instance properties cannot be there in an interface. | Instance properties can be there in an abstract class. |
| An Interface is publicly visible or not visible. | An abstract class can be public, private and protected visibile. |
| Any change in the interface will impact the classes implementing the interface. | Adding a method to an abstract class and implementing it does not require change in the code for derived classes. |
| An Interface cannot have constructors | An abstract class can have constructors |
| Interfaces are slow in terms of performance | Abstract classes are fast in execution of the methods in the derived classes. |

28. What are pass by reference and pass by value?

When an object is passed by value, this means that a copy of the object is passed. Thus, even if changes are made to that object, it does not affect the original value. When an object is passed by reference, this means that the actual object is not passed, rather a reference of the object is passed. Thus, any changes made by the external method, are also reflected in all places.

Sample code is presented below which shows pass by value.

*Pass by Value*

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14 | public class ComputingEngine  {      public static void main(String[] args)      {          int x = 15;          ComputingEngine engine = new ComputingEngine();          engine.modify(x);          System.out.println("The value of x after passing by value "+x);      }      public  void modify(int x)      {          x = 12;      }  } |

Below example shows pass by reference in the code.

*Pass by Reference*

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | public class ComputingEngine  {      public static void main(String[] args)      {            ComputingEngine engine = new ComputingEngine();              Computation computation = new Computation(65);          engine.changeComputedValue(computation);            System.out.println("The value of x after passing by reference "+ computation.x);        }            public void changeComputedValue(Computation computation)      {          computation = new Computation();          computation.x = 40;      }  }      class Computation  {      int x;      Computation(int i) { x = i; }      Computation()      { x = 1; }  } |

29. What is the purpose of a Volatile Variable?

[Volatile](https://www.javacodegeeks.com/2018/03/volatile-java-works-example-volatile-keyword-java.html) variable values can be modified by different threads. They will never have the chance to block and hold a lock. Synchronization will happen whenever the variables are accessed. Using volatile may be faster than a lock, but it will not work in some situations. The range of situations in which volatile is effective was expanded in Java 5; in particular, double-checked locking now works correctly.

Sample code for volatile variable is shown below:

*Volatile Variable*

|  |  |
| --- | --- |
| 1  2  3  4  5 | public class DistributedObject {        public volatile int count = 0;    } |

30. What is the purpose of a Transient variable?

A transient variable would not be serialized even if the class to which it belongs is serialized.

Sample class which has transient variable is shown below:

*Transitent variable*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | public class Paper implements Serializable  {      private int id;      private String title;      private String author;      private transient int version = 1;    } |

31. What is Local Variable and Instance Variable?

|  |  |
| --- | --- |
| **Local variable** | **Instance variable** |
| Local variable is declared inside a method or constructor. It can be declared within a block | Instance variable is declared inside a class. |
| Local variable need to be initialized before use. The code will not compile. | Instance variable initialization is not necessary. If not initialized, default value is used. |

32. What are the different access modifiers available in Java?

There are four types of Access modifiers:

* Public – accessible from everywhere in the application
* Protected – accessible within the package and the subclasses in any package
* Package Private (Default) – accessible strictly within the package
* Private – accessible only within the same class where it is declared

33. Difference between static binding and dynamic binding

|  |  |
| --- | --- |
| **Static Binding** | **Dynamic Binding** |
| Definition of a procedure is related to static binding | An example for dynamic binding is activation of a procedure |
| Declaration of a name for a variable is done to bind statically the variable. | Binding of a name can be dynamic bound. |
| The Scope of the declaration is statically bound. | Lifetime of a binding is dynamically bound. |

Sample code for static binding is shown below:

*Static Binding*

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15 | public class Shape  {    public void getArea()     {       System.out.println("Shape Area");      }      public static void main(String args[])    {      Shape shape = new Shape();        shape.getArea();    }  } |

Sample code for Dynamic binding is shown below:

*Dynamic Binding*

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19 | public class Rectangle extends Shape{        public void getArea()    {      System.out.println("Rectangle Area");      }          public static void main(String args[])    {        Shape shape = new Rectangle();        shape.getArea();    }  } |

34. What are wrapper classes?

A wrapper class converts java primitives into objects. So a primitive wrapper class is a wrapper class that encapsulates, hides or wraps data types from the eight primitive data types so that these can be used to create instantiated objects with methods in another class or in other classes. The primitive wrapper classes are found in the Java API.

35. What is singleton class and how can we make a class singleton?

In a singleton class we:

* ensure that only one instance of the singleton class ever exists
* provide global access to that instance

To create a singleton class we:

* declare all constructors of the class as private
* provide a static method that returns a reference to the instance

Sample code below shows Double checked Singleton class implementation.

*Singleton Class*

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14 | public class DoubleCheckedSingleton {      private static volatile DoubleCheckedSingleton instance;      public static DoubleCheckedSingleton getInstance() {          if (instance == null) {              synchronized (DoubleCheckedSingleton .class) {                  if (instance == null) {                      instance = new DoubleCheckedSingleton();                  }              }          }          return instance;      }    } |

C.Java Threads

36. What is the difference between processes and threads?

A process is an execution of a program, while a Thread is a single execution sequence within a process. A process can contain multiple threads. A Thread is sometimes called a lightweight process.

|  |  |
| --- | --- |
| **Processes** | **Threads** |
| Process is related to execution of a program . | Process consists of multiple threds. |
| Processes communicate with each other using inter-process communication . | Threads of a process can communicate with each other. |
| Processes have control over the child processes. | Threads of a process can have control over other threads. |
| Any modification in the parent process does not alter child processes | Any modification in the main thread may impact the behavior of the other threads of the process. |
| Processes get executed in separate memory spaces. | Threads are executed in shared memory spaces. |
| Operating sytem controls the rocess . | Developer of the software has control over the usage of the threads. |
| Processes are independent of each other. | Threads are dependent on each other. |

37. Explain different ways of creating a thread. Which one would you prefer and why?

There are three ways that can be used in order for a Thread to be created:

* A class may extend the[Thread](https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) class.
* A class may implement the[Runnable](https://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html) interface.
* An application can use the[Executor](https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/Executor.html) framework, in order to create a thread pool.

The Runnable interface is preferred, as it does not require an object to inherit the Thread class. In case your application design requires multiple inheritance, only interfaces can help you. Also, the thread pool is very efficient and can be implemented and used very easily.

38. Explain the available thread states in a high-level.

During its execution, a thread can reside in one of the following[states](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html):

* [NEW](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html#NEW): The thread becomes ready to run, but does not necessarily start running immediately.
* [RUNNABLE](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html#RUNNABLE): The Java Virtual Machine (JVM) is actively executing the thread’s code.
* [BLOCKED](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html#BLOCKED): The thread is in a blocked state while waiting for a monitor lock.
* [WAITING](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html#WAITING): The thread waits for another thread to perform a particular action.
* [TIMED\_WAITING](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html#TIMED_WAITING): The thread waits for another thread to perform a particular action up to a specified waiting time.
* [TERMINATED](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html#TERMINATED): The thread has finished its execution.

39. What is the difference between a method and block that are synchronized?

In Java programming, each object has a lock. A thread can acquire the lock for an object by using the synchronized keyword. The synchronized keyword can be applied in a method level (coarse-grained lock) or block level of code (fine-grained lock).

40. How does thread synchronization occurs inside a monitor?

The JVM uses locks in conjunction with monitors. A monitor is basically a guardian that watches over a sequence of synchronized code and ensuring that only one thread at a time executes a synchronized piece of code. Each monitor is associated with an object reference. The thread is not allowed to execute the code until it obtains the lock.

41. What is a deadlock?

A condition that occurs when[two processes are waiting for each other to complete](https://www.javacodegeeks.com/2013/01/java-deadlock-example-how-to-analyze-deadlock-situation.html), before proceeding. The result is that both processes wait endlessly.

42. How do you ensure that N threads can access N resources without deadlock?

A very simple way to avoid deadlock while using N threads is to impose an ordering on the locks and force each thread to follow that ordering. Thus, if all threads lock and unlock the mutexes in the same order, no deadlocks can arise.

43. What are the differences between wait and sleep method in Java?

|  |  |  |
| --- | --- | --- |
|  | **Wait** | **Sleep** |
| **Call on** | current thread synchronizes on the lock object when there is a call on the object. | Call on a Thread happens on the currently executing thread. |
| **Synchronized** | Synchronized is used to access the same Object from multiple threads.. | Synchronized is used to sleep over the Sleeping thread from multiple threads. |
| **Hold lock** | release the lock for other objects to have the chance to execute | keep lock for at least t times if timeout specified or somebody interrupt. |
| **Wake up condition** | until call notify(), notifyAll() from object | until at least time expire or call interrupt(). |
| **Usage** | for time-synchronization | for multi-thread-synchronization |

D.Java Collections

44. What are the basic interfaces of Java Collections Framework?

[Java Collections Framework](https://docs.oracle.com/javase/7/docs/technotes/guides/collections/overview.html) provides a well-designed set of interfaces and classes that support operations on a collections of objects. The most basic interfaces that reside in the Java Collections Framework are:

* [Collection](https://docs.oracle.com/javase/7/docs/api/java/util/Collection.html), which represents a group of objects known as its elements.
* [Set](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html), which is a collection that cannot contain duplicate elements.
* [List](https://docs.oracle.com/javase/7/docs/api/java/util/List.html), which is an ordered collection and can contain duplicate elements.
* [Map](https://docs.oracle.com/javase/7/docs/api/java/util/Map.html), which is an object that maps keys to values and cannot contain duplicate keys.

Collection Hierarchy

45. Why Collection does not extend Cloneable and Serializable interfaces?

The [Collection](https://docs.oracle.com/javase/7/docs/api/java/util/Collection.html) interface specifies groups of objects known as elements. Each concrete implementation of a Collection can choose its own way of how to maintain and order its elements. Some collections allow duplicate keys, while some other collections do not. The semantics and the implications of either cloning or serialization come into play when dealing with actual implementations. Thus, the concrete implementations of collections should decide how they can be cloned or serialized.

46. What is an Iterator?

The[Iterator](https://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) interface provides a number of methods that are able to iterate over any[Collection](https://docs.oracle.com/javase/7/docs/api/java/util/Collection.html). Each Java Collection contains the iterator method that returns an Iterator instance. Iterators are[capable of removing elements from the underlying collection](https://www.javacodegeeks.com/2011/05/avoid-concurrentmodificationexception.html) during the iteration.

47. What differences exist between Iterator and ListIterator?

The differences of these elements are listed below:

* An[Iterator](https://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) can be used to traverse the[Set](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html) and[List](https://docs.oracle.com/javase/7/docs/api/java/util/List.html) collections, while the[ListIterator](https://docs.oracle.com/javase/7/docs/api/java/util/ListIterator.html) can be used to iterate only over Lists.
* The Iterator can traverse a collection only in the forward direction, while the ListIterator can traverse a List in both directions.
* The ListIterator implements the Iterator interface and contains extra functionality, such as adding an element, replacing an element, getting the index position for previous and next elements, etc.

48. What is the difference between fail-fast and fail-safe?

The[Iterator’s](https://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) fail-safe property works with the clone of the underlying collection and thus, it is not affected by any modification in the collection. All the collection classes in java.util package are fail-fast, while the collection classes in java.util.concurrent are fail-safe. Fail-fast iterators throw a[ConcurrentModificationException](http://examples.javacodegeeks.com/java-basics/exceptions/java-util-concurrentmodificationexception-how-to-handle-concurrent-modification-exception/), while fail-safe iterator never throws such an exception.

49. How HashMap works in Java?

A[HashMap in Java stores key-value pairs](https://www.javacodegeeks.com/2014/03/how-hashmap-works-in-java.html). The[HashMap](https://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) requires a hash function and uses[hashCode](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#hashCode%28%29) and equals methods, in order to put and retrieve elements to and from the collection respectively. When the put method is invoked, the HashMap calculates the hash value of the key and stores the pair in the appropriate index inside the collection. If the key exists, its value is updated with the new value. Some important characteristics of a HashMap are its capacity, its load factor and the threshold resizing.

50. What is the importance of hashCode() and equals() methods?

In Java, a[HashMap](https://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) uses the[hashCode](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#hashCode%28%29) and[equals](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#equals%28java.lang.Object%29) methods to determine the index of the key-value pair and to detect duplicates. More specifically, the hashCode method is used in order to determine where the specified key will be stored. Since different keys may produce the same hash value, the equals method is used, in order to determine whether the specified key actually exists in the collection or not. Therefore, the implementation of both methods is crucial to the accuracy and efficiency of the HashMap.

51. What is the difference between HashMap and Hashtable?

Both the[HashMap](https://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) and[Hashtable](https://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html) classes implement the Map interface and thus, have very similar characteristics. However, they differ in the following features:

* A HashMap allows the existence of null keys and values, while a Hashtable does not allow neither null keys, nor null values.
* A Hashtable is synchronized, while a HashMap is not. Thus, HashMap is preferred in single-threaded environments, while a Hashtable is suitable for multi-threaded environments.
* A HashMap provides its set of keys and a Java application can iterate over them. Thus, a HashMap is fail-fast. On the other hand, a Hashtable provides an[Enumeration](https://docs.oracle.com/javase/7/docs/api/java/util/Enumeration.html) of its keys.
* The Hashtable class is considered to be a legacy class.

52. What is the difference between Array and ArrayList? When will you use Array over ArrayList?

The[Array](https://docs.oracle.com/javase/7/docs/api/java/lang/reflect/Array.html) and[ArrayList](https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html) classes differ on the following features:

* Arrays can contain primitive or objects, while an ArrayList can contain only objects.
* Arrays havefixed size, while an ArrayList is dynamic.
* An ArrayList provides more methods and features, such as[addAll](https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html#addAll(java.util.Collection)),[removeAll](https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html#removeAll(java.util.Collection)),[iterator](https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html#iterator()), etc.
* For a list of primitive data types, the collections use autoboxing to reduce the coding effort. However, this approach makes them slower when working on fixed size primitive data types.

|  |  |
| --- | --- |
| **Array** | **ArrayList** |
| Array should not have values of different data types | Array List can have values of different data types. |
| Size of the arrray is defined at the time of declaration | Size of the ArrayList can be dynamically changed |
| You haveto specify the index in order to add data in an array | You do not need to specify the index in an ArrayList |
| Arrays are not type parameterized | Arraylists can be type paramaterized. |
| Arrays can have primitive data types as well as objects | Arraylists can have only objects, no primitive data types are allowed |

53. What is difference between ArrayList and LinkedList?

Both the[ArrayList](https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html) and[LinkedList](https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html) classes implement the List interface, but they differ on the following features:

* An ArrayList is an index based data structure backed by an Array. It provides random access to its elements with a performance equal to O(1). On the other hand, a LinkedList stores its data as list of elements and every element is linked to its previous and next element. In this case, the search operation for an element has execution time equal to O(n).
* The Insertion, addition and removal operations of an element are faster in a LinkedList compared to an ArrayList, because there is no need of resizing an array or updating the index when an element is added in some arbitrary position inside the collection.
* A LinkedList consumes more memory than an ArrayList, because every node in a LinkedList stores two references, one for its previous element and one for its next element.

Check also our article[ArrayList vs. LinkedList](https://www.javacodegeeks.com/2013/12/arraylist-vs-linkedlist.html).

54. What is the difference between Comparable and Comparator?

* Java provides the[Comparable](https://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html) interface, which contains only one method, called[compareTo](https://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html#compareTo(T)). This method compares two objects, in order to impose an order between them. Specifically, it returns a negative integer, zero, or a positive integer to indicate that the input object is less than, equal or greater than the existing object.
* Java provides the[Comparator](https://docs.oracle.com/javase/7/docs/api/java/util/Comparator.html) interface, which contains two methods, called[compare](https://docs.oracle.com/javase/7/docs/api/java/util/Comparator.html#compare(T,%20T)) and[equals](https://docs.oracle.com/javase/7/docs/api/java/util/Comparator.html#equals(java.lang.Object)). The first method compares its two input arguments and imposes an order between them. It returns a negative integer, zero, or a positive integer to indicate that the first argument is less than, equal to, or greater than the second. The second method requires an object as a parameter and aims to decide whether the input object is equal to the comparator. The method returns true, only if the specified object is also a comparator and it imposes the same ordering as the comparator.

55. What is Java Priority Queue?

The[PriorityQueue](https://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html) is an unbounded queue, based on a priority heap and its elements are ordered in their natural order. At the time of its creation, we can provide a Comparator that is responsible for ordering the elements of the PriorityQueue. A PriorityQueue does not allow[null values](http://examples.javacodegeeks.com/java-basics/exceptions/java-lang-nullpointerexception-how-to-handle-null-pointer-exception/), those objects that do not provide natural ordering or those objects that do not have any comparator associated with them. Finally, the Java PriorityQueue is not thread-safe and it requires O(log(n)) time for its enqueuing and dequeuing operations.

56. What do you know about the big-O notation and can you give some examples with respect to different data structures?

The[Big-O notation](https://www.javacodegeeks.com/2011/04/simple-big-o-notation-post.html) simply describes how well an algorithm scales or performs in the worst case scenario as the number of elements in a data structure increases. The Big-O notation can also be used to describe other behavior such as memory consumption. Since the collection classes are actually data structures, we usually use the Big-O notation to choose the best implementation to use, based on time, memory and performance. Big-O notation can give a good indication about performance for large amounts of data.

57. What is the trade-off between using an unordered array versus an ordered array?

The major advantage of an ordered array is that the search times have time complexity of O(log n), compared to that of an unordered array, which is O (n). The disadvantage of an ordered array is that the insertion operation has a time complexity of O(n), because the elements with higher values must be moved to make room for the new element. Instead, the insertion operation for an unordered array takes constant time of O(1).

58. What are some of the best practices related to the Java Collection framework?

* Choosing the right type of the collection to use, based on the application’s needs, is very crucial for its performance. For example if the size of the elements is fixed and know a priori, we shall use an[Array](https://docs.oracle.com/javase/7/docs/api/java/lang/reflect/Array.html), instead of an[ArrayList](https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html).
* Some collection classes allow us to specify their initial capacity. Thus, if we have an estimation on the number of elements that will be stored, we can use it to avoid rehashing or resizing.
* Always use Generics for type-safety, readability, and robustness. Also, by using Generics you avoid the[ClassCastException](https://docs.oracle.com/javase/7/docs/api/java/lang/ClassCastException.html) during runtime.
* Use immutable classes provided by the Java Development Kit (JDK) as a key in a Map, in order to avoid the implementation of the[hashCode](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#hashCode%28%29) and equals methods for our custom class.
* Program in terms of interface not implementation.
* Return zero-length collections or arrays as opposed to returning a null in case the underlying collection is actually empty.

59. What is the difference between Enumeration and Iterator interfaces?

[Enumeration](https://docs.oracle.com/javase/7/docs/api/java/util/Enumeration.html) is twice as fast as compared to an Iterator and uses very less memory. However, the[Iterator](https://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) is much safer compared to Enumeration, because other threads are not able to modify the collection object that is currently traversed by the iterator. Also, Iterators allow the caller to remove elements from the underlying collection, something which is not possible with Enumerations.

60. What is the difference between HashSet and TreeSet?

The[HashSet](https://docs.oracle.com/javase/7/docs/api/java/util/HashSet.html) is Implemented using a hash table and thus, its elements are not ordered. The add, remove, and contains methods of a HashSet have constant time complexity O(1). On the other hand, a[TreeSet](https://docs.oracle.com/javase/7/docs/api/java/util/TreeSet.html) is implemented using a tree structure. The elements in a TreeSet are sorted, and thus, the add, remove, and contains methods have time complexity of O(logn).

E.Garbage Collectors

61. What is the purpose of garbage collection in Java, and when is it used?

The purpose of garbage collection is to identify and discard those objects that are no longer needed by the application, in order for the resources to be reclaimed and reused.

62. What does System.gc() and Runtime.gc() methods do?

These methods can be used as a hint to the JVM, in order to start a garbage collection. However, this it is up to the Java Virtual Machine (JVM) to start the garbage collection immediately or later in time.

Sample class ReferenceObject is shown below to demonstrate the usage of System.gc and Runtime.gc methods.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18 | public class ReferenceObject  {   public void finalize()   {      System.out.println("object is garbage collected");     }     public static void main(String args[]){    ReferenceObject refObj1=new ReferenceObject();    ReferenceObject refObj2=new ReferenceObject();    refObj1=null;    refObj2=null;    System.gc();      Runtime.gc();   }  } |

63. When is the finalize() called? What is the purpose of finalization?

The finalize method is called by the garbage collector, just before releasing the object’s memory. It is normally advised to release resources held by the object inside the finalize method.

Finalize method in ReferenceObject class is shown below as an example.

*Finalize Method*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | public class ReferenceObject  {   public void finalize()   {      System.out.println("object is garbage collected");     }   } |

64. If an object reference is set to null, will the Garbage Collector immediately free the memory held by that object?

No, the object will be available for garbage collection in the next cycle of the garbage collector.

65. What is structure of Java Heap?

The[JVM has a heap](https://www.javacodegeeks.com/2012/07/5-tips-for-proper-java-heap-size.html) that is the runtime data area from which memory for all class instances and arrays is allocated. It is created at the JVM start-up. Heap memory for objects is reclaimed by an automatic memory management system which is known as a garbage collector. Heap memory consists of live and dead objects. Live objects are accessible by the application and will not be a subject of garbage collection. Dead objects are those which will never be accessible by the application, but have not been collected by the garbage collector yet. Such objects occupy the heap memory space until they are eventually collected by the garbage collector.

66. What is the difference between Serial and Throughput Garbage collector?

The throughput garbage collector uses a parallel version of the young generation collector and is meant to be used with applications that have medium to large data sets. On the other hand, the serial collector is usually adequate for most small applications (those requiring heaps of up to approximately 100MB on modern processors).

67. When does an Object becomes eligible for Garbage collection in Java?

A Java object is subject to garbage collection when it becomes unreachable to the program in which it is currently used.

68. Does Garbage collection occur in permanent generation space in JVM?

Garbage Collection does occur in PermGen space and if PermGen space is full or cross a threshold, it can trigger a full garbage collection. If you look carefully at the output of the garbage collector, you will find that PermGen space is also garbage collected. This is the reason why correct sizing of PermGen space is important to avoid frequent full garbage collections. Also check our article[Java 8: PermGen to Metaspace](https://www.javacodegeeks.com/2013/02/java-8-from-permgen-to-metaspace.html).

F.Exception Handling

69. What are the differences between Checked Exception and Unchecked Exception?

|  |  |
| --- | --- |
| **Checked Exception** | **Unchecked Exception** |
| known as compile time exceptions | known as Runtime exceptions |
| propagated using throws keyword | automatically propagated |
| can create custom exception by extending java.lang.Exception class | can create custom exception by extending Runtime exception |

70. What is the difference between Exception and Error in java?

[Exception](https://docs.oracle.com/javase/7/docs/api/java/lang/Exception.html) and[Error](https://docs.oracle.com/javase/7/docs/api/java/lang/Error.html) classes are both subclasses of the[Throwable](https://docs.oracle.com/javase/7/docs/api/java/lang/Throwable.html) class. The Exception class is used for exceptional conditions that a user’s program should catch. The Error class defines exceptions that are not expected to be caught by the user program.

71. What is the difference between throw and throws?

The throw keyword is used to explicitly raise a exception within the program. On the contrary, the throws clause is used to indicate those exceptions that are not handled by a method. Each method must explicitly specify which exceptions does not handle, so the callers of that method can guard against possible exceptions. Finally, multiple exceptions are separated by a comma.

|  |  |
| --- | --- |
| **Throw** | **Throws** |
| Throw is used for throwing an exception explicitly. | To declar an exception,throws is used. |
| Using throw only, Checked exceptions can not be propagated. | Using throws, Checked exception can be propagated. |
| Throw is always used with an instance. | Throws is always used with a class. |
| Throw is used inside the method. | Throws is used always with the method signature. |
| You should not throw multiple exception | You can declare multiple exceptions. |

72. What is the importance of finally block in exception handling?

A finally block will always be executed, whether or not an exception is actually thrown. Even in the case where the catch statement is missing and an exception is thrown, the finally block will still be executed. Last thing to mention is that the finally block is used to release resources like I/O buffers, database connections, etc.

Sample code below shows the finally block when exception is thrown.

*Finally Block*

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17 | public class DivideByZeroException  {       public static void main(String []args){          try{              int a = 1;              System.out.println(a/0);          }          catch(Exception exception)          {            System.out.println("exception is thrown");          }          finally          {              System.out.println("after the exception is handled");          }       }  } |

73. What will happen to the Exception object after exception handling?

The[Exception](https://docs.oracle.com/javase/7/docs/api/java/lang/Exception.html) object will be garbage collected in the next garbage collection.

74. What purpose does the keywords final, finally, and finalize fulfill?

* **Final** keyword is used to apply restrictions on class(immutable), method(cannot override) and variable(constant).
* **Finally** is a block that always executes when the try block exits even if an unexpected exception occurs.
* **Finalize** is a method called to clean or release the resources by the Garbage Collector before destroying the object.

G.Java Applets

75. What is an Applet?

A java applet is program that can be included in a HTML page and be executed in a java enabled client browser. Applets are used for creating dynamic and interactive web applications.

76. Explain the life cycle of an Applet.

An applet may undergo the following states:

* Init : An applet is initialized each time is loaded.
* Start : Begin the execution of an applet.
* Stop : Stop the execution of an applet.
* Destroy : Perform a final cleanup, before unloading the applet.

Applet Lifecycle

77. What happens when an applet is loaded?

First of all, an instance of the applet’s controlling class is created. Then, the applet initializes itself and finally, it starts running.

78. What is the difference between an Applet and a Java Application?

Applets are executed within a java enabled browser, but a Java application is a standalone Java program that can be executed outside of a browser. However, they both require the existence of a Java Virtual Machine (JVM). Furthermore, a Java application requires a main method with a specific signature, in order to start its execution. Java applets do not need such a method to start their execution. Finally, Java applets typically use a restrictive security policy, while Java applications usually use more relaxed security policies.

79. What are the restrictions imposed on Java applets?

Mostly due to security reasons, the following restrictions are imposed on Java applets:

* An applet cannot load libraries or define native methods.
* An applet cannot ordinarily read or write files on the execution host.
* An applet cannot read certain system properties.
* An applet cannot make network connections except to the host that it came from.
* An applet cannot start any program on the host that is executing it.

80. What are untrusted applets?

Untrusted applets are those Java applets that cannot access or execute local system files. By default, all downloaded applets are considered as untrusted.

81. What is the difference between applets loaded over the internet and applets loaded via the file system?

Regarding the case where an applet is loaded over the internet, the applet is loaded by the applet classloader and is subject to the restrictions enforced by the applet security manager. Regarding the case where an applet is loaded from the client’s local disk, the applet is loaded by the file system loader. Applets loaded via the file system are allowed to read files, write files and to load libraries on the client. Also, applets loaded via the file system are allowed to execute processes and finally, applets loaded via the file system are not passed through the byte code verifier.

82. What is the applet class loader, and what does it provide?

When an applet is loaded over the internet, the applet is loaded by the applet classloader. The class loader enforces the Java namespace hierarchy. Also, the class loader guarantees that a unique namespace exists for classes that come from the local file system, and that a unique namespace exists for each network source. When a browser loads an applet over the net, that applet’s classes are placed in a private namespace associated with the applet’s origin. Then, those classes loaded by the class loader are passed through the verifier. The verifier checks that the class file conforms to the Java language specification. Among other things, the verifier ensures that there are no stack overflows or underflows and that the parameters to all bytecode instructions are correct.

83. What is the applet security manager, and what does it provide?

The applet security manager is a mechanism to impose restrictions on Java applets. A browser may only have one security manager. The security manager is established at startup, and it cannot thereafter be replaced, overloaded, overridden, or extended.

[H.Swing](http://examples.javacodegeeks.com/desktop-java/swing/)

84. What is the difference between a Choice and a List?

A Choice is displayed in a compact form that must be pulled down, in order for a user to be able to see the list of all available choices. Only one item may be selected from a Choice. A[List](http://examples.javacodegeeks.com/desktop-java/swing/jlist/create-jlist-example/) may be displayed in such a way that several List items are visible. A List supports the selection of one or more List items.

85. What is a layout manager?

A layout manager is the used to organize the components in a container.

86. What is the difference between a Scrollbar and a JScrollPane?

A[Scrollbar](https://docs.oracle.com/javase/7/docs/api/java/awt/Scrollbar.html) is a[Component](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html), but not a[Container](https://docs.oracle.com/javase/7/docs/api/java/awt/Container.html). A[ScrollPane](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html) is a Container. A ScrollPane handles its own events and performs its own scrolling.

87. Which Swing methods are thread-safe?

There are only three thread-safe methods: repaint, revalidate, and invalidate.

88. Name three Component subclasses that support painting.

The[Canvas](https://docs.oracle.com/javase/7/docs/api/java/awt/Canvas.html),[Frame](https://docs.oracle.com/javase/7/docs/api/java/awt/Frame.html),[Panel](https://docs.oracle.com/javase/7/docs/api/java/awt/Panel.html), and Applet classes support painting.

89. What is clipping?

Clipping is defined as the process of confining paint operations to a limited area or shape.

90. What is the difference between a MenuItem and a CheckboxMenuItem?

The[CheckboxMenuItem](https://docs.oracle.com/javase/7/docs/api/java/awt/CheckboxMenuItem.html) class extends the[MenuItem](https://docs.oracle.com/javase/7/docs/api/java/awt/MenuItem.html) class and supports a menu item that may be either checked or unchecked.

91. How are the elements of a BorderLayout organized?

The elements of a[BorderLayout](https://docs.oracle.com/javase/7/docs/api/java/awt/BorderLayout.html) are organized at the borders (North, South, East, and West) and the center of a container.

92. How are the elements of a GridBagLayout organized?

The elements of a[GridBagLayout](https://docs.oracle.com/javase/7/docs/api/java/awt/GridBagLayout.html) are organized according to a grid. The elements are of different sizes and may occupy more than one row or column of the grid. Thus, the rows and columns may have different sizes.

93. What is the difference between a Window and a Frame?

The[Frame](https://docs.oracle.com/javase/7/docs/api/java/awt/Frame.html) class extends the[Window](https://docs.oracle.com/javase/7/docs/api/java/awt/Window.html) class and defines a main application window that can have a menu bar.

94. What is the relationship between clipping and repainting?

When a window is repainted by the AWT painting thread, it sets the clipping regions to the area of the window that requires repainting.

95. What is the relationship between an event-listener interface and an event-adapter class?

An event-listener interface defines the methods that must be implemented by an event handler for a particular event. An event adapter provides a default implementation of an event-listener interface.

96. How can a GUI component handle its own events?

A GUI component can handle its own events, by implementing the corresponding event-listener interface and adding itself as its own event listener.

97. What advantage do Java’s layout managers provide over traditional windowing systems?

Java uses layout managers to lay out components in a consistent manner, across all windowing platforms. Since layout managers are not tied to absolute sizing and positioning, they are able to accomodate platform-specific differences among windowing systems.

98. What is the design pattern that Java uses for all Swing components?

The design pattern used by Java for all Swing components is the Model View Controller (MVC) pattern.

MVC

I.JDBC

99. What is JDBC?

[JDBC](https://www.javacodegeeks.com/jdbc-tutorials) is an abstraction layer that allows users to choose between databases.[JDBC enables developers to write database applications in Java](https://www.javacodegeeks.com/2014/03/java-8-friday-java-8-will-revolutionize-database-access.html), without having to concern themselves with the underlying details of a particular database.

100. What are the JDBC API components?

The java.sql package contains:

**Interfaces**:

* Driver
* Connection
* Statement
* PreparedStatement
* CallableStatement
* ResultSet

**Classes**:

* DriverManager
* SQLException

101. Explain the role of Driver in JDBC.

The JDBC Driver provides vendor-specific implementations of the abstract classes provided by the JDBC API. Each driver must provide implementations for the following interfaces of the java.sql package: [Connection](https://docs.oracle.com/javase/7/docs/api/java/sql/Connection.html),[Statement](https://docs.oracle.com/javase/7/docs/api/java/sql/Statement.html),[PreparedStatement](https://docs.oracle.com/javase/7/docs/api/java/sql/PreparedStatement.html),[CallableStatement](https://docs.oracle.com/javase/7/docs/api/java/sql/CallableStatement.html),[ResultSet](https://docs.oracle.com/javase/7/docs/api/java/sql/ResultSet.html) and[Driver](https://docs.oracle.com/javase/7/docs/api/java/sql/Driver.html).

102. What is JDBC Connection interface?

Connection interface maintains a session with the database. SQL statements are executed and results are returned within the context of a connection. A Connection object’s database is able to provide information describing its tables, its supported SQL grammar, its stored procedures, the capabilities of this connection, and so on. This information is obtained with the getMetaData method.

103. What does Connection pooling mean?

The interaction with a database can be costly, regarding the opening and closing of database connections. Especially, when the number of database clients increases, this cost is very high and a large number of resources is consumed.A pool of database connections is obtained at start up by the application server and is maintained in a pool. A request for a connection is served by a[connection residing in the pool](http://examples.javacodegeeks.com/enterprise-java/hibernate/hibernate-connection-pool-configuration-with-c3p0-example/). In the end of the connection, the request is returned to the pool and can be used to satisfy future requests.

104. What is the role of JDBC DriverManager class?

The DriverManager provides the user with a basic service for managing a set of JDBC drivers. It maintains contact with the available drivers and establishes a database connection with an appropriate one.

105. What is the purpose Class.forName method?

This method is used to load the driver that will establish a connection to the database.

Sample Class ClassLoader is shown below to demonstrate the usage of Class.forName() method.

*Class.forName*

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21 | public class ClassLoader  {       public static void main(String[] args) {          try        {           Class cls = Class.forName("BasicClass");             .....             System.out.println("Class = " + cls.getName());          }         catch(ClassNotFoundException exception)        {           System.out.println(exception.toString());        }    } |

106. What is the advantage of PreparedStatement over Statement?

PreparedStatement is precompiled and thus,[performance is much better](http://examples.javacodegeeks.com/core-java/sql/batch-statement-execution-example/). Also, PreparedStatement objects can be reused with different input values to their queries.

107. What is the use of CallableStatement?

A[CallableStatement](https://docs.oracle.com/javase/7/docs/api/java/sql/CallableStatement.html) is used to execute stored procedures. Stored procedures are stored and offered by a database. Stored procedures may take input values from the user and may return a result. The usage of stored procedures is highly encouraged, because it offers security and modularity.The method that prepares a CallableStatement is CallableStatement.prepareCall().

108. What do you mean by batch processing in JDBC?

Batch processing groups related SQL statements and execute multiple queries when the batch size reaches a desired threshold. This makes the performance faster.

J.Remote Method Invocation (RMI)

109. What is RMI?

The Java Remote Method Invocation (Java RMI) is a Java API that performs the object-oriented equivalent of remote procedure calls (RPC), with support for direct transfer of serialized Java classes and distributed garbage collection. Remote Method Invocation (RMI) can also be seen as the process of activating a method on a remotely running object. RMI offers location transparency because a user feels that a method is executed on a locally running object. Check some[RMI Tips here](https://www.javacodegeeks.com/2013/11/two-things-to-remember-when-using-java-rmi.html).

110. What is the basic principle of RMI architecture?

The RMI architecture is based on a very important principle which states that the definition of the behavior and the implementation of that behavior, are separate concepts. RMI allows the code that defines the behavior and the code that implements the behavior to remain separate and to run on separate JVMs.

RMI Architecture

111. What are the layers of RMI Architecture?

The RMI architecture consists of the following layers:

* Stub and Skeleton layer : This layer lies just beneath the view of the developer. This layer is responsible for intercepting method calls made by the client to the interface and redirect these calls to a remote RMI Service.
* Remote Reference Layer : The second layer of the RMI architecture deals with the interpretation of references made from the client to the server’s remote objects. This layer interprets and manages references made from clients to the remote service objects. The connection is a one-to-one (unicast) link.
* Transport layer : This layer is responsible for connecting the two JVM participating in the service. This layer is based on TCP/IP connections between machines in a network. It provides basic connectivity, as well as some firewall penetration strategies.

112. What is the role of Remote Interface in RMI?

The Remote interface serves to identify interfaces whose methods may be invoked from a non-local virtual machine. Any object that is a remote object must directly or indirectly implement this interface. A class that implements a remote interface should declare the remote interfaces being implemented, define the constructor for each remote object and provide an implementation for each remote method in all remote interfaces.

113. What is the role of the java.rmi.Naming Class?

The java.rmi.Naming class provides methods for storing and obtaining references to remote objects in the remote object registry. Each method of the Naming class takes as one of its arguments a name that is a String in URL format.

114. What is meant by binding in RMI?

Binding is the process of associating or registering a name for a remote object, which can be used at a later time, in order to look up that remote object. A remote object can be associated with a name using the bind or rebind methods of the Naming class.

115. What is the difference between using bind() and rebind() methods of Naming Class?

The bind method bind is responsible for binding the specified name to a remote object, while the rebind method is responsible for rebinding the specified name to a new remote object. In case a binding exists for that name, the binding is replaced.

116. What are the steps involved to make work a RMI program?

The following steps must be involved in order for a RMI program to work properly:

* Compilation of all source files.
* Generation of the stubs using rmic.
* Start the rmiregistry.
* Start the RMIServer.
* Run the client program.

RMI Flow

117. What is the role of stub in RMI?

A stub for a remote object acts as a client’s local representative or proxy for the remote object. The caller invokes a method on the local stub, which is responsible for executing the method on the remote object. When a stub’s method is invoked, it undergoes the following steps:

* It initiates a connection to the remote JVM containing the remote object.
* It marshals the parameters to the remote JVM.
* It waits for the result of the method invocation and execution.
* It unmarshals the return value or an exception if the method has not been successfully executed.
* It returns the value to the caller.

118. What is DGC and how does it work?

DGC stands for Distributed Garbage Collection. Remote Method Invocation (RMI) uses DGC for automatic garbage collection. Since RMI involves remote object references across JVMs, garbage collection can be quite difficult. DGC uses a reference counting algorithm to provide automatic memory management for remote objects.

119. What is the purpose of using RMISecurityManager in RMI?

RMISecurityManager provides a security manager that can be used by RMI applications, which use downloaded code. The class loader of RMI will not download any classes from remote locations, if the security manager has not been set.

120. Explain Marshalling and demarshalling.

When an application wants to pass its memory objects across a network to another host or persist it to storage, the in-memory representation must be converted to a suitable format. This process is called marshalling and the revert operation is called demarshalling.

121. Explain Serialization and Deserialization.

Java provides a mechanism, called object serialization where an object can be represented as a sequence of bytes and includes the object’s data, as well as information about the object’s type, and the types of data stored in the object. Thus, serialization can be seen as a way of flattening objects, in order to be stored on disk, and later, read back and reconstituted. Deserialisation is the reverse process of converting an object from its flattened state to a live object.

K.Servlets

122. What is a Servlet?

[The servlet](http://examples.javacodegeeks.com/enterprise-java/servlet/sample-java-servlet/) is a Java programming language class used to process client requests and generate dynamic web content. Servlets are mostly used to process or store data submitted by an HTML form, provide dynamic content and manage state information that does not exist in the stateless HTTP protocol.

123. Explain the architecture of a Servlet.

The core abstraction that must be implemented by all servlets is the javax.servlet.Servlet interface. Each servlet must implement it either directly or indirectly, either by extending javax.servlet.GenericServlet or javax.servlet.http.HTTPServlet. Finally, each servlet is able to serve multiple requests in parallel using multithreading.

Servlet Architecture

124. What is the difference between an Applet and a Servlet?

An Applet is a client side java program that runs within a Web browser on the client machine. On the other hand, a servlet is a server side component that runs on the web server.An applet can use the user interface classes, while a servlet does not have a user interface. Instead, a servlet waits for client’s HTTP requests and generates a response in every request.

125. What is the difference between GenericServlet and HttpServlet?

GenericServlet is a generalized and protocol-independent servlet that implements the Servlet and ServletConfig interfaces. Those servlets extending the GenericServlet class shall override the service method. Finally, in order to develop an HTTP servlet for use on the Web that serves requests using the HTTP protocol, your servlet must extend the HttpServlet instead. Check[Servlet examples here](http://examples.javacodegeeks.com/tag/servlet/).

126. Explain the life cycle of a Servlet.

On every client’s request, the Servlet Engine loads the servlets and invokes its init methods, in order for the servlet to be initialized. Then, the Servlet object handles all subsequent requests coming from that client, by invoking the service method for each request separately. Finally, the servlet is removed by calling the server’s destroy method.

Servlet Lifecycle

127. What is the difference between doGet() and doPost()?

doGET: The GET method appends the name-value pairs on the request’s URL. Thus, there is a limit on the number of characters and subsequently on the number of values that can be used in a client’s request. Furthermore, the values of the request are made visible and thus, sensitive information must not be passed in that way. doPOST: The POST method overcomes the limit imposed by the GET request, by sending the values of the request inside its body. Also, there is no limitations on the number of values to be sent across. Finally, the sensitive information passed through a POST request is not visible to an external client.

The code below shows the BasicServlet class which has doGet and doPost methods to be implemented.

*Get and Post methods*

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17 | public class BasicServlet extends HttpServlet  {       public void doGet(HttpServletRequest request, HttpServletResponse response)        throws ServletException, IOException        {          }        public void doPost(HttpServletRequest request, HttpServletResponse response)        throws ServletException, IOException        {          }    } |

128. What is meant by a Web Application?

A Web application is a dynamic extension of a Web or application server. There are two types of web applications: presentation-oriented and service-oriented. A presentation-oriented Web application generates interactive web pages, which contain various types of markup language and dynamic content in response to requests. On the other hand, a service-oriented web application implements the endpoint of a web service. In general, a Web application can be seen as a collection of servlets installed under a specific subset of the server’s URL namespace.

129. What is a Server Side Include (SSI)?

Server Side Includes (SSI) is a simple interpreted server-side scripting language, used almost exclusively for the Web, and is embedded with a servlet tag. The most frequent use of SSI is to include the contents of one or more files into a Web page on a Web server. When a Web page is accessed by a browser, the Web server replaces the servlet tag in that Web page with the hyper text generated by the corresponding servlet.

130. What is Servlet Chaining?

Servlet Chaining is the method where the output of one servlet is sent to a second servlet. The output of the second servlet can be sent to a third servlet, and so on. The last servlet in the chain is responsible for sending the response to the client.

131. How do you find out what client machine is making a request to your servlet?

The ServletRequest class has functions for finding out the IP address or host name of the client machine. getRemoteAddr() gets the IP address of the client machine and getRemoteHost() gets the host name of the client machine. See example[here](http://examples.javacodegeeks.com/enterprise-java/servlet/get-client-s-address-and-hostname-in-servlet/).

132. What is the structure of the HTTP response?

The HTTP response consists of three parts:

* Status Code: describes the status of the response. It can be used to check if the request has been successfully completed. In case the request failed, the status code can be used to find out the reason behind the failure. If your servlet does not return a status code, the success status code, HttpServletResponse.SC\_OK, is returned by default.
* HTTP Headers: they contain more information about the response. For example, the headers may specify the date/time after which the response is considered stale, or the form of encoding used to safely transfer the entity to the user. See[how to retrieve headers in Servlet here](http://examples.javacodegeeks.com/enterprise-java/servlet/get-all-request-headers-in-servlet/).
* Body: it contains the content of the response. The body may contain HTML code, an image, etc. The body consists of the data bytes transmitted in an HTTP transaction message immediately following the headers.

133. What is a cookie?

[A cookie](http://examples.javacodegeeks.com/core-java/net/urlconnection/get-cookies-from-http-connection/) is a bit of information that the Web server sends to the browser. The browser stores the cookies for each Web server in a local file. In a future request, the browser, along with the request, sends all stored cookies for that specific Web server.

134. What is the difference between session and cookie?

The differences between session and a cookie are the following:

* The session should work, regardless of the settings on the client browser. The client may have chosen to disable cookies. However, the sessions still work, as the client has no ability to disable them in the server side.
* The session and cookies also differ in the amount of information the can store. The HTTP session is capable of storing any Java object, while a cookie can only store String objects.

135. Which protocol will be used by browser and servlet to communicate?

The browser communicates with a servlet by using the HTTP protocol.

136. What is HTTP Tunneling?

HTTP Tunneling is a technique by which, communications performed using various network protocols are encapsulated using the HTTP or HTTPS protocols. The HTTP protocol therefore acts as a wrapper for a channel that the network protocol being tunneled uses to communicate. The masking of other protocol requests as HTTP requests is HTTP Tunneling.

137. What’s the difference between sendRedirect and forward methods?

The sendRedirect method creates a new request, while the forward method just forwards a request to a new target. The previous request scope objects are not available after a redirect, because it results in a new request. On the other hand, the previous request scope objects are available after forwarding. FInally, in general, the sendRedirect method is considered to be slower compare to the forward method.

|  |  |
| --- | --- |
| **SendRedirect** | **Forward** |
| This method sends a new request always. Th is because it uses the URL bar of the browser for redirecting. | This method sends the request to another resource by forwarding it. |
| This method is used at client side. | This method is usead at server side. |
| This method is used inside and outside the web server. | This method is used inside the web server only. |

138. What is URL Encoding and URL Decoding?

The URL encoding procedure is responsible for replacing all the spaces and every other extra special character of a URL, into their corresponding Hex representation. In correspondence, URL decoding is the exact opposite procedure.

139. What is Request Dispatcher?

[Servlet Request Dispatcher](https://examples.javacodegeeks.com/enterprise-java/servlet/java-servlet-requestdispatcher-tutorial/) is an interface whose implementation defines that an object can dispatch requests to any resource (such as HTML, Image, JSP, Servlet etc.) on the server. Another advantage of this interface is that it is used in two cases:

* To include the response of one Servlet into another (i.e. the client gets the response of both Servlets)
* To forward the client request to another Servlet to honor the request (i.e. the client calls a Servlet but the response to client is given by another Servlet)

L.JSP

140. What is a JSP Page?

A Java Server Page ([JSP](https://www.javacodegeeks.com/2015/06/jsp-tutorial.html)) is a text document that contains two types of text: static data and JSP elements. Static data can be expressed in any text-based format, such as HTML or XML. JSP is a technology that mixes static content with dynamically-generated content. See[JSP example here](http://examples.javacodegeeks.com/enterprise-java/jsp/sample-jsp-java-server-page/).

141. How are the JSP requests handled?

On the arrival of a JSP request, the browser first requests a page with a .jsp extension. Then, the Web server reads the request and using the JSP compiler, the Web server converts the JSP page into a servlet class. Notice that the JSP file is compiled only on the first request of the page, or if the JSP file has changed.The generated servlet class is invoked, in order to handle the browser’s request. Once the execution of the request is over, the servlet sends a response back to the client. See[how to get Request parameters in a JSP](http://examples.javacodegeeks.com/enterprise-java/jsp/get-request-parameter-in-jsp-page/).

142. What are the advantages of JSP?

The advantages of using the JSP technology are shown below:

* JSP pages are dynamically compiled into servlets and thus, the developers can easily make updates to presentation code.
* JSP pages can be pre-compiled.
* JSP pages can be easily combined to static templates, including HTML or XML fragments, with code that generates dynamic content.
* Developers can offer customized JSP tag libraries that page authors access using an XML-like syntax.
* Developers can make logic changes at the component level, without editing the individual pages that use the application’s logic.

143. What are Directives?

Directives are instructions that are processed by the JSP engine, when the page is compiled to a servlet. Directives are used to set page-level instructions, insert data from external files, and specify custom tag libraries.

144. What are the different types of Directives available in JSP?

Directives are defined between < %@ and % >.The different types of directives are shown below:

* Include directive: it is used to include a file and merges the content of the file with the current page.
* Page directive: it is used to define specific attributes in the JSP page, like error page and buffer.
* Taglib: it is used to declare a custom tag library which is used in the page.

145. What are JSP actions?

JSP actions use constructs in XML syntax to control the behavior of the servlet engine. They are executed when a JSP page is requested. They can be dynamically inserted into a file, re-use JavaBeans components, forward the user to another page, or generate HTML for the Java plugin.Some of the available actions are listed below:

* jsp:include includes a file, when the JSP page is requested.
* jsp:useBean finds or instantiates a JavaBean.
* jsp:setProperty sets the property of a JavaBean.
* jsp:getProperty gets the property of a JavaBean.
* jsp:forward forwards the requester to a new page.
* jsp:plugin generates browser-specific code.

146. What are Scriptlets?

In Java Server Pages (JSP) technology, a scriptlet is a piece of Java-code embedded in a JSP page. The scriptlet is everything inside the tags. Between these tags, a user can add any valid scriptlet.

147. What are Declarations?

Declarations are similar to variable declarations in Java. Declarations are used to declare variables for subsequent use in expressions or scriptlets. To add a declaration, you must use the sequences to enclose your declarations.

Sample code is added below to show the JSP declarations.

*Declarations*

|  |  |
| --- | --- |
| 1  2  3 | <%! int j = 0; %>  <%! int d, e, f; %>  <%! Shape a = new Shape(); %> |

148. What are Expressions?

A JSP expression is used to insert the value of a scripting language expression, converted into a string, into the data stream returned to the client, by the web server. Expressions are defined between <% = and %> tags.

*JSP Expresssion*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <html>     <head><title>My Blog</title></head>       <body>        <p>Today's Date is: <%= (new java.util.Date()).toLocaleString()%></p>     </body>  </html> |

149. What is meant by implicit objects and what are they?

JSP implicit objects are those Java objects that the JSP Container makes available to developers in each page. A developer can call them directly, without being explicitly declared. JSP Implicit Objects are also called pre-defined variables.The following objects are considered implicit in a JSP page:

* application
* page
* request
* response
* session
* exception
* out
* config
* pageContext

JSP sample tags for disabling session is shown below:

*Disabling Session*

|  |  |
| --- | --- |
| 1 | <%@ page session=“false” %> |

150. What are the different tags provided in JSTL?

There are 5 type of JSTL tags:

**Core:**

* Variable support
* Flow control
* URL management
* Miscellaneous

**XML:**

* Core
* Flow control
* Transformation

**Internationalization:**

* Locale
* Message formatting
* Number and date formatting

**Database:**

* SQL

**Functions:**

* Collection length
* String manipulation

More Tutorials

* [What is Java used for](https://examples.javacodegeeks.com/what-is-java-used-for/)
* [Best Way to Learn Java Programming Online](https://www.javacodegeeks.com/learn-java-programming-online.html)
* [Java Tutorial for Beginners](https://examples.javacodegeeks.com/java-tutorial-for-beginners/)
* [Java Array – java.util.Arrays Example](https://examples.javacodegeeks.com/java-array-example/)
* [Java String Class Example](https://examples.javacodegeeks.com/java-string-class-example/)
* [Java List Example](https://examples.javacodegeeks.com/java-list-example/)
* [Java Constructor Example](https://examples.javacodegeeks.com/java-constructor-example/)
* [Try Catch Java Example](https://examples.javacodegeeks.com/try-catch-java-example/)
* [Polymorphism Java Example](https://examples.javacodegeeks.com/polymorphism-java-example/)
* [Java Collections Tutorial](https://examples.javacodegeeks.com/java-collections-tutorial/)
* [Java Set Example](https://examples.javacodegeeks.com/java-set-example/)
* [LinkedList Java Example](https://examples.javacodegeeks.com/linkedlist-java-example/)
* [ArrayList Java Example – How to use arraylist](https://examples.javacodegeeks.com/arraylist-java-example/)
* [Hashmap Java Example](https://examples.javacodegeeks.com/hashmap-java-example/)
* [String CompareTo Java Example](https://examples.javacodegeeks.com/string-compareto-java-example/)
* [Hashset Java Example](https://examples.javacodegeeks.com/hashset-java-example/)
* [java.lang.NullPointerException Example – How to handle Java Null Pointer Exception](https://examples.javacodegeeks.com/java-lang-nullpointerexception-example/)

Want more Java interview questions?

Are you still with us? Wow, that was a huge article about different types Java Interview Questions. If you enjoyed this, then [**subscribe to our newsletter**](http://eepurl.com/xRlkj) to enjoy weekly updates and complimentary whitepapers! Also, check out our [**courses**](http://academy.javacodegeeks.com/) for more advanced training!

*So, what other Java Interview Questions are there? Let us know in the comments and we will include them in the article! Happy coding!*

**Last updated on Apr. 17th, 2021**

Tagged with: [INTERVIEW](https://www.javacodegeeks.com/tag/interview) [INTERVIEW QUESTIONS](https://www.javacodegeeks.com/tag/interview-questions) [JAVA APPLETS](https://www.javacodegeeks.com/tag/java-applets) [JDBC](https://www.javacodegeeks.com/tag/jdbc) [JSP](https://www.javacodegeeks.com/tag/jsp) [RMI](https://www.javacodegeeks.com/tag/rmi) [SERVLETS](https://www.javacodegeeks.com/tag/servlets) [SWING](https://www.javacodegeeks.com/tag/swing) [ULTIMATE](https://www.javacodegeeks.com/tag/ultimate)

 (***+6****rating,****8****votes*)  
*You need to be a registered member to rate this.* [154 Comments](https://www.javacodegeeks.com/java-interview-questions.html#comments) 47940 Views [Tweet it!](https://twitter.com/share?url=https://www.javacodegeeks.com/java-interview-questions.html&via=javacodegeeks&text=150%20Java%20Interview%20Questions%20and%20Answers%20%E2%80%93%20The%20ULTIMATE%20List%20(PDF%20Download))

Do you want to know how to develop your skillset to become a Java Rockstar?

Subscribe to our newsletter to start Rocking right now!

To get you started we give you our best selling eBooks for FREE!

**1.** JPA Mini Book

**2.** JVM Troubleshooting Guide

**3.** JUnit Tutorial for Unit Testing

**4.** Java Annotations Tutorial

**5.** Java Interview Questions

**6.** Spring Interview Questions

**7.** Android UI Design

and many more ....



I agree to the [Terms](https://www.javacodegeeks.com/about/terms-of-use)and [Privacy Policy](https://www.javacodegeeks.com/about/privacy-policy)

[**Sign up**](https://www.javacodegeeks.com/java-interview-questions.html)

LIKE THIS ARTICLE? READ MORE FROM JAVA CODE GEEKS

 Subscribe

Top of Form

{}[+]

Bottom of Form

This site uses Akismet to reduce spam. [Learn how your comment data is processed](https://akismet.com/privacy/).

**154 COMMENTS**

Oldest

[**aerobless**](http://ghoul.io/)

 7 years ago

That list seems really good. I study computer science and it’ll definitely come in handy once I start looking for a job.

0

 Reply

[**tomi crow**](https://www.javacodegeeks.com/user/tomicrow/)

 4 years ago

*Reply to*[*aerobless*](https://www.javacodegeeks.com/java-interview-questions.html#comment-8706)

Hope this will help you for java interview more…

<http://net-informations.com/java/cjava/default.htm>

Crow

1

 Reply

**user**

 1 year ago

*Reply to*[*aerobless*](https://www.javacodegeeks.com/java-interview-questions.html#comment-8706)

good luck for you

0

 Reply

**leon**

 7 years ago

One small error on hashCode and equals: if 2 instances are equal, they must provide the same hashCode, BUT two different instances may return the same hashCode.

Having hashCode always return 1 or some other fixed number satisfies the contract, although it would kill the performance of HashMap and other hash related functionality.

The hashCode just determines in which hashmap bucket the instance is put, and within that bucket the correct instance is then looked up with the much more expensive equals.

1

 Reply

**teoman**

 7 years ago

perfect post, thanks for sharing.

0

 Reply

**Madiraju Krishna Chaitanya**

 6 years ago

Nice Post.Thanks for sharing this with us.

0

 Reply

**Sumit Bisht**

 6 years ago

This is correct, but many topics are outdated here.

2

 Reply

**Jeffrey Burch**

 6 years ago

*Reply to*[*Sumit Bisht*](https://www.javacodegeeks.com/java-interview-questions.html#comment-17715)

Yeah, there should be some questions about Java 8 features. It will tell if the candidate puts any effort into self improvement.

2

 Reply

**Nikitha**

 6 years ago

Really perfect post, thanks for sharing with us.

0

 Reply

[**Robert Martin**](http://robertmartin123.tk/)

 6 years ago

This is a quality peace of work. Thank you for the effort.

0

 Reply

**Vlad**

 6 years ago

Thanks to the author  
I appreciate your efforts to write everything in one place

0

 Reply

**raghu kumar challa**

 6 years ago

good very useful to all fresh engineering graduates and experianced to get into job easily.

-1

 Reply

**M.Hagras**

 6 years ago

Thanks a lot, it is very useful

0

 Reply

**Tk**

 6 years ago

Very good work. It would be really great if you could add more questions especially on the new versions of Java. Also, few implementation examples can be posted for different concept. You have done a great job.

0

 Reply

**RAVIKUMAR.M**

 6 years ago

It is very useful for job seekers to get a great job. thanks for sharing with us.

-1

 Reply

**Ruth**

 6 years ago

Thanks so much for this compilation. It is very useful.

0

 Reply

**Nazim**

 6 years ago

Thanks, but found some inaccuracies:  
Q33 Other threads are able to modify Collection of some Iterator, but next called method on Iterator will throw ConcurrentModificationException  
Q37 It is recommended not to override finalize() method in order to release resource as JVM does not guarantee this method to be invoked.

0

 Reply

**satish shirale**

 6 years ago

Thanks, I refreshed my java knowledge once again.

0

 Reply

**Guarav**

 6 years ago

Thanks … very useful

0

 Reply

**Madhu**

 6 years ago

Thanks to the author ! Keep updating !

1

 Reply

**Jun**

 6 years ago

nice job. Thank You. This tutorial is very useful for people who seek job in programming. I just love coding.

0

 Reply

**Akansha**

 6 years ago

nice infrmation

0

 Reply

**ِAhmed Mansour**

 6 years ago

nice infrmation

0

 Reply

Load More Comments

NEWSLETTER

Insiders are already enjoying weekly updates and complimentary whitepapers!

**Join them now** to gain exclusive access to the latest news in the Java world, as well as insights about Android, Scala, Groovy and other related technologies.



I agree to the [Terms](https://www.javacodegeeks.com/about/terms-of-use)and [Privacy Policy](https://www.javacodegeeks.com/about/privacy-policy)

[**Sign up**](https://www.javacodegeeks.com/java-interview-questions.html)

JOIN US

With 1,240,600 monthly unique visitors and over 500 authors we are placed among the top Java related sites around. Constantly being on the lookout for partners; we encourage you to join us. So If you have a blog with unique and interesting content then you should check out our [**JCG**](https://www.javacodegeeks.com/join-us/jcg) partners program. You can also be a [**guest writer**](https://www.javacodegeeks.com/join-us/w4g) for Java Code Geeks and hone your writing skills!

KNOWLEDGE BASE

* [Courses](https://courses.javacodegeeks.com/)
* [Examples](https://examples.javacodegeeks.com/)
* [Minibooks](https://www.javacodegeeks.com/minibook)
* [Resources](https://www.javacodegeeks.com/resources)
* [Tutorials](https://www.javacodegeeks.com/tutorials)

PARTNERS

* [Mkyong](https://www.mkyong.com/)

THE CODE GEEKS NETWORK

* [.NET Code Geeks](https://www.dotnetcodegeeks.com/)
* [Java Code Geeks](https://www.javacodegeeks.com/)
* [System Code Geeks](https://www.systemcodegeeks.com/)
* [Web Code Geeks](https://www.webcodegeeks.com/)

HALL OF FAME

* [“Android Full Application Tutorial” series](https://www.javacodegeeks.com/2010/10/android-full-application-tutorial.html)
* [11 Online Learning websites that you should check out](https://www.javacodegeeks.com/2013/01/15-online-learning-websites-that-you-should-check-out.html)
* [Advantages and Disadvantages of Cloud Computing – Cloud computing pros and cons](https://www.javacodegeeks.com/2013/04/advantages-and-disadvantages-of-cloud-computing-cloud-computing-pros-and-cons.html)
* [Android Google Maps Tutorial](https://www.javacodegeeks.com/2011/02/android-google-maps-tutorial.html)
* [Android JSON Parsing with Gson Tutorial](https://www.javacodegeeks.com/2011/01/android-json-parsing-gson-tutorial.html)
* [Android Location Based Services Application – GPS location](https://www.javacodegeeks.com/2010/09/android-location-based-services.html)
* [Android Quick Preferences Tutorial](https://www.javacodegeeks.com/2011/01/android-quick-preferences-tutorial.html)
* [Difference between Comparator and Comparable in Java](https://www.javacodegeeks.com/2013/03/difference-between-comparator-and-comparable-in-java.html)
* [GWT 2 Spring 3 JPA 2 Hibernate 3.5 Tutorial](https://www.javacodegeeks.com/2010/05/gwt-2-spring-3-jpa-2-hibernate-35-2.html)
* [Java Best Practices – Vector vs ArrayList vs HashSet](https://www.javacodegeeks.com/2010/08/java-best-practices-vector-arraylist.html)

ABOUT JAVA CODE GEEKS

JCGs (Java Code Geeks) is an independent online community focused on creating the ultimate Java to Java developers resource center; targeted at the technical architect, technical team lead (senior developer), project manager and junior developers alike. JCGs serve the Java, SOA, Agile and Telecom communities with daily news written by domain experts, articles, tutorials, reviews, announcements, code snippets and open source projects.

DISCLAIMER

All trademarks and registered trademarks appearing on Java Code Geeks are the property of their respective owners. Java is a trademark or registered trademark of Oracle Corporation in the United States and other countries. Examples Java Code Geeks is not connected to Oracle Corporation and is not sponsored by Oracle Corporation.

Java Code Geeks and all content copyright © 2010-2021, [Exelixis Media P.C.](https://www.exelixismedia.com/) | [Terms of Use](https://www.javacodegeeks.com/about/terms-of-use) | [Privacy Policy](https://www.javacodegeeks.com/about/privacy-policy) | [Contact](mailto:support@javacodegeeks.com)

[Remote jobs](https://arc.dev/remote-jobs)

[Remote companies](https://arc.dev/companies)

**Resources**

[For employers](https://arc.dev/hire)

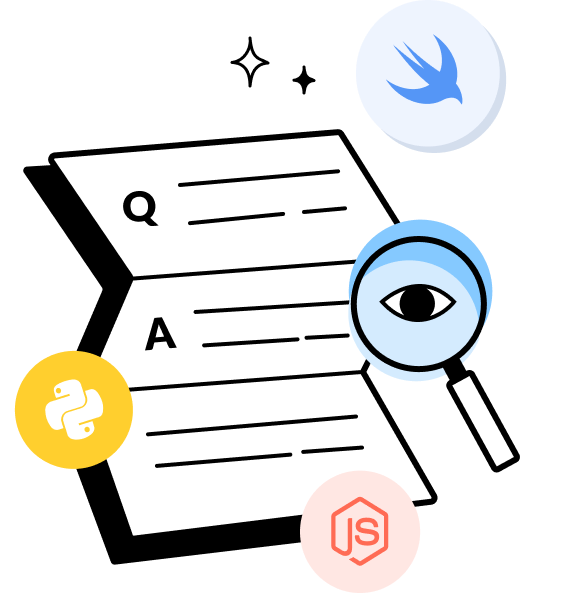
SIGN UPLOG IN

[Arc Developers](https://arc.dev/)>[Interview](https://arc.dev/interview)>Java interview questions and answers

**Java Interview Questions and Answers You Should Know**

Preparing for a Java interview? Here are the top Java interview questions that you should know — and how to answer them.

Whether you are going to a junior Java developer interview or a senior Java developer interview, use these commonly-asked interview questions and answers to help you successfully land the job!



**🔥 Java jobs — hiring right now**



[**Revolut**](https://arc.dev/companies/revolut-alufu52r6z)

Insurance

More than 1000

[Java Backend Engineer - Hungary Remotely](https://arc.dev/remote-jobs/j/revolut-java-backend-engineer-hungary-remotely-bj9hbw6j85)[Java Backend Engineer - Romania Remotely](https://arc.dev/remote-jobs/j/revolut-java-backend-engineer-romania-remotely-at6inpo4pu)[Java Backend Engineer - Ukraine Remotely](https://arc.dev/remote-jobs/j/revolut-java-backend-engineer-ukraine-remotely-arf3tn5txx)



[**ThoughtWorks**](https://arc.dev/companies/thoughtworks-aluftvbrfi)

Internet Software & Services

More than 1000

[Senior Java Engineer (Remote)](https://arc.dev/remote-jobs/j/thoughtworks-senior-java-engineer-remote-bvsw07ydrj)[Senior Java Engineer - Full-stack (Remote)](https://arc.dev/remote-jobs/j/thoughtworks-senior-java-engineer-full-stack-remote-bx1xgx6rl9)[Senior Java Engineer (Full-stack)](https://arc.dev/remote-jobs/j/thoughtworks-senior-java-engineer-full-stack-bvsp9cyf59)



[**Scalable Capital**](https://arc.dev/companies/scalable-capital-alufugf5x7)

Financial Services

51 - 250

[Java Engineer (m/f/x) onsite or remote (anywhere in Germany or UK)](https://arc.dev/remote-jobs/j/scalable-capital-gmbh-java-engineer-m-f-x-onsite-or-remote-anywhere-in-germany-or-uk-ahg2yo8yky)[Java Engineer (m/f/x) remote anywhere in Germany or onsite](https://arc.dev/remote-jobs/j/scalable-capital-gmbh-java-engineer-m-f-x-remote-anywhere-in-germany-or-onsite-a3l8ysu6tf)

**1931** more Java Jobs

[View all jobs](https://arc.dev/remote-jobs/java)

Need to interview a Java developer for a freelance project or job? Here are 37 essential interview questions (and answers!) provided by some of our top Java experts on Codementor.

Although technical interviews can't really gauge how well a candidate would perform on a real-life project, this is still an integral part of the hiring process. Here are some Java interview questions that you can ask a developer to evaluate their understanding of the language.

**Java Interview Question #1**

**What’s the difference between String, StringBuffer andStringBuilder?**  
*(Question provided by*[*Matt Goldpink*](https://www.codementor.io/mattgoldspink)*)*

String is an immutable class. In older JDK’s the recommendation when programmatically building a String was to use StringBuffer since this was optimized to concatenate multiple Strings together. However, the methods on StringBuffer were marked as sychronized, which meant that there was a performance penalty, hence StringBuilder was introduced to provide a non-synchronized way to efficiently concatenate and modify Strings.

[**¶**](https://arc.dev/interview/java-interview-questions-and-answers#java-interview-question-%232)**Java Interview Question #2**

**How do you run a Java application on the command line and set the classpath with multiple jars?**  
*(Question provided by*[*Matt Goldpink*](https://www.codementor.io/mattgoldspink)*)*

Some people will be thinking “what!?” but I’ve met a lot of Java developers who’ve not run a Java application outside of an IDE for years.

java -cp /dev/myapp.jar:/dev/mydependency.jar com.codementor.MyApp

**Java Interview Question #3**

**What is the difference between final, finalize and finally?**  
*(Question provided by*[*Matt Goldpink*](https://www.codementor.io/mattgoldspink)*)*

final is a Java keyword used to indicate that either a method can not override in a subclass, or a class can not be extended or a field can not be modified. finalize is a method that gets called on an instance of an Object when it is garbage collected. finally is a Java keyword used in exception handling to indicate a block of code that should always be run whether an exception is thrown or not.

**Java Interview Question #4**

**How does Garbage Collection prevent a Java application from going out of memory?**  
*(Question provided by*[*Matt Goldpink*](https://www.codementor.io/mattgoldspink)*)*

This is a tricky one… it doesn’t! Garbage Collection simply cleans up unused memory when an object goes out of scope and is no longer needed. However an application could create a huge number of large objects that causes an OutOfMemoryError.

**Java Interview Question #5**

**What’s the difference between a ClassNotFoundException and NoClassDefFoundError?**  
*(Question provided by*[*Matt Goldpink*](https://www.codementor.io/mattgoldspink)*)*

A ClassNotFoundException means the class file for a requested class is not on the classpath of the application. A NoClassDefFoundErrormeans that the class file existed at runtime, but for some reason the class could not be turned into a Class definition. A common cause is an exception being thrown in static initialization blocks.



***Author Bio***

*Matt is a Lead engineer with 10+ years development experience, and has been giving technical interviews for 12 years now.*[*Hire Matt Now*](https://www.codementor.io/mattgoldspink/profile)*.*

**Java Interview Question #6**

**Why isn’t String‘s .length() accurate?**  
*(Question provided by*[*Francis Galiegue*](https://www.codementor.io/fge)*)*

It isn’t accurate because it will only account for the number of characters within the String. In other words, it will fail to account for code points outside of what is called the BMP (Basic Multilingual Plane), that is, code points with a value of U+10000 or greater.

The reason is historical: when Java was first defined, one of its goal was to treat all text as Unicode; but at this time, Unicode did not define code points outside of the BMP. By the time Unicode defined such code points, it was too late for char to be changed.

This means that code points outside the BMP are represented with two chars in Java, in what is called a **surrogate pair**. Technically, a char in Java is a UTF-16 code unit.

The correct way to count the real numbers of characters within a String, i.e. the number of code points, is either:

someString.codePointCount(0, someString.length())

or, with Java 8:

someString.codePoints().count()

**Java Interview Question #7**

**Given two double values d1, d2, why isn’t it reliable to test their equality using:**  
*(Question provided by*[*Francis Galiegue*](https://www.codementor.io/fge)*)*

d1 == d2

Because of Double.NaN (literally: “Not a Number”).

This code:

**final** **double** d1 = Double.NaN;

**final** **double** d2 = Double.NaN;

System.out.println(d1 == d2);

will print false.

The most accurate way to tell whether two double values are equal to one another is to use Double.compare() and test against 0, as in:

System.out.println(Double.compare(d1, d2) == 0);

**Java Interview Question #8**

**What is the problem with this code:**  
*(Question provided by*[*Francis Galiegue*](https://www.codementor.io/fge)*)*

**final** **byte**[] bytes = someString.getBytes();

There are, in fact, two problems:

* the code relies on the default Charset of the JVM;
* it supposes that this default Charset can handle all characters.

While the second problem is rarely a concern, the first certainly is a concern.

For instance, in most Windows installations, the default charset is CP1252; but on Linux installations, the default charset will be UTF-8.

As such, such a simple string as “é” will give a different result for this operation depending on whether this code is run on Windows or Linux.

The solution is to always specify a Charset, as in, for instance:

final byte[] bytes = someString.getBytes(StandardCharsets.UTF\_8);

**Java Interview Question #9**

**What is the JIT?**  
*(Question provided by*[*Francis Galiegue*](https://www.codementor.io/fge)*)*

The JIT is the JVM’s mechanism by which it can optimize code at runtime.

JIT means Just In Time. It is a central feature of any JVM. Among other optimizations, it can perform code inlining, lock coarsening or lock eliding, escape analysis etc.

The main benefit of the JIT is on the programmer’s side: code should be written so that it just works; if the code can be optimized at runtime, more often than not, the JIT will find a way.

(On a more advanced note: the JIT is such a complex piece of machinery that it makes it complicated to do accurate performance benchmarks for JVM code; this is why such frameworks as JMH exist.)

**Java Interview Question #10**

*(Question provided by*[*Francis Galiegue*](https://www.codementor.io/fge)*)*

**This code:**

**final** **double** d = 1 / 2;

System.out.println(d);

**prints 0. Why? How do you make this code print 0.5 instead?**

The problem here is that this expression:

1 / 2

has integer literals on both sides of the operator: 1 and 2. As a consequence, an integer division will be performed, and the result of 1 divided by 2 in an integer division is 0.

In order for the result to be a double as expected, at least one operand of the operation needs to be a double. For instance:

**final** **double** d = 1 / 2.0;

or:

**final** **double** d = 1.0 / 2;

**Java Interview Question #11**

*(Question provided by*[*Francis Galiegue*](https://www.codementor.io/fge)*)*

**In this code:**

IntStream.range(0, 10).forEach(System.out::println);

**what is the inferred type of the method reference System.out::println?**

It is an IntConsumer.

IntStream.range(0, 10) returns an IntStream, and IntStream defines a .forEach() method accepting an IntConsumer as an argument, whose prototype is:

**void** **accept**(**int** value);

System.out is a PrintStream, and a PrintStream has a method named println which takes an int as an argument and returns void. This matches the signature of an IntConsumer, hence the result.

**Java Interview Question #12**

**What is the problem with this code?**  
*(Question provided by*[*Francis Galiegue*](https://www.codementor.io/fge)*)*

**final** Path path = Paths.get(...);

Files.lines(path).forEach(System.out::println);

The problem is that the Stream returned by Files.lines() is not closed.

This should be used instead:

**try** (

**final** Stream<String> stream = Files.lines(path);

) {

stream.forEach(System.out::println);

}

Stream extends BaseStream, and BaseStream extends AutoCloseable. While this has no influence on streams you obtain from collections for instance, the stream returned by Files.lines() is I/O bound. Neglecting to close it correctly may lead to a resource leak in the event of an error occurring while processing the stream.

**Java Interview Question #13**

**Consider the following piece of code:**  
*(Question provided by*[*Francis Galiegue*](https://www.codementor.io/fge)*)*

**final** List<Integer> list = **new** ArrayList<>();

list.add(1);

list.add(2);

list.add(3);

list.remove(2);

**What will be the contents of the list after this operation and why?**

The contents will be:

[ 1, 2 ]

The reason is that there are two removal operations on a List:

* remove(int index)
* remove(Object obj)

The JVM will always select the most specific overload of a method; and here we pass an int as an argument, the code therefore removes the element at index 2.

To remove the \_element\_ 2 from the list, the following needs to be written:

list.remove(Integer.valueOf(2));



***Author Bio***

*Francis is an open source Java developer, and a*[*top 0.25% StackOverflow User*](http://stackoverflow.com/users/1093528/fge)*. He's also an author of two of the three JSON Schema IETF drafts.*[*Hire Francis Now*](https://www.codementor.io/fge/profile)*.*

[](https://www.codementor.io/java-developers)

***Find top Java developers today.***

*CodementorX will find you the best engineers for your project.*

**Java Interview Question #14**

**Write a function to detect if two strings are anagrams (for example, SAVE and VASE)**  
*(Question provided by*[*Codementor Steven Noto*](https://www.codementor.io/stevennoto)*)*

This is my go-to first interview question. It helps me gauge a candidate’s ability to understand a problem and write an algorithm to solve it.

If someone has not solved the problem before, I expect to see some code with loops and if/then’s. Maybe some HashMaps. The things I look for are ability to break down the problem to see what you need to check, what the edge cases are, and whether the code meets those criteria.

The naive solution is often to loop through the letters of the first string and see if they’re all in the second string. The next thing to look for is, the candidate should also do that in reverse too (check string 1 for string 2’s letters)? The next thing to look for is, what about strings with duplicate letters, like VASES?

If you can realize that these are all required and create a functional, non-ridiculous solution, I am happy.

Of course, one can solve it trivially by sorting both strings and comparing them. If someone catches this right away, usually they have seen the problem before. But that’s a good sign that someone cares enough to do prep work. Then we can tackle a harder problem.

**public** **static** **boolean** **isAcronym**(String s1, String s2) {

**if** (s1.length() != s2.length()) **return** **false**;

HashMap<Character, Integer> charCounts = **new** HashMap<>();

*// Calculate chracter counts*

**for** (**int** i = 0; i < s1.length(); i++) {

**if** (charCounts.containsKey(s1.charAt(i))) {

charCounts.put(s1.charAt(i), charCounts.get(s1.charAt(i)) + 1);

} **else** {

charCounts.put(s1.charAt(i), 1);

}

}

*// Compare counts with characters in s2*

**for** (**int** i = 0; i < s2.length(); i++) {

**if** (charCounts.containsKey(s2.charAt(i))) {

charCounts.put(s2.charAt(i), charCounts.get(s2.charAt(i)) - 1);

} **else** {

**return** **false**;

}

}

*// Check all letters matched*

**for** (**int** count : charCounts.values()) {

**if** (count != 0) **return** **false**;

}

**return** **true**;

}

The details of the implementation are not important; what’s important is someone understanding what they need to do, and then understanding why their solution works or doesn’t work. If you can demonstrate this, you’re on the right track.

Here is one way to implement a better solution, comparing sorted strings:

**public** **static** **boolean** **isAcronymMoreBetter**(String s1, String s2) {

**char**[] s1Chars = s1.toCharArray();

**char**[] s2Chars = s2.toCharArray();

Arrays.sort(s1Chars);

Arrays.sort(s2Chars);

**return** Arrays.equals(s1Chars, s2Chars);

}



***Author Bio***

*Steven is a lead software engineer proficient in enterprise application development. He has 15 years of experience designing, developing, and managing.*[*Hire Steven Now*](https://www.codementor.io/stevennoto/profile)*.*

**Java Interview Question #15**

**What is the contract between equals and hashCode of an object?**  
*(Question provided by*[*Akmal Muqeeth*](https://www.codementor.io/akmal_muqeeth)*)*

The only obligation is that for any objects o1 and o2 then if o1.equals(o2) is true then o1.hashCode() == o2.hashCode() is true.

Note that this relationship goes only one way: for any o1, o2 of some class C, where none of o1 and o2 are null, then it can happen that o1.hashCode() == o2.hashCode() is true BUT o1.equals(o2) is false.

**Java Interview Question #16**

**Can an enum be extended?**  
*(Question provided by*[*Akmal Muqeeth*](https://www.codementor.io/akmal_muqeeth)*)*

No. Enum types are final by design.

**Java Interview Question #17**

**How threadsafe is enum in Java?**  
*(Question provided by*[*Akmal Muqeeth*](https://www.codementor.io/akmal_muqeeth)*)*

Creation of an enum is guaranteed to be threadsafe. However, the methods on an enum type are not necessarily threadsafe

**Java Interview Question #18**

**How does the JVM handle storing local variables vs storing objects?**  
*(Question provided by*[*Akmal Muqeeth*](https://www.codementor.io/akmal_muqeeth)*)*

Objects are stored on the heap. Variables are a reference to the object.

Local variables are stored on the stack.

**Java Interview Question #19**

**Identify the problem in the below code:**  
*(Question provided by*[*Akmal Muqeeth*](https://www.codementor.io/akmal_muqeeth)*)*

**public** **class** **Foo** {

**public** **Foo**() {

doSomething();

}

**public** **void** **doSomething**() {

System.out.println("do something acceptable");

}

}

**public** **class** **Bar** **extends** **Foo** {

**public** **void** **doSomething**() {

System.out.println("yolo");

Zoom zoom = **new** Zoom(**this**);

}

}

Classic example for escaping references.

When an object of Bar is created, the super constructor in Foo gets called first, which in turn calls the ‘overridden’ doSomething method.  
The doSomething method passes the this instance to the class Zoom. Zoom now can use the ‘this‘ instance before it is created entirely. BAD!!!  
([source](http://stackoverflow.com/questions/20474521/allowing-the-this-reference-to-escape))

***Author Bio***

*Akmal is a senior software engineer at Brighter.com with experience primarily with developing backend for web applications using Java technologies.*[*Hire Akmal Now*](https://www.codementor.io/akmal_muqeeth/profile)*.*

**Java Interview Question #20**

**When do you use volatile variables?**  
*(Question provided by*[*Anubhava Srivastava*](https://www.codementor.io/bantoo)*)*

When a member variable is accessed by multiple threads and want the value of a volatile field to be visible to all readers (other threads in particular) after a write operation completes on it.

**Java Interview Question #21**

**Why do you need to use synchronized methods or blocks?**  
*(Question provided by*[*Anubhava Srivastava*](https://www.codementor.io/bantoo)*)*

If threads are being used and a number of threads have to go through a synchronized section of code, only one of them may be executed at a time. This is used to make sure shared variables are not updated by multiple threads.

**Java Interview Question #22**

**What is the difference between HashMap and ConcurrentHashMap?**  
*(Question provided by*[*Anubhava Srivastava*](https://www.codementor.io/bantoo)*)*

ConcurrentHashMap is thread-safe; that is the code can be accessed by single thread at a time while HashMap is not thread-safe. ConcurrentHashMap does not allow NULL keys while HashMap allows it.

**Java Interview Question #23**

**When do you need to override the equals and hashCode methods in Java?**  
*(Question provided by*[*Anubhava Srivastava*](https://www.codementor.io/bantoo)*)*

By defining equals() and hashCode() consistently, the candidate can improve the usability of classes as keys in hash-based collections such as HashMap.

**Java Interview Question #24**

**What is a Service?**  
*(Question provided by*[*Anubhava Srivastava*](https://www.codementor.io/bantoo)*)*

A service is a function that is well-defined, self-contained, and does not depend on the context or state of other services.

**Java Interview Question #25**

**What is a good usecase of calling System.gc()?**  
*(Question provided by*[*Anubhava Srivastava*](https://www.codementor.io/bantoo)*)*

One may call System.gc() when profiling an application to search for possible memory leaks. All the profilers call this method just before taking a memory snapshot.

**Java Interview Question #26**

**What is the marker interface in Java?**  
*(Question provided by*[*Anubhava Srivastava*](https://www.codementor.io/bantoo)*)*

The marker interface in Java is an interfaces with no field or methods. In other words, it an empty interface in java is called a marker interface. An example of a marker interface is a Serializable, Clonable and Remote interface. These are used to indicate something to the compiler or JVM.

**Java Interview Question #27**

**How are Annotations better than a Marker Interfaces?**  
*(Question provided by*[*Anubhava Srivastava*](https://www.codementor.io/bantoo)*)*

Annotations lets one achieve the same purpose of conveying metadata about the class to its consumers without creating a separate type for it. Annotations are more powerful, too, letting programmers pass more sophisticated information to classes that “consume” it.



***Author Bio***

*Anubhava is a Lead Engineer-Architect at AOL Inc. He specializes in JavaScript, PHP, and Java.*[*Hire Anubhava Now*](https://www.codementor.io/bantoo/profile)*.*

[](https://www.codementor.io/java-developers)

***Find top Java developers today.***

*CodementorX will help you find you the best engineers for your project.*

**Java Interview Question #28**

**What are checked and unchecked exceptions? When do you use them?**  
*(Question provided by*[*Tung Dao*](https://www.codementor.io/tungd)*)*

A checked exception is an exception that must be catch, they are checked by the compiler. An unchecked exception is mostly runtime exception, and is not required to be catch. In general, use checked exception when the situation is recoverable (retry, display reasonable error message).



***Author Bio***

*Tung Dao is a*[*top 4% StackOverflow Users*](http://stackoverflow.com/users/401539/tungd)*and an experienced full-stack developer.*[*Hire Tung Dao Now*](https://www.codementor.io/tungd/profile)*.*

**Java Interview Question #29**

**int a = 1L; won’t compile and int b = 0; b += 1L; compiles fine. Why ?**  
*(Question provided by*[*Suresh Atta*](https://www.codementor.io/sureshatta)*)*

When += is used, that’s a compound statement and the compiler internally casts it. Whereas in the first case, the compiler straightaway shouts at you since it is a direct statement.

Compiler behavior and statement types can be confusing, so questions like this will test a candidate's grasp of these concepts.

**Java Interview Question #30**

**Why aren’t you allowed to extend more than one class in Java but are allowed to implement multiple interfaces?**  
*(Question provided by*[*Suresh Atta*](https://www.codementor.io/sureshatta)*)*

Extending classes may cause ambiguity problems. On the other hand, in terms of interfaces, the single method implementation in one class can serve more than one interfaces.

**Java Interview Question #31**

**Why doesn’t the following code generate a NullPointerException even when the instance is null?**  
*(Question provided by*[*Suresh Atta*](https://www.codementor.io/sureshatta)*)*

Test t = **null**;

t.someMethod();

**public** **static** **void** **someMethod**() {

...

}

There is no need for an instance while invoking a static member or method, since static members belongs to a class rather than an instance.

A null reference may be used to access a class (static) variable without causing an exception.

**Java Interview Question #32**

*(Question provided by*[*Suresh Atta*](https://www.codementor.io/sureshatta)*)*

**public** **class** **Test**

{

**public** **static** **void** **main**(String[] args)

{

Integer a = 1000, b = 1000;

System.out.println(a == b);

Integer c = 100, d = 100;

System.out.println(c == d);

}

}

outputs:

**false**

**true**

**Why is the code printing true in the second and false in the first case?**

JVM’s cache behavior can be confusing, so this question tests that concept. The second output is true as we are comparing the references, because the JVM tries to save memory when the Integer falls within a range (from -128 to 127). At point 2, no new reference of type Integer is created for ‘d’. Instead of creating a new object for the Integer type reference variable ‘d’, it is only assigned with a previously created object referenced by ‘c’. All of these are done by JVM.

**Java Interview Question #33**

**How do you check if the given 2 Strings below are Anagrams or Not?**  
*(Question provided by*[*Suresh Atta*](https://www.codementor.io/sureshatta)*)*

String s1="home";

String s2="mohe";

**boolean** result = **new** String(Arrays.sort(s1.toCharArray()))

.equals(**new** String(Arrays.sort(s2.toCharArray())));

**Java Interview Question #34**

**How do you reverse String("Java Programming") without using Iteration and Recursion?**  
*(Question provided by*[*Suresh Atta*](https://www.codementor.io/sureshatta)*)*

System.out.println("reverse = " + **new** StringBuilder(givenString).reverse());

**Java Interview Question #35**

**Give real world examples of when to use an ArrayList and when to use LinkedList.**  
*(Question provided by*[*Suresh Atta*](https://www.codementor.io/sureshatta)*)*

ArrayList is preferred when there are more get(int), or when search operations need to be performed as every search operation runtime is O(1).

If an application requires more insert(int) and delete(int) operations, then LinkedList is preferred, as LinkedList does not need to maintain back and forth to preserve continued indices as arraylist does. Overall this question tests the proper usage of collections.

**Java Interview Question #36**

**What is the difference between an Iterator and a ListIterator ?**  
*(Question provided by*[*Suresh Atta*](https://www.codementor.io/sureshatta)*)*

This question tests the proper usage of collection iterators. One can only use ListIterator to traverse Lists, and cannot traverse a Set using ListIterator.

What’s more, one can only traverse in a forward direction using Iterators. Using ListIterator, one can traverse a List in both the directions (forward and backward).

One cannot obtain indexes while using Iterator. Indexes can be obtained at any point of time while traversing a list using ListIterator. The methods nextIndex() and previousIndex() are used for this purpose.

**Java Interview Question #37**

**What is the advantage of generic collection?**  
*(Question provided by*[*Suresh Atta*](https://www.codementor.io/sureshatta)*)*

They enable stronger type checks at compile time.

A Java compiler applies strong type checking to generic code, and issues errors if the code violates type safety. Fixing compile-time errors is easier than fixing runtime errors, which can be difficult to find.



***Author Bio***

*Suresh Atta is Senior Web and Java Developer. He was also a*[*top 0.15% StackOverflow user*](http://stackoverflow.com/users/1927832/s%E1%B4%9C%CA%80%E1%B4%87s%CA%9C-%E1%B4%80%E1%B4%9B%E1%B4%9B%E1%B4%80)*for the fourth quarter of 2015.*[*Hire Suresh Now*](https://www.codementor.io/sureshatta/profile)*.*

**Conclusion**

Hopefully, you’ve found these interview questions useful when vetting Java developers. Keep in mind that the technical interview is just *one* portion of the hiring process. Whether you're hiring for freelance of full-time developers, you also want to evaluate their [soft skills](https://www.codementor.io/blog/10-freelance-engineer-interview-questions-that-will-make-or-break-your-project-262nursssi) like communication, problem solving, time management, and more.

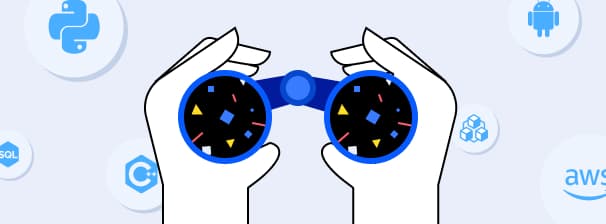
*We also wrote a list of*[*Android interview questions*](https://www.codementor.io/developers/android-app?view=guide)*and answers that you may find helpful.*

**💰Remote Java Developer Salaries**

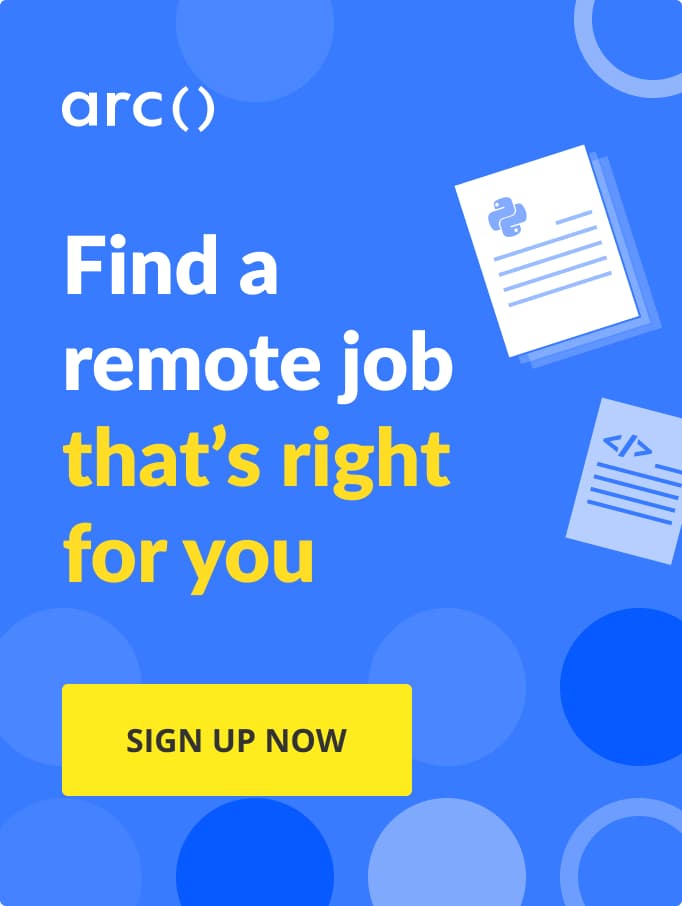
Average salary: **$72,809**

[See more remote salaries in your region](https://arc.dev/salaries/java-developers)

**🔎Remote Java Developer Hiring Guide**



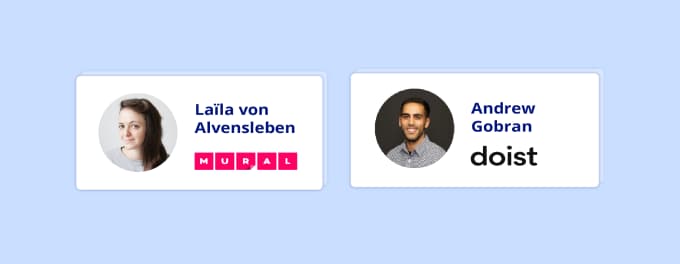
[See more](https://arc.dev/hire-developers/java#post)



**Remote Developer Interview Tips**

From preparing your portfolio, to thinking about soft skill answers in advance: here are some curated resources to help you nail your upcoming Java interview.

Learn from engineering leaders what they're looking for in a Java developer candidate, so you can stand out from the pack!



How to Get Hired for a Remote Job: Doist & MURAL Interview

[**Read More**](https://arc.dev/blog/get-hired-for-remote-job-doist-mural-9v1gkbk1ss)



Career Success Stories

[**Read More**](https://arc.dev/w/get-remote-job)



How to Land Remote Developer Jobs: Interview Tips from Shopify, Zapier & Help Scout

[**Read More**](https://arc.dev/blog/remote-developer-interview-tips-shopify-zapier-help-scout-9t1usiby5j)

**More Interview Questions**

**Front-end**

[Angular Interview Questions](https://arc.dev/interview/angular-interview-questions-and-answers)[React Interview Questions](https://arc.dev/interview/reactjs-interview-questions-and-answers)

**Backend**

[Java Interview Questions](https://arc.dev/interview/java-interview-questions-and-answers)[PHP Interview Questions](https://arc.dev/interview/php-interview-questions-and-answers)[Ruby on Rails Interview Questions](https://arc.dev/interview/ruby-on-rails-interview-questions-and-answers)

**Data**

[Data Engineer Interview Questions](https://arc.dev/interview/data-engineer-interview-questions-and-answers)

**Mobile App**

[Android Interview Questions](https://arc.dev/interview/android-interview-questions-and-answers)

**Others**

[DevOps Interview Questions](https://arc.dev/interview/devops-interview-questions-and-answers)[Engineering Manager Interview Questions](https://arc.dev/interview/engineering-manager-interview-questions-and-answers)

**Find a remote job that's right for you**

SIGN UP NOW

[I want to hire a developer](https://arc.dev/)

By using Arc, you agree to our [Cookie Policy](https://arc.dev/cookies).

ACCEPT

**PRODUCTS**

[**Arc**](https://arc.dev/)

The remote career platform for developers

[**Codementor**](https://www.codementor.io/)

Find a mentor to help you in real time

**LINKS**

[About us](https://arc.dev/about-us?developer=1)[Arc Careers - Hiring Now!](https://www.notion.so/codementor/Hi-we-re-Arc-and-Codementor-1d506da9d8d64e83b44eaf0aa4a79a0f)[Remote Junior Jobs](https://arc.dev/remote-jr-jobs)[Developer Salaries](https://arc.dev/salaries)[Career Success Stories](https://arc.dev/w/get-remote-job)[Virtual Events](https://arc.dev/virtual-events)

**JOBS BY EXPERTISE**

[Remote Front-end Developer Jobs](https://arc.dev/remote-jobs/front-end%20developer)[Remote Back-end Developer Jobs](https://arc.dev/remote-jobs/back-end%20developer)[Remote Full-stack Developer Jobs](https://arc.dev/remote-jobs/full-stack%20developer)[Remote Data Analyst Jobs](https://arc.dev/remote-jobs/data%20analyst)[Remote Data Science Jobs](https://arc.dev/remote-jobs/data%20science)[Remote Game Developer Jobs](https://arc.dev/remote-jobs/game%20developer)[Remote IT Jobs](https://arc.dev/remote-jobs/IT%20administrator)[Remote Software Engineer Jobs](https://arc.dev/remote-jobs/software%20engineer)[Remote Web Developer Jobs](https://arc.dev/remote-jobs/web%20developer)[Remote Mobile Developer Jobs](https://arc.dev/remote-jobs/mobile%20developer)

**JOBS BY TECH STACKS**

[Remote AWS Developer Jobs](https://arc.dev/remote-jobs/amazon-web-services)[Remote DevOps Engineer Jobs](https://arc.dev/remote-jobs/devops)[Remote Java Developer Jobs](https://arc.dev/remote-jobs/java)[Remote Javascript Developer Jobs](https://arc.dev/remote-jobs/javascript)[Remote Python Developer Jobs](https://arc.dev/remote-jobs/python)[Remote React Developer Jobs](https://arc.dev/remote-jobs/reactjs)[Remote Shopify Developer Jobs](https://arc.dev/remote-jobs/shopify)[Remote SQL Developer Jobs](https://arc.dev/remote-jobs/sql)[Remote Unity Developer Jobs](https://arc.dev/remote-jobs/unity)[Remote Wordpress Developer Jobs](https://arc.dev/remote-jobs/wordpress)

© Copyright 2021 Arc

[Cookie Policy](https://arc.dev/cookies)[Privacy Policy](https://arc.dev/privacy)[Terms of Service](https://arc.dev/terms)

By continuing to use this site you agree to our [Cookie Policy](https://www.toptal.com/cookie-policy).

Got it

[®](https://www.toptal.com/" \o "Toptal)

[Top 3%](https://www.toptal.com/top-3-percent)[Why](https://www.toptal.com/why)[Clients](https://www.toptal.com/clients)[Enterprise](https://www.toptal.com/enterprise)[Community](https://www.toptal.com/community)[Blog](https://www.toptal.com/blog)[About Us](https://www.toptal.com/about)

* [Apply as a Freelancer](https://www.toptal.com/talent/apply)
* [**Hire Top Talent**](https://www.toptal.com/hire)
* [Log In](https://www.toptal.com/users/login)

21 Essential Java Interview Questions [\*](https://www.toptal.com/java/interview-questions#note)

Looking for [Freelance Java Developer jobs](https://www.toptal.com/freelance-jobs/developers/java)? Design your lifestyle as a Java developer with Toptal.

[**Submit an Interview Question**](https://www.toptal.com/java/interview-questions#submit-question)

**Describe and compare fail-fast and fail-safe iterators. Give examples.**

Hide answer

The main distinction between **fail-fast** and **fail-safe** iterators is whether or not the collection can be modified *while* it is being iterated. Fail-safe iterators allow this; fail-fast iterators do not.

* **Fail-fast** iterators operate directly on the collection itself. During iteration, fail-fast iterators fail as soon as they realize that the collection has been modified (i.e., upon realizing that a member has been added, modified, or removed) and will throw a [ConcurrentModificationException](http://docs.oracle.com/javase/6/docs/api/java/util/ConcurrentModificationException.html). Some examples include [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html), [HashSet](http://docs.oracle.com/javase/7/docs/api/java/util/HashSet.html), and [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) (most JDK1.4 collections are implemented to be fail-fast).
* **Fail-safe** iterates operate on a *cloned copy* of the collection and therefore do *not* throw an exception if the collection is modified during iteration. Examples would include iterators returned by [ConcurrentHashMap](http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ConcurrentHashMap.html) or [CopyOnWriteArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/CopyOnWriteArrayList.html).

ArrayList**,**LinkedList**, and**Vector**are all implementations of the**List**interface. Which of them is most efficient for adding and removing elements from the list? Explain your answer, including any other alternatives you may be aware of.**

Hide answer

Of the three, [LinkedList](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html) is generally going to give you the best performance. Here’s why:

[ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html) and [Vector](http://docs.oracle.com/javase/7/docs/api/java/util/Vector.html) each use an array to store the elements of the list. As a result, when an element is inserted into (or removed from) the middle of the list, the elements that follow must all be shifted accordingly. Vector is synchronized, so if a thread-safe implementation is *not* needed, it is recommended to use ArrayList rather than Vector.

[LinkedList](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html), on the other hand, is implemented using a doubly linked list. As a result, an inserting or removing an element only requires updating the links that immediately precede and follow the element being inserted or removed.

However, it is worth noting that if performance is that critical, it’s better to just use an array and manage it yourself, or use one of the high performance 3rd party packages such as [Trove](http://trove.starlight-systems.com/) or [HPPC](http://labs.carrotsearch.com/hppc.html).

**Why would it be more secure to store sensitive data (such as a password, social security number, etc.) in a character array rather than in a String?**

Hide answer

In Java, Strings are [immutable](http://docs.oracle.com/javase/tutorial/essential/concurrency/immutable.html) and are stored in the String pool. What this means is that, once a String is created, it stays in the pool in memory until being garbage collected. Therefore, even after you’re done processing the string value (e.g., the password), it remains available in memory for an indeterminate period of time thereafter (again, until being garbage collected) which you have no real control over. Therefore, anyone having access to a memory dump can potentially extract the sensitive data and exploit it.

In contrast, if you use a mutable object like a character array, for example, to store the value, you can set it to blank once you are done with it with confidence that it will no longer be retained in memory.

**Apply to Join Toptal's Development Network**

and enjoy reliable, steady, remote Freelance Java Developer Jobs.

[**Apply as a Freelancer**](https://www.toptal.com/talent/apply)

**What is the**ThreadLocal**class? How and why would you use it?**

Hide answer

A single [ThreadLocal](http://docs.oracle.com/javase/7/docs/api/java/lang/ThreadLocal.html) instance can store different values for each thread independently. Each thread that accesses the get() or set() method of a ThreadLocal instance is accessing its own, independently initialized copy of the variable. ThreadLocal instances are typically private static fields in classes that wish to associate state with a thread (e.g., a user ID or transaction ID). The example below, from the [ThreadLocal Javadoc](http://docs.oracle.com/javase/7/docs/api/java/lang/ThreadLocal.html), generates unique identifiers local to each thread. A thread’s id is assigned the first time it invokes ThreadId.get() and remains unchanged on subsequent calls.

public class ThreadId {

// Next thread ID to be assigned

private static final AtomicInteger nextId = new AtomicInteger(0);

// Thread local variable containing each thread's ID

private static final ThreadLocal<Integer> threadId =

new ThreadLocal<Integer>() {

@Override protected Integer initialValue() {

return nextId.getAndIncrement();

}

};

// Returns the current thread's unique ID, assigning it if necessary

public static int get() {

return threadId.get();

}

}

Each thread holds an implicit reference to its copy of a thread-local variable as long as the thread is alive and the ThreadLocal instance is accessible; after a thread goes away, all of its copies of thread-local instances are subject to garbage collection (unless other references to these copies exist).

**What is the**volatile**keyword? How and why would you use it?**

Hide answer

In Java, each thread has its own stack, including its own copy of variables it can access. When the thread is created, it copies the value of all accessible variables into its own stack. The volatile keyword basically says to the JVM “Warning, this variable may be modified in another Thread”.

In all versions of Java, the volatile keyword guarantees global ordering on reads and writes to a variable. This implies that every thread accessing a volatile field will read the variable’s current value instead of (potentially) using a cached value.

In Java 5 or later, volatile reads and writes establish a [happens-before](http://docs.oracle.com/javase/specs/jls/se7/html/jls-17.html#jls-17.4.5) relationship, much like acquiring and releasing a mutex.

Using volatile may be faster than a lock, but it will not work in some situations. The range of situations in which volatile is effective was expanded in Java 5; in particular, [double-checked locking](http://www.javamex.com/tutorials/double_checked_locking_fixing.shtml) now works correctly.

The volatile keyword is also useful for 64-bit types like long and double since they are written in two operations. Without the volatile keyword you risk stale or invalid values.

One common example for using volatile is for a flag to terminate a thread. If you’ve started a thread, and you want to be able to safely interrupt it from a different thread, you can have the thread periodically check a flag (i.e., to stop it, set the flag to true). By making the flag volatile, you can ensure that the thread that is checking its value will see that it has been set to true without even having to use a synchronized block. For example:

public class Foo extends Thread {

private volatile boolean close = false;

public void run() {

while(!close) {

// do work

}

}

public void close() {

close = true;

// interrupt here if needed

}

}

**Compare the**sleep()**and**wait()**methods in Java, including when and why you would use one vs. the other.**

Hide answer

sleep() is a blocking operation that keeps a hold on the monitor / lock of the shared object for the specified number of milliseconds.

wait(), on the other hand, simply *pauses* the thread until *either* (a) the specified number of milliseconds have elapsed *or* (b) it receives a desired notification from another thread (whichever is first), *without* keeping a hold on the monitor/lock of the shared object.

sleep() is most commonly used for polling, or to check for certain results, at a regular interval. wait() is generally used in multithreaded applications, in conjunction with notify() / notifyAll(), to achieve synchronization and avoid race conditions.

**Tail recursion is functionally equivalent to iteration. Since Java does not yet support tail call optimization, describe how to transform a simple tail recursive function into a loop and why one is typically preferred over the other.**

Hide answer

Here is an example of a typical recursive function, computing the arithmetic series 1, 2, 3…N. Notice how the addition is performed after the function call. For each recursive step, we add another frame to the stack.

public int sumFromOneToN(int n) {

if (n < 1) {

return 0;

}

return n + sumFromOneToN(n - 1);

}

Tail recursion occurs when the recursive call is in the tail position within its enclosing context - after the function calls itself, it performs no additional work. That is, once the base case is complete, the solution is apparent. For example:

public int sumFromOneToN(int n, int a) {

if (n < 1) {

return a;

}

return sumFromOneToN(n - 1, a + n);

}

Here you can see that a plays the role of the accumulator - instead of computing the sum on the way down the stack, we compute it on the way up, effectively making the return trip unnecessary, since it stores no additional state and performs no further computation. Once we hit the base case, the work is done - below is that same function, “unrolled”.

public int sumFromOneToN(int n) {

int a = 0;

while(n > 0) {

a += n--;

}

return a;

}

Many functional languages natively support tail call optimization, however the JVM does not. In order to implement recursive functions in Java, we need to be aware of this limitation to avoid StackOverflowErrors. In Java, iteration is almost universally preferred to recursion.

**How can you swap the values of two numeric variables without using any other variables?**

Hide answer

You can swap two values a and b without using any other variables as follows:

a = a + b;

b = a - b;

a = a - b;

**How can you catch an exception thrown by another thread in Java?**

Hide answer

This can be done using [Thread.UncaughtExceptionHandler](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.UncaughtExceptionHandler.html).

Here’s a simple example:

// create our uncaught exception handler

Thread.UncaughtExceptionHandler handler = new Thread.UncaughtExceptionHandler() {

public void uncaughtException(Thread th, Throwable ex) {

System.out.println("Uncaught exception: " + ex);

}

};

// create another thread

Thread otherThread = new Thread() {

public void run() {

System.out.println("Sleeping ...");

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

System.out.println("Interrupted.");

}

System.out.println("Throwing exception ...");

throw new RuntimeException();

}

};

// set our uncaught exception handler as the one to be used when the new thread

// throws an uncaught exception

otherThread.setUncaughtExceptionHandler(handler);

// start the other thread - our uncaught exception handler will be invoked when

// the other thread throws an uncaught exception

otherThread.start();

**What is the Java Classloader? List and explain the purpose of the three types of class loaders.**

Hide answer

The [Java Classloader](https://docs.oracle.com/javase/7/docs/api/java/lang/ClassLoader.html) is the part of the Java runtime environment that loads classes on demand (lazy loading) into the JVM (Java Virtual Machine). Classes may be loaded from the local file system, a remote file system, or even the web.

When the JVM is started, three class loaders are used: 1. **Bootstrap Classloader:** Loads core java API file rt.jar from folder. 2. **Extension Classloader:** Loads jar files from folder. 3. **System/Application Classloader:** Loads jar files from path specified in the CLASSPATH environment variable.

**Is a**finally**block executed when an exception is thrown from a**try**block that does not have a**catch**block, and if so, when?**

Hide answer

A finally block is executed even if an exception is thrown or propagated to the calling code block.

Example:

public class FinallyExecution {

public static void main(String[] args) {

try{

FinallyExecution.divide(100, 0);}

finally{

System.out.println("finally in main");

}

}

public static void divide(int n, int div){

try{

int ans = n/div; }

finally{

System.out.println("finally of divide");

}

}

}

Output can vary, being either:

finally of divide

finally in main

Exception in thread "main" java.lang.ArithmeticException: / by zero

at exceptions.FinallyExecution.divide(FinallyExecution.java:20)

at exceptions.FinallyExecution.main(FinallyExecution.java:9)

…or…

Exception in thread "main" java.lang.ArithmeticException: / by zero

at exceptions.FinallyExecution.divide(FinallyExecution.java:20)

at exceptions.FinallyExecution.main(FinallyExecution.java:9)

finally of divide

finally in main

**When designing an abstract class, why should you avoid calling abstract methods inside its constructor?**

Hide answer

This is a problem of initialization order. The subclass constructor will not have had a chance to run yet and there is no way to force it to run it before the parent class. Consider the following example class:

public abstract class Widget {

private final int cachedWidth;

private final int cachedHeight;

public Widget() {

this.cachedWidth = width();

this.cachedHeight = height();

}

protected abstract int width();

protected abstract int height();

}

This seems like a good start for an abstract Widget: it allows subclasses to fill in width and height, and caches their initial values. However, look when you spec out a typical subclass implementation like so:

public class SquareWidget extends Widget {

private final int size;

public SquareWidget(int size) {

this.size = size;

}

@Override

protected int width() {

return size;

}

@Override

protected int height() {

return size;

}

}

Now we’ve introduced a subtle bug: Widget.cachedWidth and Widget.cachedHeight will always be zero for SquareWidget instances! This is because the this.size = size assignment occurs *after* the Widget constructor runs.

Avoid calling abstract methods in your abstract classes’ constructors, as it restricts how those abstract methods can be implemented.

**What variance is imposed on generic type parameters? How much control does Java give you over this?**

Hide answer

Java’s generic type parameters are *invariant*. This means for any distinct types A and B, G<A> is not a subtype or supertype of G<B>. As a real world example, List<String> is not a supertype or subtype of List<Object>. So even though String extends (i.e. is a subtype of) Object, both of the following assignments will fail to compile:

List<String> strings = Arrays.<Object>asList("hi there");

List<Object> objects = Arrays.<String>asList("hi there");

Java does give you some control over this in the form of *use-site variance*. On individual methods, we can use ? extends Type to create a *covariant* parameter. Here’s an example:

public double sum(List<? extends Number> numbers) {

double sum = 0;

for (Number number : numbers) {

sum += number.doubleValue();

}

return sum;

}

List<Long> longs = Arrays.asList(42L, 128L, -10L);

double sumOfLongs = sum(longs);

Even though longs is a List<Long> and not List<Number>, it can be passed to sum.

Similarly, ? super Type lets a method parameter be *contravariant*. Consider a function with a callback parameter:

public void forEachNumber(Callback<? super Number> callback) {

callback.call(50.0f);

callback.call(123123);

callback.call((short) 99);

}

forEachNumber allows Callback<Object> to be a subtype of Callback <Number>, which means any callback that handles a supertype of Number will do:

forEachNumber(new Callback<Object>() {

@Override public void call(Object value) {

System.out.println(value);

}

});

Note, however, that attempting to provide a callback that handles only Long (a subtype of Number) will rightly fail:

// fails to compile!

forEachNumber(new Callback<Long>() { ... });

Liberal application of use-site variance can prevent many of the unsafe casts that often appear in Java code and is crucial when designing interfaces used by multiple developers.

**What are static initializers and when would you use them?**

Hide answer

A static initializer gives you the opportunity to run code during the initial loading of a class and it guarantees that this code will only run once and will finish running before your class can be accessed in any way.

They are useful for performing initialization of complex static objects or to register a type with a static registry, as JDBC drivers do.

Suppose you want to create a static, immutable Map containing some feature flags. Java doesn’t have a good one-liner for initializing maps, so you can use static initializers instead:

public static final Map<String, Boolean> FEATURE\_FLAGS;

static {

Map<String, Boolean> flags = new HashMap<>();

flags.put("frustrate-users", false);

flags.put("reticulate-splines", true);

flags.put(...);

FEATURE\_FLAGS = Collections.unmodifiableMap(flags);

}

Within the same class, you can repeat this pattern of declaring a static field and immediately initializing it, since multiple static initializers are allowed.

**If one needs a**Set**, how do you choose between**HashSet**vs.**TreeSet**?**

Hide answer

At first glance, HashSet is superior in almost every way: O(1) add, remove and contains, vs. O(log(N)) for TreeSet.

However, TreeSet is indispensable when you wish to maintain order over the inserted elements or query for a range of elements within the set.

Consider a Set of timestamped Event objects. They could be stored in a HashSet, with equals and hashCode based on that timestamp. This is efficient storage and permits looking up events by a specific timestamp, but how would you get all events that happened on any given day? That would require a O(n) traversal of the HashSet, but it’s only a O(log(n)) operation with TreeSet using the tailSet method:

public class Event implements Comparable<Event> {

private final long timestamp;

public Event(long timestamp) {

this.timestamp = timestamp;

}

@Override public int compareTo(Event that) {

return Long.compare(this.timestamp, that.timestamp);

}

}

...

SortedSet<Event> events = new TreeSet<>();

events.addAll(...); // events come in

// all events that happened today

long midnightToday = ...;

events.tailSet(new Event(midnightToday));

If Event happens to be a class that we cannot extend or that doesn’t implement Comparable, TreeSet allows us to pass in our own Comparator:

SortedSet<Event> events = new TreeSet<>(

(left, right) -> Long.compare(left.timestamp, right.timestamp));

Generally speaking, TreeSet is a good choice when order matters and when reads are balanced against the increased cost of writes.

**What are method references, and how are they useful?**

Hide answer

Method references were introduced in Java 8 and allow constructors and methods (static or otherwise) to be used as lambdas. They allow one to discard the boilerplate of a lambda when the method reference matches an expected signature.

For example, suppose we have a service that must be stopped by a shutdown hook. Before Java 8, we would have code like this:

final SomeBusyService service = new SomeBusyService();

service.start();

onShutdown(new Runnable() {

@Override

public void run() {

service.stop();

}

});

With lambdas, this can be cut down considerably:

onShutdown(() -> service.stop());

However, stop matches the signature of Runnable.run (void return type, no parameters), and so we can introduce a method reference to the stop method of that specific SomeBusyService instance:

onShutdown(service::stop);

This is terse (as opposed to verbose code) and clearly communicates what is going on.

Method references don’t need to be tied to a specific instance, either; one can also use a method reference to an arbitrary object, which is useful in Stream operations. For example, suppose we have a Person class and want just the lowercase names of a collection of people:

List<Person> people = ...

List<String> names = people.stream()

.map(Person::getName)

.map(String::toLowerCase)

.collect(toList());

A complex lambda can also be pushed into a static or instance method and then used via a method reference instead. This makes the code more reusable and testable than if it were “trapped” in the lambda.

So we can see that method references are mainly used to improve code organization, clarity and terseness.

**How are Java enums more powerful than integer constants? How can this capability be used?**

Hide answer

Enums are essentially final classes with a fixed number of instances. They can implement interfaces but cannot extend another class.

This flexibility is useful in implementing the strategy pattern, for example, when the number of strategies is fixed. Consider an address book that records multiple methods of contact. We can represent these methods as an enum and attach fields, like the filename of the icon to display in the UI, and any corresponding behaviour, like how to initiate contact via that method:

public enum ContactMethod {

PHONE("telephone.png") {

@Override public void initiate(User user) {

Telephone.dial(user.getPhoneNumber());

}

},

EMAIL("envelope.png") {

@Override public void initiate(User user) {

EmailClient.sendTo(user.getEmailAddress());

}

},

SKYPE("skype.png") {

...

};

ContactMethod(String icon) {

this.icon = icon;

}

private final String icon;

public abstract void initiate(User user);

public String getIcon() {

return icon;

}

}

We can dispense with switch statements entirely by simply using instances of ContactMethod:

ContactMethod method = user.getPrimaryContactMethod();

displayIcon(method.getIcon());

method.initiate(user);

This is just the beginning of what can be done with enums. Generally, the safety and flexibility of enums means they should be used in place of integer constants, and switch statements can be eliminated with liberal use of abstract methods.

**What does it mean for a collection to be “backed by” another? Give an example of when this property is useful.**

Hide answer

If a collection backs another, it means that changes in one are reflected in the other and vice-versa.

For example, suppose we wanted to create a whitelist function that removes invalid keys from a Map. This is made far easier with Map.keySet, which returns a set of keys that is backed by the original map. When we remove keys from the key set, they are also removed from the backing map:

public static <K, V> Map<K, V> whitelist(Map<K, V> map, K... allowedKeys) {

Map<K, V> copy = new HashMap<>(map);

copy.keySet().retainAll(asList(allowedKeys));

return copy;

}

retainAll writes through to the backing map, and allows us to easily implement something that would otherwise require iterating over the entries in the input map, comparing them against allowedKey, etcetera.

Note, it is important to consult the documentation of the backing collection to see which modifications will successfully write through. In the example above, map.keySet().add(value) would fail, because we cannot add a key to the backing map without a value.

**What is reflection? Give an example of functionality that can only be implemented using reflection.**

Hide answer

Reflection allows programmatic access to information about a Java program’s types. Commonly used information includes: methods and fields available on a class, interfaces implemented by a class, and the runtime-retained annotations on classes, fields and methods.

Examples given are likely to include:

* Annotation-based serialization libraries often map class fields to JSON keys or XML elements (using annotations). These libraries need reflection to inspect those fields and their annotations and also to access the values during serialization.
* Model-View-Controller frameworks call controller methods based on routing rules. These frameworks must use reflection to find a method corresponding to an action name, check that its signature conforms to what the framework expects (e.g. takes a Request object, returns a Response), and finally, invoke the method.
* Dependency injection frameworks lean heavily on reflection. They use it to instantiate arbitrary beans for injection, check fields for annotations such as @Inject to discover if they require injection of a bean, and also to set those values.
* Object-relational mappers such as Hibernate use reflection to map database columns to fields or getter/setter pairs of a class, and can go as far as to infer table and column names by reading class and getter names, respectively.

A concrete code example could be something simple, like copying an object’s fields into a map:

Person person = new Person("Doug", "Sparling", 31);

Map<String, Object> values = new HashMap<>();

for (Field field : person.getClass().getDeclaredFields()) {

values.put(field.getName(), field.get(person));

}

// prints {firstName=Doug, lastName=Sparling, age=31}

System.out.println(values);

Such tricks can be useful for debugging, or for utility methods such as a toString method that works on any class.

Aside from implementing generic libraries, direct use of reflection is rare but it is still a handy tool to have. Knowledge of reflection is also useful for when these mechanisms fail.

However, it is often prudent to avoid reflection unless it is strictly necessary, as it can turn straightforward compiler errors into runtime errors.

**Nested classes can be static or non-static (also called an inner class). How do you decide which to use? Does it matter?**

Hide answer

The key difference between is that inner classes have full access to the fields and methods of the enclosing class. This can be convenient for event handlers, but comes at a cost: every instance of an inner class retains and requires a reference to its enclosing class.

With this cost in mind, there are many situations where we should prefer static nested classes. When instances of the nested class will outlive instances of the enclosing class, the nested class should be static to prevent memory leaks. Consider this implementation of the factory pattern:

public interface WidgetParser {

Widget parse(String str);

}

public class WidgetParserFactory {

public WidgetParserFactory(ParseConfig config) {

...

}

public WidgetParser create() {

new WidgetParserImpl(...);

}

private class WidgetParserImpl implements WidgetParser {

...

@Override public Widget parse(String str) {

...

}

}

}

At a glance, this design looks good: the WidgetParserFactory hides the implementation details of the parser with the nested class WidgetParserImpl. However, WidgetParserImpl is not static, and so if WidgetParserFactory is discarded immediately after the WidgetParser is created, the factory will leak, along with all the references it holds.

WidgetParserImpl should be made static, and if it needs access to any of WidgetParserFactory’s internals, they should be passed into WidgetParserImpl’s constructor instead. This also makes it easier to extract WidgetParserImpl into a separate class should it outgrow its enclosing class.

Inner classes are also harder to construct via reflection due to their “hidden” reference to the enclosing class, and this reference can get sucked in during reflection-based serialization, which is probably not intended.

So we can see that the decision of whether to make a nested class static is important, and that one should aim to make nested classes static in cases where instances will “escape” the enclosing class or if reflection on those nested classes is involved.

**What is the difference between**String s = "Test"**and**String s = new String("Test")**? Which is better and why?**

Hide answer

In general, String s = "Test" is more efficient to use than String s = new String("Test").

In the case of String s = "Test", a String with the value “Test” will be created in the String pool. If another String with the same value is then created (e.g., String s2 = "Test"), it will reference this same object in the String pool.

However, if you use String s = new String("Test"), in addition to creating a String with the value “Test” in the String pool, that String object will then be passed to the constructor of the String Object (i.e., new String("Test")) and will create another String object (not in the String pool) with that value. Each such call will therefore create an additional String object (e.g., String s2 = new String("Test") would create an addition String object, rather than just reusing the same String object from the String pool).

There is more to interviewing than tricky technical questions, so these are intended merely as a guide. Not every “A” candidate worth hiring will be able to answer them all, nor does answering them all guarantee an “A” candidate. At the end of the day, [hiring remains an art, a science — and a lot of work](https://www.toptal.com/freelance/in-search-of-the-elite-few-finding-and-hiring-the-best-developers-in-the-industry).

Submit an interview question

Submitted questions and answers are subject to review and editing, and may or may not be selected for posting, at the sole discretion of Toptal, LLC.

Top of Form

Name



Email



Enter Your Question Here …



Enter Your Answer Here …





I agree with the Terms and Conditions of Toptal, LLC's [Privacy Policy](https://www.toptal.com/privacy)

\* All fields are required

Submit a Question

Bottom of Form

**[View Full Profile](https://www.toptal.com/resume/kelly-ann-martines)**

[**View Kelly**](https://www.toptal.com/resume/kelly-ann-martines)

[**Kelly Ann Martines**](https://www.toptal.com/resume/kelly-ann-martines)

Freelance Java Developer

United StatesFreelance Java Developer at Toptal Since May 18, 2018

Kelly has 12 years of experience working as a software engineer, focusing primarily on C++, C, and Java. She's worked on desktop applications in addition to concurrent systems, including multithreaded systems, clusters, or networks of embedded systems in avionics.

[Show More](https://www.toptal.com/resume/kelly-ann-martines)

[Java](https://www.toptal.com/java)[C++](https://www.toptal.com/c-plus-plus)[Android](https://www.toptal.com/android)[C](https://www.toptal.com/c)[Node.js](https://www.toptal.com/nodejs)

**[View Full Profile](https://www.toptal.com/resume/radek-ostrowski)**

[**View Radek**](https://www.toptal.com/resume/radek-ostrowski)

[**Radek Ostrowski**](https://www.toptal.com/resume/radek-ostrowski)

Freelance Java Developer

AustraliaFreelance Java Developer at Toptal Since July 25, 2014

Radek is a certified Toptal blockchain engineer particularly interested in Ethereum and smart contracts. In the fiat world, he is experienced in big data/machine learning projects. He is a triple winner in two different international IBM Apache Spark competitions, co-creator of PlayStation 4's back end, a successful hackathon competitor, and speaker at conferences in Australia, Poland, and Serbia.

[Show More](https://www.toptal.com/resume/radek-ostrowski)

[Java](https://www.toptal.com/java)[Git](https://www.toptal.com/git)[Scala](https://www.toptal.com/scala)[Android](https://www.toptal.com/android)[Spring](https://www.toptal.com/spring)[Drupal](https://www.toptal.com/drupal)[Hadoop](https://www.toptal.com/hadoop)[Play Framework](https://www.toptal.com/play-framework)[Docker](https://www.toptal.com/docker)[Agile Software Development](https://www.toptal.com/agile)[Linux](https://www.toptal.com/linux)[NoSQL](https://www.toptal.com/nosql)[Apache Spark](https://www.toptal.com/big-data)

**[View Full Profile](https://www.toptal.com/resume/rizwan-rizvi)**

[**View Rizwan**](https://www.toptal.com/resume/rizwan-rizvi)

[**Rizwan Rizvi**](https://www.toptal.com/resume/rizwan-rizvi)

Freelance Java Developer

United StatesFreelance Java Developer at Toptal Since May 21, 2018

Rizwan has a reputation for overcoming complex challenges through clear thinking, innovative approaches, and enhancing the communication between different parts of organizations. Throughout his career, he has optimized the efforts of diverse and dispersed teams of IT professionals and consistently has delivered projects profitably in challenging environments.

[Show More](https://www.toptal.com/resume/rizwan-rizvi)

[Java](https://www.toptal.com/java)[C](https://www.toptal.com/c)[LINQ](https://www.toptal.com/linq)[Android](https://www.toptal.com/android)[C#](https://www.toptal.com/c-sharp)[C#.NET](https://www.toptal.com/dot-net)[Visual Basic 6 (VB6)](https://www.toptal.com/visual-basic)[Visual Basic .NET (VB.NET)](https://www.toptal.com/vb-net)[Visual Basic for Applications (VBA)](https://www.toptal.com/visual-basic)[Python](https://www.toptal.com/python)[HTML5](https://www.toptal.com/html5)[ASP.NET MVC](https://www.toptal.com/asp-dot-net-mvc)[AWS Lambda](https://www.toptal.com/aws-lambda)

Looking for Java Developers?

Looking for [Java Developers](https://www.toptal.com/java)? Check out Toptal’s Java developers.

Toptal Connects the [Top 3%](https://www.toptal.com/top-3-percent) of Freelance Talent All Over The World.

Join the Toptal community.

[Learn more](https://www.toptal.com/freelance-jobs/developers/java)

**Footer**

Skills in High Demand by Clients

* [React.js](https://www.toptal.com/freelance-jobs/developers/react)
* [Node.js](https://www.toptal.com/freelance-jobs/developers/nodejs)
* [Ruby on Rails](https://www.toptal.com/freelance-jobs/developers/ruby-on-rails)
* [Azure](https://www.toptal.com/freelance-jobs/developers/azure)
* [React Native](https://www.toptal.com/freelance-jobs/developers/react-native)
* [QA](https://www.toptal.com/freelance-jobs/developers/qa)

About

* [Top 3%](https://www.toptal.com/top-3-percent)
* [Clients](https://www.toptal.com/clients)
* [Toptal Talent Assessment](https://www.toptal.com/freelance-jobs/talent-assessment)
* [Ultimate Freelancing Guide](https://www.toptal.com/ultimate-freelancing-guide)
* [Blog](https://www.toptal.com/blog)
* [Community](https://www.toptal.com/community)
* [About Us](https://www.toptal.com/about)

Contact

* [Contact Us](https://www.toptal.com/contact)
* [Press Center](https://www.toptal.com/press-center)
* [Careers](https://www.toptal.com/careers)
* [FAQ](https://www.toptal.com/faq)

Social

[®](https://www.toptal.com/freelance-jobs" \o "Toptal Home Page)

The World’s Top Talent, On Demand ™

* Copyright 2010 - 2021 Toptal, LLC
* [Privacy Policy](https://www.toptal.com/privacy)
* [Website Terms](https://www.toptal.com/tos)

**[2021 OFFER :](https://www.janbasktraining.com/schedule-your-call-now/abc)**[Pay for 1 & Get 3 Months of Unlimited Class Access GRAB DEAL](https://www.janbasktraining.com/schedule-your-call-now/abc)

[Janbask Training](https://www.janbasktraining.com/)

* [**Select Course**](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/)
* [**Blog**](https://www.janbasktraining.com/blog)
* [**Corporate Training**](https://www.janbasktraining.com/corporate-training)
  + [Sign Up](https://www.janbasktraining.com/register)
  + [Login](https://www.janbasktraining.com/login)
* +91 7503384506

**- JAVA BLOGS -**

* [Home](https://www.janbasktraining.com/)

* [Blogs](https://www.janbasktraining.com/blog/)

* [Java](https://www.janbasktraining.com/blog/courses/java)

**Top 30 Core Java Interview Questions and Answers for Fresher, Experienced Developer**

* Jan 30, 2018

* 330.3k

* [Janbask Training](https://www.janbasktraining.com/blog/profile/admin)

Java is a programming language and also a platform for computing. Java was first released by Sun Microsystems in the year 1995. There are lots of applications and websites that will not work unless Java is installed in them. Each day there are more and more applications created using Java. It is a very simple, powerful and robust, programming language which provides a fast, secure, and reliable platform to work on. Java is being used everywhere, from laptops to data centers, game consoles to scientific supercomputers, and cell phones to the Internet. Due to its ease of use and simplicity, Java is used almost everywhere in the IT sector. This means that the professionals who are skilled in Java are always required. If you are someone who is likely to attend a job that would test you on your Java skills, then please go through the list of questions and their answers provided herein. You may also refer to them just to add on to your knowledge.

**Core Java Interview Questions and Answers cover:**

1. Core Java Interview Questions for Fresher
2. Core Java interview Questions for Experienced
3. Core Java Interview Questions for Senior Developer

**Core Java Interview Questions**

1. [Explain the difference between JDK, JRE, and JVM.](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#1)
2. [What makes Java a platform-independent programming language?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#2)
3. [Would it be correct to say that Java is not 100% Object-oriented?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#3)
4. [What is a constructor in Java?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#4)
5. [Differentiate between a constructor and a method? Can we mark constructors final?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#5)
6. [What is a class in Java?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#6)
7. [State the significance of public, private, protected class.](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#7)
8. [What is the final class?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#8)
9. [What is a wrapper class?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#9)
10. [What is an abstract class?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#10)
11. [Describe synchronization with respect to multi-threading.](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#11)
12. [What is the purpose of garbage collection in Java, and when is it used?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#12)
13. [What are the different ways of implementing thread? Which one is more advantageous?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#13)
14. [What if the main() method is declared as private? What happens when the static modifier is removed from the signature of the main() method?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#14)
15. [What is the first argument of the String array in main() method?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#15)
16. [What is a Java servlet?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#16)
17. [Describe the Life-cycle of a Java Servlet.](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#17)
18. [Explain the Request Dispatcher?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#18)
19. [List the steps to connect to a database in java?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#19)
20. [What do you mean by a JDBC Driver?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#20)
21. [What is a Java applet?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#21)
22. [What is a numeric promotion?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#22)
23. [What is false sharing in the context of multi-threading?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#23)
24. [What are the methods used to implement for the key Object in HashMap?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#24)
25. [What is an immutable object?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#25)
26. [Differentiate between String Buffer and String Builder in Java?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#26)
27. [What is the difference between factory and abstract factory pattern?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#27)
28. [Differentiate JAR and WAR files](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#28)
29. [What is a JIT compiler?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#29)
30. [What is the multi-catch block in Java?](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/#30)

**Java Interview Questions & Answers for Developer**

For a Java enthusiast who is looking to attend a Java interview anytime soon, here are the most popular interview questions and answers that might come your way. We have tried to include the top frequently asked questions as well as their answers to help freshers and the experienced professionals with their job interview.

**Q1). Explain the difference between JDK, JRE, and JVM.**

|  |  |  |
| --- | --- | --- |
| **JDK** | **JRE** | **JVM** |
| JDK stands for Java Development Kit and is the tool used to compile, package and document Java programs. | JRE stands for Java Runtime Environment and is a runtime environment which is a pre-requisite to run java byte-code. | JVM stands for Java Virtual Machine and is a specification that facilitates a run-time environment in which Java’s bytecode can be implemented. |

**Q2). What makes Java a platform-independent programming language?**

Java is platform independent because it provides the feasibility to run and compile the program in one platform and execute the program on any other platform of your choice.

**Q3). Would it be correct to say that Java is not 100% Object-oriented?**

Yes, it would be correct to say that Java is not 100% Object-oriented because it utilizes eight kinds of primitive data types and they are - Boolean, byte, char, int, float, double, long, short. These datatypes are not objects.

**Q4). What is a constructor in Java?**

A constructor is a block of code which is used in a Java environment to initialize an object. It carries the same name that you had given to the class. A Java constructor is automatically called when an object is created.

**Q5). Differentiate between a constructor and a method? Can we mark constructors final?**

A constructor constructs the value, by providing data for the object. It is a special type of method that is used to initialize the object. The constructor has the same name as the class itself, has no return type, and is invoked using the new operator. A method is an ordinary member function of a class. A method can be invoked using the dot operator and has its own name, and a return type. No, declaring the constructor as final is not possible.

**Q6). What is a class in Java?**

A class, in the environment of Java, are some templates which are used for object creation, and to define certain object datatypes and their methods.

**Q7). State the significance of public, private, protected class.**

* **Public**: When a class is public, the public class is visible in other packages, the field is visible everywhere.
* **Private**: Private variables or methods can be used by an instance of the same class only which declares the variable or method. A private feature can be accessed by the class that owns the feature.
* **Protected**: Protected variable is available to all classes in the same package. It is also available to all subclasses of the class that owns the protected feature. Subclasses that reside in a different package also is provided with the access from the class that owns the protected feature.

**Q8). What is the final class?**

A final class is a constant value of a final variable. Extending A final class is not possible ie., final class may not be sub-classed. A final method cannot be overridden when its class is inherited.

[**Read: What are the Benefits of loading jQuery from CDN**](https://www.janbasktraining.com/blog/basics-jquery-use-java-script-code/)

**Q9). What is a wrapper class?**

The classes that “wrap” the primitive data type into an object of that class are called Java wrapper classes.

**Core Java interview Questions for Experienced**

**Q10). What is an abstract class?**

Abstract serves as a template. In order to use an abstract class, it must be extended or subclassed. An abstract class may not be instantiated. Any class with an abstract method is abstract itself, hence is declared as such.

**Q11). Describe synchronization with respect to multi-threading.**

Synchronization is the method to control the access of multiple threads to shared resources, with respect to multi-threading. One thread can modify a shared variable when not in synchronization even when another thread is in the process of using or updating the same shared variable. This can lead to significant errors.

**Q12). What is the purpose of garbage collection in Java, and when is it used?**

The main purpose of garbage collection is to analyze a program, identify and discard the objects that are no longer needed. It is helpful as the resources can be reclaimed and reused. A Java object is subjected to garbage collection when the object is not reachable to the program in which it is used.

**Q13). What are the different ways of implementing thread? Which one is more advantageous?**

The thread can be implemented by Using runnable interface Inheriting from the Thread class. The use of the Runnable interface is more advantageous because when going for multiple inheritances, the only interface can help.

**Q14). What if the main() method is declared as private? What happens when the static modifier is removed from the signature of the main() method?**

When the main() method is declared as private, the program compiles but during runtime it shows “main() method not public.” Message. When the static modifier is removed from the signature of the main() method, the Program compiles but at runtime throws an error “NoSuchMethodError”.

**Q15). What is the first argument of the String array in main() method?**

Unlike in C/C++ where the first element by default is the program name, the string array in main() has no element, the String array is empty.

**Q16). What is a Java servlet?**

A Servlet in Java refers to the server-side technologies which are used to extend the competence of web servers by giving it support for a dynamic response as well as data persistence.

[**Read: What is A Java Constructor? Type of Constructors & Uses in Java**](https://www.janbasktraining.com/blog/what-is-a-java-constructor/)

**Q17). Describe the Life-cycle of a Java Servlet.**

Here is the life-cycle of a Java Servlet-

* Servlet is loaded
* Servlet is instantiated
* Servlet is destroyed
* Servlet is initialized
* Service the request

**Q18). Explain the Request Dispatcher?**

Request Dispatcher is a Java interface which is used to forward a request to some other resource which can be HTML, JSP or any another Java servlet within the same application.

**Q19). List the steps to connect to a database in java?**

Follow the given steps to connect to a database in java-

* Close the Connection
* Register the Drive Class
* Create Connection
* Create Statement
* Execute Queries

**Q20). What do you mean by a JDBC Driver?**

JDBC Driver in Java refers to a software component that allows Java application to relate with the database.There are 4 types of JDBC drivers in Java-

* JDBC-ODBC bridge driver
* Native-API driver (partially java driver)
* Network Protocol driver (fully java driver)
* Thin driver (fully java driver)

**Core Java Interview Questions for Senior Developer**

**Q21). What is a Java applet?**

The applet is a Java program which is designed for transmitting the Java code over the internet. The execution works automatically by Java-enabled Web Browser. The applet can respond to the user input immediately as it is dynamically programmed.

**Q22). What is a numeric promotion?**

Numeric promotions of a numeric operator are used for the conversion of the operands into a common type. In order to perform calculation easily, numeric promotion, conversion is performed. It is the conversion of a smaller numeric type to a larger numeric type so that integer and floating-point operations can be performed over it. Here byte, char, and short values are converted to int values. The int values are converted to long values, and the long and float values are converted to double values.

**Q23). What is false sharing in the context of multi-threading?**

On multi-core systems, false sharing is one of the well-known performance issues. Here each process has its local cache. When threads on a different processor, modify variables false sharing occurs, that resides on the same cache line as shown in the following imagesAs the thread may access different global variables completely, false sharing can be hard to detect.

**Q24). What are the methods used to implement for the key Object in HashMap?**

Equals and hashcode methods are to be implemented In order to use any object as Key in HashMap, in Java.

[**Read: How to Convert Char to Int in Java**](https://www.janbasktraining.com/blog/convert-char-to-int-in-java/)

**Q25). What is an immutable object?**

Java classes whose objects cannot be modified once they are created are known as Immutable classes. Any modification of Immutable object results formation of the new object.

**Q26). Differentiate between String Buffer and String Builder in Java?**

The only difference between StringBuffer and StringBuilder is that StringBuffer methods are synchronized while StringBuilder is not synchronized. StringBuilder in Java was introduced in Java 5.

**Q27). What is the difference between factory and abstract factory pattern?**

Abstract Factory provides one more level of abstraction. Consider different factories each extended from an Abstract Factory and responsible for the creation of different hierarchies of objects based on the type of factory. E.g. Abstract Factory extended by Automobile Factory, User Factory, Role Factory etc. Each individual factory would be responsible for the creation of objects in that genre.

**Q28). Differentiate JAR and WAR files**

We are going to share Difference between JAR & WAR File:

|  |  |
| --- | --- |
| **JAR FILES** | **WAR FILES** |
| Full form of JAR files is Java Archive Files. | Full form of WAR files is Web Archive Files. |
| Aggregating many files into one is allowed in JAR files | XML, Java classes, and JavaServer pages are stored in WAR |
| The JAR is usually used to hold Java classes in a library. | Mainly used for Web Application purposes.` |

**Q29). What is a JIT compiler?**

Just-In-Time(JIT) compiler is used to improve the performance. JIT compiles parts of the bytecode that has similar functionality which in turn reduces the amount of time needed for compilation. The term “compiler” here refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

**Q30). What is the multi-catch block in Java?**

Multi-catch block makes the code shorter and cleaner when every catch block has a similar code. We can catch multiple exceptions in a single catch block using this feature.

[Janbask Training](https://www.janbasktraining.com/blog/profile/admin)

A dynamic, highly professional, and a global online training course provider committed to propelling the next generation of technology learners with a whole new way of training experience.

**Comments**

Top of Form









Post Comment

Bottom of Form

* S

[Sanaya](javascript:;)

Thank you for taking the time and sharing this information with us. It was indeed very helpful and insightful while being straight forward and to the point.

[Reply](javascript:;)

* N

[Narendra Chinnu](javascript:;)

Thanks for your information, you have given very useful and important information. Good choice for learners.

[Reply](javascript:;)

**Trending Courses**

AWS

* AWS & Fundamentals of Linux
* Amazon Simple Storage Service
* Elastic Compute Cloud
* Databases Overview & Amazon Route 53

Upcoming Class

-0 day 14 May 2021

[View Details](https://www.janbasktraining.com/aws-training)

DevOps

* Intro to DevOps
* GIT and Maven
* Jenkins & Ansible
* Docker and Cloud Computing

Upcoming Class

7 days 21 May 2021

[View Details](https://www.janbasktraining.com/devops-certification-training)

Data Science

* Data Science Introduction
* Hadoop and Spark Overview
* Python & Intro to R Programming
* Machine Learning

Upcoming Class

-0 day 14 May 2021

[View Details](https://www.janbasktraining.com/data-science)

Hadoop

* Architecture, HDFS & MapReduce
* Unix Shell & Apache Pig Installation
* HIVE Installation & User-Defined Functions
* SQOOP & Hbase Installation

Upcoming Class

-0 day 14 May 2021

[View Details](https://www.janbasktraining.com/hadoop-big-data-analytics)

Salesforce

* Salesforce Configuration Introduction
* Security & Automation Process
* Sales & Service Cloud
* Apex Programming, SOQL & SOSL

Upcoming Class

1 day 15 May 2021

[View Details](https://www.janbasktraining.com/online-salesforce-training)

QA

* Introduction and Software Testing
* Software Test Life Cycle
* Automation Testing and API Testing
* Selenium framework development using Testing

Upcoming Class

-0 day 14 May 2021

[View Details](https://www.janbasktraining.com/online-qa-training)

Business Analyst

* BA & Stakeholders Overview
* BPMN, Requirement Elicitation
* BA Tools & Design Documents
* Enterprise Analysis, Agile & Scrum

Upcoming Class

15 days 29 May 2021

[View Details](https://www.janbasktraining.com/business-analyst-training)

MS SQL Server

* Introduction & Database Query
* Programming, Indexes & System Functions
* SSIS Package Development Procedures
* SSRS Report Design

Upcoming Class

-0 day 14 May 2021

[View Details](https://www.janbasktraining.com/online-sql-server-training)

Python

* Features of Python
* Python Editors and IDEs
* Data types and Variables
* Python File Operation

Upcoming Class

-0 day 14 May 2021

[View Details](https://www.janbasktraining.com/online-python-training)

Artificial Intelligence

* Components of AI
* Categories of Machine Learning
* Recurrent Neural Networks
* Recurrent Neural Networks

Upcoming Class

8 days 22 May 2021

[View Details](https://www.janbasktraining.com/ai-certification-training-online)

Machine Learning

* Introduction to Machine Learning & Python
* Machine Learning: Supervised Learning
* Machine Learning: Unsupervised Learning

Upcoming Class

21 days 04 Jun 2021

[View Details](https://www.janbasktraining.com/machine-learning)

Tableau

* Introduction to Tableau Desktop
* Data Transformation Methods
* Configuring tableau server
* Integration with R & Hadoop

Upcoming Class

1 day 15 May 2021

[View Details](https://www.janbasktraining.com/tableau-online-training)

**Browse Categories**

* [Salesforce](https://www.janbasktraining.com/blog/courses/salesforce)
* [Azure](https://www.janbasktraining.com/blog/courses/azure)
* [QA Testing](https://www.janbasktraining.com/blog/courses/qa)
* [SQL Server](https://www.janbasktraining.com/blog/courses/sql)
* [Business Analyst](https://www.janbasktraining.com/blog/courses/business-analyst)
* [Hadoop](https://www.janbasktraining.com/blog/courses/hadoop)
* [AWS](https://www.janbasktraining.com/blog/courses/aws)
* [DevOps](https://www.janbasktraining.com/blog/courses/devops)
* [Data Science](https://www.janbasktraining.com/blog/courses/data-science)
* [Java](https://www.janbasktraining.com/blog/courses/java)
* [Digital Marketing](https://www.janbasktraining.com/blog/courses/digital-marketing)
* [Dotnet](https://www.janbasktraining.com/blog/courses/dotnet)
* [PMP](https://www.janbasktraining.com/blog/courses/pmp)
* [Selenium](https://www.janbasktraining.com/blog/courses/selenium)
* [Worth To Visit](https://www.janbasktraining.com/blog/courses/worth-to-visit)
* [Machine Learning](https://www.janbasktraining.com/blog/courses/machine-learning)
* [Python](https://www.janbasktraining.com/blog/courses/python)
* [Oracle DBA](https://www.janbasktraining.com/blog/courses/oracle-dba)
* [Data Analyst](https://www.janbasktraining.com/blog/courses/data-analyst)
* [Tableau](https://www.janbasktraining.com/blog/courses/tableau)
* [Six Sigma](https://www.janbasktraining.com/blog/courses/six-sigma)
* [Scrum Master](https://www.janbasktraining.com/blog/courses/scrum-master)
* [Blockchain](https://www.janbasktraining.com/blog/courses/blockchain)
* [Artificial Intelligence](https://www.janbasktraining.com/blog/courses/artificial-intelligence)
* [Android](https://www.janbasktraining.com/blog/courses/android)
* [Cyber Security](https://www.janbasktraining.com/blog/courses/cyber-security)
* [VMware](https://www.janbasktraining.com/blog/courses/vmware)
* [Online IT Training](https://www.janbasktraining.com/blog/courses/online-it-training)

**Related Posts**

[Java String Functions & Methods with Examples](https://www.janbasktraining.com/blog/java-string-functions-methods-examples)

**Jan 12, 2018  785.4k**

[Read More](https://www.janbasktraining.com/blog/java-string-functions-methods-examples/)

[Java.net Connectexception connection refused](https://www.janbasktraining.com/blog/java-net-connectexception-connection-refused)

**Oct 30, 2017  520.7k**

[Read More](https://www.janbasktraining.com/blog/java-net-connectexception-connection-refused/)

[Java Developer Role & Responsibilities - Job Description](https://www.janbasktraining.com/blog/java-developer-role)

**Dec 28, 2017  992.1k**

[Read More](https://www.janbasktraining.com/blog/java-developer-role/)

**Search Posts**

Top of Form







Search [**Reset**](javascript:;)

Bottom of Form

**Trending Posts**

[What is SFDC? What does SFDC stand for?](https://www.janbasktraining.com/blog/what-is-sfdc/)   **2.2k**

[How to Add A New Column to a Table in SQL?](https://www.janbasktraining.com/blog/add-a-new-column-to-a-table-in-sql/)   **1.6k**

[What is the SQL Insert Query? How to Insert (Date, Multiple Rows, and Values in Table)](https://www.janbasktraining.com/blog/what-is-the-sql-insert-query/)   **1.5k**

[What Is The Difference Between Tables And Views In SQL?](https://www.janbasktraining.com/blog/tables-vs-views-in-sql/)   **1.4k**

[How to Call a Method in Java?](https://www.janbasktraining.com/blog/how-to-call-a-method-in-java/)   **1.3k**

**Related Posts**

[Java String Functions & Methods with Examples](https://www.janbasktraining.com/blog/java-string-functions-methods-examples/)   **785.4k**

[Java.net Connectexception connection refused](https://www.janbasktraining.com/blog/java-net-connectexception-connection-refused/)   **520.7k**

[Java Developer Role & Responsibilities - Job Description](https://www.janbasktraining.com/blog/java-developer-role/)   **992.1k**

[What is Reactjs? React JS Tutorial Guide for Beginners](https://www.janbasktraining.com/blog/reactjs-tutorial/)   **495.7k**

[What is A Java Constructor? Type of Constructors & Uses in Java](https://www.janbasktraining.com/blog/what-is-a-java-constructor/)   **452.9k**

**Java Course**  
Upcoming Batches

15

 May

**Mon - Fri**

6 Weeks

20

 May

**Mon - Fri**

6 Weeks

[**View Detail**](https://www.janbasktraining.com/online-java-training)

Receive Latest Materials and Offers on **Java Course**

Top of Form







Subscribe Now

Bottom of Form

**Online Course**

* [Salesforce](https://www.janbasktraining.com/online-salesforce-training)
* [SQL Server](https://www.janbasktraining.com/online-sql-server-training)
* [QA Testing](https://www.janbasktraining.com/online-qa-training)
* [DevOps](https://www.janbasktraining.com/devops-certification-training)
* [Java](https://www.janbasktraining.com/online-java-training)
* [.NET](https://www.janbasktraining.com/dotnet-training)
* [Python](https://www.janbasktraining.com/online-python-training)
* [Blockchain](https://www.janbasktraining.com/blockchain-certification-training)
* [Oracle DBA](https://www.janbasktraining.com/oracle-dba-training)
* [iOS Developer](https://www.janbasktraining.com/iphone-application-training)
* [Business Analyst](https://www.janbasktraining.com/business-analyst-training)
* [AWS](https://www.janbasktraining.com/aws-training)
* [Hadoop](https://www.janbasktraining.com/hadoop-big-data-analytics)
* [Data Science](https://www.janbasktraining.com/data-science)
* [Data Analytics](https://www.janbasktraining.com/data-analytics-training)
* [Digital Marketing](https://www.janbasktraining.com/digital-marketing-training)
* [Android Developer](https://www.janbasktraining.com/android-training)
* [VMWare](https://www.janbasktraining.com/vmware-training)
* [Tableau](https://www.janbasktraining.com/tableau-online-training)
* [Artificial Intelligence](https://www.janbasktraining.com/ai-certification-training-online)

**Tutorials**

[Salesforce Tutorial](https://www.janbasktraining.com/blog/salesforce-tutorial/)

[Software Testing Tutorial](https://www.janbasktraining.com/blog/software-testing-tutorial/)

[SQL Tutorial](https://www.janbasktraining.com/blog/sql-tutorial/)

[Business Analyst Tutorial](https://www.janbasktraining.com/blog/what-is-business-analyst/)

[Devops Tutorial](https://www.janbasktraining.com/blog/devops-tutorial/)

[Data Science Tutorial](https://www.janbasktraining.com/blog/data-science-tutorial/)

[AWS Tutorial](https://www.janbasktraining.com/blog/aws-tutorial/)

[Hadoop Tutorial](https://www.janbasktraining.com/blog/big-data-hadoop-tutorial-beginners/)

[SSRS Tutorial](https://www.janbasktraining.com/blog/ssrs-tutorial/)

[AWS S3 Tutorial](https://www.janbasktraining.com/blog/aws-s3-tutorial/)

[Functional Testing Tutorial](https://www.janbasktraining.com/blog/functional-testing-tutorial/)

[SSAS Tutorial](https://www.janbasktraining.com/blog/ssas-tutorial/)

[Automation Testing Tutorial](https://www.janbasktraining.com/blog/automation-testing-tutorial/)

[Manual Testing Tutorial](https://www.janbasktraining.com/blog/manual-testing-tutorial)

[Selenium Tutorial](https://www.janbasktraining.com/blog/selenium-testing-webdriver-tutorial-beginners/)

[Kubernetes Tutorial](https://www.janbasktraining.com/blog/kubernetes-tutorial/)

[Scala Tutorial](https://www.janbasktraining.com/blog/scala-tutorial/)

[ETL Testing Tutorial](https://www.janbasktraining.com/blog/etl-testing-tutorial/)

[Python Tutorial](https://www.janbasktraining.com/blog/python-programming-tutorial/)

[Pyspark Tutorial](https://www.janbasktraining.com/blog/pyspark-tutorial/)

[R Tutorial](https://www.janbasktraining.com/blog/r-for-data-science-tutorial/)

[Unit Testing Tutorial](https://www.janbasktraining.com/blog/unit-testing-tutorial/)

[API Testing Tutorial](https://www.janbasktraining.com/blog/what-is-api-testing/)

[Puppet Tutorial](https://www.janbasktraining.com/blog/what-is-puppet/)

[Integration Testing Tutorial](https://www.janbasktraining.com/blog/integration-testing-tutorial/)

[Chef Tutorial](https://www.janbasktraining.com/blog/chef-tutorial/)

[Jenkins Tutorial](https://www.janbasktraining.com/blog/jenkins-tutorial/)

[Ansible Tutorial](https://www.janbasktraining.com/blog/ansible-tutorial/)

[Vagrant Tutorial](https://www.janbasktraining.com/blog/vagrant-tutorial/)

[Docker Tutorial](https://www.janbasktraining.com/blog/what-is-docker/)

**Interviews**

* [Business Analyst Interview Questions](https://www.janbasktraining.com/blog/top-business-analyst-interview-questions-and-answers/)

* [DevOps Interview Questions](https://www.janbasktraining.com/blog/top-devops-interview-questions-and-answers/)

* [AWS Interview Questions](https://www.janbasktraining.com/blog/top-aws-interview-question-and-answers/)

* [QA Testing Interview Questions](https://www.janbasktraining.com/blog/top-20-qa-testing-interview-questions-and-answers/)

* [Software Testing Interview Questions](https://www.janbasktraining.com/blog/software-testing-interview/)

* [SQL Interview Questions](https://www.janbasktraining.com/blog/sql-interview-questions/)

* [Salesforce Interview Questions](https://www.janbasktraining.com/blog/top-salesforce-interview-questions-and-answers/)

* [Java Interview Questions](https://www.janbasktraining.com/blog/top-java-interview-questions-and-answers/)

* [Hibernate Interview Questions](https://www.janbasktraining.com/blog/hibernate-interview-questions-answers/)

* [Spark Interview Questions](https://www.janbasktraining.com/blog/apache-spark-interview-questions-answers/)

* [Vmware Interview Questions](https://www.janbasktraining.com/blog/important-vmware-interview-questions-and-answers/)

* [Data Science Interview Questions](https://www.janbasktraining.com/blog/top-10-mostly-asked-data-science-interview-questions-answers/)

* [Digital Marketing Interview Questions](https://www.janbasktraining.com/blog/top-digital-marketing-interview-questions-and-answers/)

* [API Testing Interview Questions](https://www.janbasktraining.com/blog/api-testing-interview-questions/)

* [SSAS Interview Questions](https://www.janbasktraining.com/blog/ssas-interview-questions/)

* [Power BI Interview Questions](https://www.janbasktraining.com/blog/power-bi-interview-questions/)

* [Cloud Computing Interview Questions](https://www.janbasktraining.com/blog/cloud-computing-interview-questions-answers/)

* [SSRS Interview Questions](https://www.janbasktraining.com/blog/ssrs-interview-questions/)

* [Manual Testing Interview Questions](https://www.janbasktraining.com/blog/manual-testing-interview-questions-answers/)

* [Social Media Interview Questions](https://www.janbasktraining.com/blog/social-media-marketing-interview-questions/)

* [Performance Testing Interview Questions](https://www.janbasktraining.com/blog/performance-testing-interview-questions/)

* [MSBI Interview Questions](https://www.janbasktraining.com/blog/msbi-interview-questions/)

* [QTP Interview Questions](https://www.janbasktraining.com/blog/qtp-interview-questions/)

* [Automation Testing Interview Questions](https://www.janbasktraining.com/blog/automation-testing-interview-questions-answers/)

* [SSIS Interview Questions](https://www.janbasktraining.com/blog/ssis-interview-questions/)

* [GIT Interview Questions](https://www.janbasktraining.com/blog/git-interview-questions/)

**Follow Us**

**Janbask Training**

* [Trending Courses](https://www.janbasktraining.com/courses)
* [Blog](https://www.janbasktraining.com/blog)
* [Contact Us](https://www.janbasktraining.com/contact-us)
* [About Us](https://www.janbasktraining.com/about-us)

**Join Us**

* [Corporate Training](https://www.janbasktraining.com/corporate-training)
* [Become a Learner](https://www.janbasktraining.com/login)
* [Become an Instructor](https://www.janbasktraining.com/insructor-login)

**Useful Links**

* [Term of Use](https://www.janbasktraining.com/term-of-use)
* [Privacy Policy](https://www.janbasktraining.com/privacy-policy)
* [Disclaimer](https://www.janbasktraining.com/disclaimer)
* [Trainer Login](https://www.janbasktraining.com/trainer)

© 2019 Copyright - Janbasktraining | All Rights Reserved

**[Be a Java Expert in 6 weeks](https://www.janbasktraining.com/online-java-training?utm_source=Blog&utm_medium=Java&utm_campaign=footer_CTA)**[Enroll Now](https://www.janbasktraining.com/online-java-training?utm_source=Blog&utm_medium=Java&utm_campaign=footer_CTA)

[Scroll](https://www.janbasktraining.com/blog/core-java-interview-questions-answers/)

**[Online Event] High Availability: How Klarna Maintains 99.999% Uptime** Join us for an hour of open discussion with Mite Mitreski, Engineering Director at Klarna.

[**REGISTER**](https://adevait.com/events/high-availability-how-klarna-maintains-99-uptime)

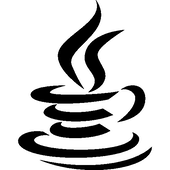
* [**Hiring for?**](https://adevait.com/java/interview-questions)

* [**Clients**](https://adevait.com/clients)

* [**Community**](https://adevait.com/community)

* [**Resources**](https://adevait.com/java/interview-questions)

[**START HIRING**](https://adevait.com/signup)



**29 Essential Java** Interview Questions

An updated list of Java interview questions that will help companies hire top software developers.

[**SUBMIT AN INTERVIEW QUESTION**](https://adevait.com/java/interview-questions)[Back to Interview Questions](https://adevait.com/interview-questions)

1

**Name a few types of Collections in Java.**

Hide the answer

Some of the most popular types of collections are: List, Set, Queue.

2

**What is the difference between fail-safe and fail-fast iterators?**

Hide the answer

* Fail-fast iterators, as the name suggests, fail fast (throw an exception) when some change is done to the underlying collection that is currently iterated. If some structural change like add, remove or replace element is done to the collection, then a **ConcurrentModificationException** is thrown. All the usually used collections have fail-fast iterators.
* Fail-safe iterators on the other hand don't throw an exception if the collection that is currently iterated is modified. The way fail-safe iterators achieve this is that they work on a clone of the collection, and not the actual collection. The 'concurrent' collections (in the **java.util.concurrent** package) have fail-safe iterators.

3

**Which Java type would you use if you want to store something per Thread?**

Hide the answer

To store something per Thread the **ThreadLocal** class is used. The **ThreadLocal** instance is usually a **private static** field that allows us to store any object inside it, for the current thread.  
Do you need to clean it up?  
In cases where some thread pool is used, the **ThreadLocal** must be cleaned at the end of the operation the Thread is executing because the same Thread might be reused later.

4

**What is a TreeSet and when would you use it?**

Hide the answer

The **TreeSet** is a Set that maintains the natural ordering of the elements (if the elements are 'Comparable' or if a 'Comparator' is provided to the **TreeSet**). And also the basic operations (add, remove, contains) have a **log(n)** time cost. Usually used when we want to use a **Set**, but need to keep the elements ordered for later iteration or other usage.

5

**What is the output of the following code snippet, and why?**

**System.out.println("asd"=="asd");  
System.out.println(new String("asd")==new String("asd"));  
System.out.println(new String("asd").equals(new String("asd")));**

Hide the answer

The output is:

**true**,  
**false**,  
**true**

The reason for this result is the **String constant pool** that Java maintains. In the first case each string is treated as a **constant** and stored at the **String constant pool**. When defining a string like this, the JVM first tries to find an item with the same value in the pool, so that's why the 2 references are the same (the constants are reused from the pool).

In the second case a new instance is created for each string, so they are different by reference.

In the third case, the 2 strings are compared by value (not reference) and by value they are equal.

6

**Each of the two examples contains 2 cases of string concatenation, which way of string concatenation is better in each example, and why?**

Example 1

**/\* case 1.1 \*/ String a = s + s + s;  
/\* case 1.2 \*/ String a1 = new StringBuilder(s).append(s).append(s).toString();**

Example 2

**String b = "";  
StringBuilder b1 = new StringBuilder();  
for(int i = 0; i < 20; i++) {**

**/\* case 2.1 \*/ b += s;**

**/\* case 2.2 \*/ b1.append(s);**

**}  
b1.toString();**

Hide the answer

In the first example, the string concatenation will be the same. The Strings in Java are immutable, meaning that the naive string concatenation with **+** in case 1.1 will be far from optimal because for each concatenation a new string is created.

So, in that case the Java compiler does some optimization at compile time, changing the **+** chaining to a **StringBuilder** **append**. At the end, the compiled code would be completely the same.

In the second example, the data is more dynamic, in a **for** loop, so the Java compiler is not able to optimize it by changing the **+** to a **StringBuilding**.

Hence, for case 2.1 a new String is created on each **+=**, meaning that the **StringBuilder** **.append** case 2.2 will be much faster.

7

**What does the synchronized keyword mean and where would you use it?**

Hide the answer

The **synchronized** keyword in Java defines a thread-safe operation and can be used on: instance methods, static methods and code blocks. The synchronization (thread-safety) is achieved by allowing a thread to acquire and release an intrinsic lock. While the thread owns the intrinsic lock, the other threads that will try to acquire it, will be blocked.

8

**What's the difference between StringBuilder and StringBuffer?**

Hide the answer

**StringBuffer** is a synchronized type, while the **StringBuilder** is not. The **StringBuilder** was created as a replacement for the **StringBuffer** used in a single thread, making the **StringBuilder** usually faster.

9

**What's the synchronized version of a HashMap?**

Hide the answer

The synchronized version of the **HashMap** is the **HashTable**.

10

**What's the difference between a HashSet and a LinkedHashSet?**

Hide the answer

They are both **Sets**, but the **LinkedHashSet** maintains a doubly-linked list of the items, preserving the insertion order.

**[BUSINESS INSIGHTS](https://adevait.com/guides/distributed-teams)**

**[Setting up and Scaling Remote and Distributed Teams](https://adevait.com/guides/distributed-teams)**

**FEATURED GUIDE**

**Setting up and Scaling Remote and Distributed Teams**

[**LEARN MORE**](https://adevait.com/guides/distributed-teams)

11

**What is a deadlock?**

Hide the answer

Deadlock is a situation where two or more threads are waiting for each other, thus, becoming blocked forever.

12

**What are generics in Java?**

Hide the answer

Generics are a mechanism that enables types to be parameters of the class, interface or method definition. This mechanism allows reusing of the same code with different inputs, while providing compile-time checks.

13

**What is the problem with the following example and does it compile?**

**class Animal<T> {**

**T animal;**

**Integer age;**

**public Animal(T animal, Integer age) {**

**this.animal = animal;**

**this.age = age;**

**}  
}  
new Animal(123, 123);**

Hide the answer

The problem is that the constructor is used without the type defined in the class definition. This leads to some warnings, but the code will still compile.

14

**Why are exceptions expensive in Java?**

Hide the answer

The most expensive part of the exceptions is the stacktrace that is created when the **Exception** is constructed.

15

**What are default methods?**

Hide the answer

Default methods are methods defined on an interface. They were introduced in Java 1.8. Before Java 1.8, interfaces could only have abstract methods, meaning that interfaces only provided the contract (the signature of the method) and the actual implementation (functionality) had to be provided in a class implementing that interface. With the default methods, the functionality (implementation) can be provided on interface level.

16

**What types of Exceptions exist in Java (and what is the difference)? Name few common Exceptions of each type.**

Hide the answer

In Java there are 2 type of Exceptions, checked and unchecked.

The checked exceptions are checked at compile and they must be either handled in a **catch** block, or they must be specified in the **throws** section of the method. The checked exceptions are subclasses of the **Exceptions** class and some of the most common exceptions of this type are: **IOException**, **SQLException**, **ClassNotFoundException** etc.

The unchecked Exceptions are not checked at compile time and don't have to be handled in a **catch** block or declared in the **throws** section of the method. The unchecked Exceptions are subclasses of the **RuntimeException** and some of the most common exceptions of this type are: **IllegalArgumentException**, **NullPointerException**, **IllegalStateException**.

17

**Given the following example code, which overrides are valid and which one aren't (with explanation)?**

**private abstract static class C1 {**

**public abstract void m1() throws Exception;**

**public abstract void m2() throws IOException;**

**public abstract void m3() throws Exception;**

**}  
private static class C2 extends C1 {**

**@Override**

**public void m1() throws IOException {}**

**@Override**

**public void m2() throws Exception {}**

**@Override**

**public void m3() {}**

**}**

Hide the answer

Overrides of **m1()** and **m3()** are valid. In the case of **m1()**, the override in C2 defines an **Exception** which is a sub-class of the one declared on the parent. This is defining a more precise exception than the parent, and it's completely legal. In the second case, **m2()** throws a more general exception, and this is an illegal override.

18

**What is the difference between an Application server and servlet container?**

Hide the answer

The application server provides the whole JavaEE support, while the servlet container provides only the Servlet specification. The servlet container (most commonly used implementation is Tomcat) can also run some of the JavaEE technologies if they are installed separately or with some framework providing some of them.

19

**What is JDBC?**

Hide the answer

JDBC stands for Java Database Connectivity and represents an API in Java for accessing SQL-based database systems. To make the JDBC API work with a specific database management system, a JDBC driver must be present on the client side. The driver acts as an adapter that converts the Java instructions to instructions that the DBMS understands.

20

**What is JPA and how is it different from JDBC?**

Hide the answer

JPA stands for Java Persistence API and represents an API for managing relational data in the application by mapping operations between Java objects and some relational database. The difference from JDBC is that JDBC defines how the client connects to the DBMS, while JPA defines how the data is managed.

21

**What are static methods and fields? What is wrong with using a static method in a class like new Class().someStaticMethod()?**

Hide the answer

Static methods and fields are class members. This means that they belong to the class, and not to an instance of the class. So, an instance of the class is not needed to gain access to these static members. The static fields are usually used for constants and the static methods for utility classes.

22

**Why is Java said to be 'write once, run everywhere'?**

Hide the answer

Java is said to be 'write once, run everywhere' thanks to the bytecode which acts as the intermediate step between the source code and the machine code. This allows the Java Bytecode to be executed on any hardware that has a proper JVM installed.

23

**What is the parent class that every other class extends from?**

Hide the answer

The parent class of every other class is the **Object.class**.

24

**Primitive vs Object types?**

Hide the answer

* Primitive types are the most basic types that Java supports. They are predefined as reserved keywords by the language and use a small amount of memory to represent a unit of data. Some primitive types are **int**, **long**, **byte**, **double** etc.
* The other **Object** types are types that have classes (which extend **Object.class**) and can be instantiated with the **new** keyword (if there's an appropriate constructor). The object types are usually complex types, a composition of properties and methods that take up a bigger chunk of the memory.

25

**What is Auto-boxing and Unboxing, and does it impact performance?**

Hide the answer

Autoboxing is the process of automatic conversion (by the Java compiler) from the primitive type to the corresponding object type (wrapper class of the primitive type). Unboxing is the reverse process, the conversion from the object type to the primitive type. Some primitive-object type pairs are **(int, Integer)**, **(long, Long)**, **(double, Double)** etc.

Since there's additional work required to execute them, both autoboxing and unboxing have an impact on the performance, so the developer should be careful and identify the points where autoboxing/unboxig happens to take care of some optimizations if needed.

26

**Identify the Autoboxing and the Unboxing in the following example:**

**List<Long> list = new ArrayList<>();  
list.add(1L);  
long l = list.get(0);**

Hide the answer

**list.add(1L)** causes autoboxing

**Long l = list.get(0)** causes unboxing

27

**Where would you use LinkedList and where an ArrayList?**

Hide the answer

* **LinkedList** is used in cases where we have a lot of add, remove and sequential iteration but not a lot of random access ( **list.get(n)** )
* **ArrayList** is used when there is a lot of random access used ( **list.get(n)** ), but not a lot of add, remove.

28

**What does the following code usually output? And why usually?**

**System.out.println(3 == 3);  
 System.out.println(15000 == 15000);  
 System.out.println((Integer)127 == (Integer)127);  
 System.out.println((Integer)128 == (Integer)128);**

Hide the answer

Output:

**true**,

**true**,

**true**,

**false**

With casting the values (**Integer**), a boxing of the primitive values to the object wrapper is done. Since this boxing has some performance impact, the JVM keeps a cache of the object wrappers in some range (for **Integer** the default range is [-127, 127]). This means that in the example **(Integer)127 == (Integer)127** , the wrapper object for 127 is taken from the cache and that's why they are same by reference. In the example **(Integer)128 == (Integer)128** , the value is false because the wrapper object is not cached (it is out of the default range), and the result is false because they are different by reference.

This is the usual case because the range is a JVM property that can be changed.

29

**What are Reflections?**

Hide the answer

Reflections are a mechanism in Java allowing to inspect or modify the application code at runtime. One of the most common usage of the reflection API is retrieving information for a certain class at runtime. Some runtime information that can be retrieved for a class is: class name, fields, methods, super classes, implemented interfaces etc. The usage of the Reflection API adds some performance overhead, so the usage of it should be limited, and if possible, avoided.

**Join 5K+ tech leaders**

Stay up to date with Software Engineering, Distributed Teams, Agile Talent, and Future of Work content

Top of Form



Bottom of Form

**Submit an interview question**

Submitted questions and answers are subject to review and editing, and may or may not be selected for posting, at the sole discretion of Adeva

Top of Form

Name \*

E-Mail \*

Your Question \*

Your Answer

SUBMIT A QUESTION

Bottom of Form

**Ready to join?**

Get in touch or schedule a call.

[**START HIRING**](https://adevait.com/signup)

**RESOURCES**

* [Guides](https://adevait.com/guides)
* [Interview Questions](https://adevait.com/interview-questions)

**INSIGHTS**

* [Agile Talent](https://adevait.com/blog/agile-work)
* [Distributed Teams](https://adevait.com/blog/remote-work)
* [Future of Work](https://adevait.com/blog/future-of-work)
* [Startups](https://adevait.com/blog/startups)

**COMMUNITY**

* [About](https://adevait.com/community)
* [Careers](https://adevait.com/careers/job-openings)
* [Partners](https://adevait.com/partners)
* [Contact](https://adevait.com/contact)
* [FAQs](https://adevait.com/faq)
* [Media Kit](https://adevait.com/media-kit)

**SOLUTIONS**

* [For Startups](https://adevait.com/startups)
* [For Enterprises](https://adevait.com/enterprise)
* © Copyright 2015 - 2021 Adeva
* [Privacy Policy](https://adevait.com/privacy)
* [Website Terms](https://adevait.com/tos)

[Skip to content](https://github.com/svozniuk/java-interviews/blob/master/Java%20Core/Java%20Basics.md#start-of-content)

* Why GitHub?
* [Team](https://github.com/team)
* [Enterprise](https://github.com/enterprise)
* Explore
* [Marketplace](https://github.com/marketplace)
* Pricing

Top of Form



Bottom of Form

[Sign in](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews%2Fblob%2Fmaster%2FJava%2520Core%2FJava%2520Basics.md)[Sign up](https://github.com/join?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F%3Cuser-name%3E%2F%3Crepo-name%3E%2Fblob%2Fshow&source=header-repo&source_repo=svozniuk%2Fjava-interviews)

[**svozniuk**](https://github.com/svozniuk)**/**[**java-interviews**](https://github.com/svozniuk/java-interviews)

* [Notifications](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews)
* [Star](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews)[**251**](https://github.com/svozniuk/java-interviews/stargazers)
* [Fork](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews)[**128**](https://github.com/svozniuk/java-interviews/network/members)
* [**Code**](https://github.com/svozniuk/java-interviews)
* [Issues3](https://github.com/svozniuk/java-interviews/issues)
* [Pull requests2](https://github.com/svozniuk/java-interviews/pulls)
* [Actions](https://github.com/svozniuk/java-interviews/actions)
* [Projects](https://github.com/svozniuk/java-interviews/projects)
* [Wiki](https://github.com/svozniuk/java-interviews/wiki)
* [Security](https://github.com/svozniuk/java-interviews/security)
* [Insights](https://github.com/svozniuk/java-interviews/pulse)

 master

[**java-interviews**](https://github.com/svozniuk/java-interviews)**/**[**Java Core**](https://github.com/svozniuk/java-interviews/tree/master/Java%20Core)**/Java Basics.md**

[Go to file](https://github.com/svozniuk/java-interviews/find/master)

Daniel Rauf [Improve formatting](https://github.com/svozniuk/java-interviews/commit/e4940d543a28858d41f99849d33e694a6cb7b680)

Latest commit [e4940d5](https://github.com/svozniuk/java-interviews/commit/e4940d543a28858d41f99849d33e694a6cb7b680) on Jun 9, 2016[**History**](https://github.com/svozniuk/java-interviews/commits/master/Java%20Core/Java%20Basics.md)

**1** contributor

465 lines (276 sloc)  23.6 KB

[Raw](https://github.com/svozniuk/java-interviews/raw/master/Java%20Core/Java%20Basics.md)[Blame](https://github.com/svozniuk/java-interviews/blame/master/Java%20Core/Java%20Basics.md)

**Java Basics**

1. **What is the difference between an Interface and an Abstract class?**

Both define an interface that has to be implemented. Abstract class can contain concrete methods as well as abstract. Abstract class can contain regular class fields. Interface can contain only public static final fields.

1. **What is the purpose of garbage collection in Java, and when is it used?**

GC in Java is the mechanism that keeps track of memory and objects residing in memory. GC collects the object when it is no longer needed (usually when no references to the object are available).

1. **What is the difference between a constructor and a method?**

We can say that constructor is a special kind of method that instantiates the object. The main differences are

* + Special treatment of constructors when they are called (memory allocation, superclass constructors chain)
  + More caution is required when writing a constructor

1. **State the significance of public, private, protected, default modifiers both singly and in combination and state the effect of package relationships on declared items qualified by these modifiers.**
   * public  
     class -- accessible from anywhere. Can be subclassed by anyone (if not declared final)  
     method -- accessible from anywhere. Can be overriden in subclasses.  
     variable -- accessible from anywhere. Usually not a good practice except for the constants  
     inner class -- accessible from anywhere.  
     nested class -- accessible from anywhere.
   * private  
     class -- only for inner classes.  
     method -- accessible only from the class where it is declared. Cannot be overridden  
     variable -- accessible only from the class where it is declared  
     inner class -- same  
     nested class -- same
   * protected  
     class -- only for inner classes  
     method -- accessible from the same package or from any subclass  
     variable -- same  
     inner class -- same  
     nested class -- can be written but doesn't make sense (protected static!?)
   * default (no access modifier)  
     class -- only this package and subclasses in this package  
     method -- same  
     variable -- same  
     inner class -- same  
     nested class -- same
2. **What is an abstract class?**

An abstract class is a java class that has one or more abstract methods (no body). Abstract classes cannot be instantiated. Abstract class defines an interface that has to be implemented by all its subclasses.

1. **What is static in java?**

static is Java Language keyword.  
a) When used with a method defines a method of a class.  
b) When used with a field defines a class field.  
c) When used on an nested class declaration defines a static nested class.  
d) Also can be used for static initialization block.  
e) Can be used as a static initialization block

1. **What is final?**

final is Java Language keyword.  
a) When used with a method protects it from being overridden in subclasses. Done for security and/or performance reasons.  
b) When used with a field means that the value stored in the field cannon be changed after initialization. Not to be confused with immutability of the object.  
c) When used with a class declaration protects it from being subclassed. Done for security and/or performance reasons. Also for immutability. Many of Java core classes are final (e.g. String)

1. **How can one prove that the array is not null but empty using one line of code?**

array == null ? xx : array.length == 0 if the array is instance field it's initialized to null. And this code works. if the array is local variable the compiler will generate an error if it has not been initialized

1. **What environment variables do I need to set on my machine in order to be able to run Java programs?**

CLASSPATH, PATH and/or JAVA\_HOME

1. **Do I need to import java.lang package any time? Why ?**

No. It is imported by default

1. **Can I import same package/class twice? Will the JVM load the package twice at runtime?**

Yes you can declare import twice in the import section. No it will not be loaded twice at runtime.

1. **What is Overriding?**

Changing method behavior in the subclasses.

1. **What are different types of inner classes?**

if declared with static -- it's nested class. Nested classes are fairly independent and treated as top-level classes. But the constructor call is different. To construct one: new OuterClassNeme.InnerClassName().

if declared without static -- inner class. Each instance of inner class can only exist within an instance of the outer class. To construct one: instanceOfOuterClass.new InnerClassName()

local class -- declared and visible only within a block of code anonymous -- same, but they don't even have a name.

1. **Are the imports checked for validity at compile time? e.g. will the code containing an import such as java.lang.ABCD compile?**

Yes, they are checked. No it won't compile.

1. *Does importing a package imports the sub-packages as well? e.g. Does importing com.MyTest. also*\* import com.MyTest.UnitTests.\*?

No. This declaration will only import classes located directly in com.MyTest package.

1. **What is the difference between declaring a variable and defining a variable?**

Declaring is declaring. Defining is assigning some value to the declared vaiable.

1. **What is the default value of an object reference declared as an instance variable?**

null

1. **Can a top level class be private or protected?**

No. It is only allowed to be public or to have default access modifier.

1. **What type of parameter passing does Java support?**

Java passes all parameters by values. The references to objects are passed by values.

1. **Primitive data types are passed by reference or pass by value?**

See above

1. **Objects are passed by value or by reference?**

See above

1. **What are pass by reference and pass by value?**

Pass by reference -- a reference to the Object is passed to the method. Pass by value -- a copy of the actual value of the Object is passed to the method. The method is then unable to modify the original Object. (Not true in Java. In Java references to objects are passed by value)

1. **Give a simplest way to find out the time a method takes for execution without using any profiling tool?**

System.currentTimeMillis() in the beginning and end of the method

1. **Is Empty .java file a valid source file?**

Yes it is.

1. **Can a .java file contain more than one java classes?**

Yes it can. It has to contain only one top level public java class but it can contain any number of inner, anonymous and top level classes with default access modifier.

1. **Is String a primitive data type in Java?**

No. String is an Object. An immutable one.

1. **Is main a keyword in Java?**

No. main is the name of the method that is treated in special way if declared properly.

1. **Is next a keyword in Java?**

No. next() is method of Iterator or Enumeration interface.

1. **Is delete a keyword in Java?**

No. delete is not a keyword. And I cannot even remember a class that has a method called delete(). Rather remove()

1. **Is exit a keyword in Java?**

No. exit() is a method of java.lang.System class

1. **What happens if you dont initialize an instance variable of any of the primitive types in Java?**

It gets assigned the default value. 0 for int and long, 0.0 for float and double, false for boolean. Though I tried to compile a class where variables were not initialized and it didn't compile.

1. **What are the different scopes for Java variables?**

static fields, instance fields, method parameters, local variables

1. **What is the default value of the local variables?**

No default value. Default values are assigned to instance fields. Local variables have to be explicitly initialized.

1. **Does garbage collection guarantee that a program will not run out of memory?**

No. Not at all. Example: recursive call. Or just create lots of objects until you get OutOfMemoryError (or is it an exception?)

1. **What is the purpose of finalization?**

Free up the resources. (e.g. close connections and streams, release a lock etc)

1. **Can a public class MyClass be defined in a source file named YourClass.java?**

no. Unless it is a nested class public class.

1. \*\*What will be the output of the following statement? System.out.println ("1" **+ 3);**

13

1. **What will be the default values of all the elements of an array defined as an instance variable?**

All elements will be initialized to default value of corresponding type.

1. **Length in bytes for primitive types**

| **Primitive type** | **length in bytes** | **Comment** |
| --- | --- | --- |
| boolean | 1 bit | saved as 4 bytes; 1 byte in an array |
| char | 2 bytes | unsigned |
| byte | 1 byte |  |
| short | 2 bytes |  |
| int | 4 bytes |  |
| long | 8 bytes |  |
| float | 4 bytes |  |
| double | 8 bytes |  |

1. **Contract between equals() and hashCode()**

if a.equals(b) returns true then a.hashCode() == b.hashCode() is also true. Note that equal hashCode doesn't mean anything.

1. **What different between StringBuffer and StringBuilder?**

StringBuilder -- new. StringBuffer -- old. StringBuffer -- synchronized. Where possible use StringBuilder. Both represent mutable sequence of characters.

1. **What internal methods of String do you know?**

static methods of String class: valueOf indexOf, lastIndexOf, replace, contains, startsWith, endsWith, substring matches, split equals, isEmpty

1. **Purpose, types and creation of nested classes**

Types of nested classes see above. Purpose: If a package-private top-level class (or interface) is used by only one class, consider making the top-level class a private nested class of the sole class that uses. This reduces its accessibility from all the classes in its package to the one class that uses it. But it is far more important to reduce the accessibility of a gratuitously public class than of a package-private top-level class: the public class is part of the package’s API, while the package-private top-level class is already part of its implementation. One common use of a static member class is as a public helper class, useful only in conjunction with its outer class If an instance of a nested class can exist in isolation from an instance of its enclosing class, then the nested class must be a static member class: it is impossible to create an instance of a nonstatic member class without an enclosing instance. One common use of a nonstatic member class is to define an Adapter that allows an instance of the outer class to be viewed as an instance of some unrelated class. For example, implementations of the Map interface typically use nonstatic member classes to implement their collection views, which are returned by Map’s keySet, entrySet, and values methods. Similarly, implementations of the collection interfaces, such as Set and List, typically use nonstatic member classes to implement their iterators.[Effective Java] A common use of private static member classes is to represent components of the object represented by their enclosing class. For example, consider a Map instance, which associates keys with values. Many Map implementations have an internal Entry object for each key-value pair in the map. One common use of anonymous classes is to create function objects on the fly. For example, the sort method invocation sorts an array of strings according to their length using an anonymous Comparator instance. To recap, there are four different kinds of nested classes, and each has its place. If a nested class needs to be visible outside of a single method or is too long to fit comfortably inside a method, use a member class. If each instance of the member class needs a reference to its enclosing instance, make it nonstatic; otherwise, make it static. Assuming the class belongs inside a method, if you need to create instances from only one location and there is a preexisting type that characterizes the class, make it an anonymous class; otherwise, make it a local class.

1. **What does it mean that an object or a class is mutable or immutable?**

Immutability: the state of the object doesn't change

1. **Is it enough to define this class as final? To make this class immutable?**

No. If the class is declared final it only means that it cannot be subclassed. If the instance of the class is declared to be final it only means that the reference will not change. The inner state of the object in both cases can change.

1. **Besides “String” do you know any other immutable classes?**

BigDecimal, BigInteger all classes that correspond to primitive types java.awt.Color

1. **Increasing/descreasing of methods visibility (inheritence)**

The main rule is that visibility cannot be reduced in the subclass

1. \*\*You need to create the string, which contains 1,000,000 **random numbers, comma separated. How would** you do that, considering performance?

I would use StringBuilder class

1. **Garbage collection principles**

The garbage collector first performs a task called marking. The garbage collector traverses the application graph, starting with the root objects; those are objects that are represented by all active stack frames and all the static variables loaded into the system. Each object the garbage collector meets is marked as being used, and will not be deleted in the sweeping stage. The sweeping stage is where the deletion of objects take place.

1. **Java de-compiler, when it could be helpful except studying, learning, stealing**

Recovering lost sources?TODO????????

1. **How is the virtual space divided in Java?**

<http://stackoverflow.com/questions/1262328/how-is-the-java-memory-pool-divided> <http://docs.oracle.com/javase/6/docs/technotes/guides/management/jconsole.html>

1. **What difference between float and BigDecimal. How they store the data?**

float is floating point number and can loose precision during the computations. BigDeciamal is fixed point number. The computations (which type of computations?) are guaranteed to maintain the needed precision. Internally BigDecimal consists of an arbitrary precision integer unscaled value and a 32-bit integer scale If no rounding mode is specified and the exact result cannot be represented, an exception is thrown

1. **What is deep copy of a Java object?**

Deep copy creates a copy of the object including deep copies of all its fields.

1. **What are utilities for JVM monitoring? What is Jconsole?**

From <http://docs.oracle.com/javase/6/docs/technotes/tools/>

jvisualvm -- A graphical tool that provides detailed information about the Java technology-based applications (Java applications) while they are running in a Java Virtual Machine. Java VisualVM provides memory and CPU profiling, heap dump analysis, memory leak detection, access to MBeans, and garbage collection.See Java VisualVM for more information.

jconsole -- A JMX-compliant graphical tool for monitoring a Java virtual machine. It can monitor both local and remote JVMs. It can also monitor and manage an application.

jps -- Experimental: JVM Process Status Tool - Lists instrumented HotSpot Java virtual machines on a target system.

jstat -- Experimental: JVM Statistics Monitoring Tool - Attaches to an instrumented HotSpot Java virtual machine and collects and logs performance statistics as specified by the command line options.

jstatd -- Experimental: JVM jstat Daemon - Launches an RMI server application that monitors for the creation and termination of instrumented HotSpot Java virtual machines and provides a interface to allow remote monitoring tools to attach to Java virtual machines running on the local system.

jhat -- Experimental - Heap Dump Browser - Starts a web server on a heap dump file (eg, produced by jmap -dump), allowing the heap to be browsed.

jmap -- Experimental - Memory Map for Java - Prints shared object memory maps or heap memory details of a given process or core file or a remote debug server.

jsadebugd -- Experimental - Serviceability Agent Debug Daemon for Java - Attaches to a process or core file and acts as a debug server.

jstack -- Experimental - Stack Trace for Java - Prints a stack trace of threads for a given process or core file or remote debug server.

Never used any of these tools yet. :(

1. **How and in what cases we need to configure sizes of memory areas in Java?**

In case of getting OutOfMemoryError: Java heap space. What other cases? JVM parameter -Xmx###m where ### is number of megabytes you need for the JVM. -Xms###m to set the initial heap size More info on this topic can be found here: <http://blog.codecentric.de/en/2011/03/java-memory-configuration-and-monitoring-3rd-act/>

1. **What is an Object and how do you allocate memory to it?**

Object is the base class in Java. Object in general case is an instance of a class. The memory is allocated when new operator is executed. The minimum size is 8 bytes (thats what you get if you call new Object). Each primitive data type takes 4 bytes, except double and long, which take 8 bytes.

1. **What are methods and how are they defined?**

Method is an abstraction that defines how a specific computation has to be carried out. Method "abstracts away" the code it contains.

1. **What is the use of bin and lib in JDK?**

bin -- all java binaries: javac, java, appletviewer, jconsole... lib -- java libraries

1. **What is casting?**

changing the type of the object.

1. **In how many ways can an argument be passed to a subroutine and explain them?**

only one. By value. See above

1. **What is the difference between an argument and a parameter?**

parameter -- abstract. argument -- concrete value of the parameter. parameters of the function are defined when the function is declared arguments of the funciton are defined when it is called

1. **What is final, finalize() and finally?**

final -- Java keyword, see above finalize() -- gets called before the object is GC-ed finally -- Java keyword used in exception handling.

1. **What is UNICODE?**

See info on Unicode here <http://docs.oracle.com/javase/1.5.0/docs/api/java/lang/Character.html>

1. **What is finalize() method?**

See above.

1. **What are Transient and Volatile Modifiers?**

Transient signifies that the field is not part of the object state (e.g. it's some derieved value or some cache). Transient fields are not present in serialized representation of the object. If field is declared with volatile keyword then any thread that reads the field will see the most recently written value [Effective Java Item 66]

1. **What is difference between overloading and overriding?**

overloading -- adding a method with the same name but different signature overriding -- changing the method implementation in the subclass

1. **What is meant by Inheritance and what are its advantages?**

Inheritance is one of principles of OOP. It allows to create class hierarchies. Classes can inherit methods and properties from the base classes thus increasing code reuse.

1. **What is the difference between this() and super()?**

this() calls the constructor of current class. super() calls the superclass constructor super() has to be the first statement of subclass constructor; this and super are references to the current object and to current object treated as superclass. this.new Something() has to be used to create inner classes

1. **What is the difference between superclass and subclass?**

Obvious.

1. **What modifiers may be used with top-level class?**

only public or default (package-private)

1. **What is a package?**

In Java package is a mechanism to oragnize classes into modules.

1. **What is a reflection package?**

Not sure the question is clearly stated. What I would answer is pasted from javadoc of java.lang.reflect Provides classes and interfaces for obtaining reflective information about classes and objects. Reflection allows programmatic access to information about the fields, methods and constructors of loaded classes, and the use reflected fields, methods, and constructors to operate on their underlying counterparts on objects, within security restrictions.

1. **What is the difference between Integer and int?**

Integer is a wrapper class for int primitive type. Integer can be used in generic collections whereas int cannot. Also contains a number of utility methods.

1. **What is a cloneable interface and how many methods does it contain?**

Cloneable -- is a marker interface and it doesn't contain any methods. It determines the behavior of Object’s protected clone implementation: if a class implements Cloneable, Object’s clone method returns a field-by-field copy of the object; otherwise it throws CloneNotSupportedException

1. **Can you have an inner class inside a method and what variables can you access?**

You can create a local or anonymous class inside the method. It can access only final variables.

1. **What is the difference between String and StringBuffer?**
2. **What is the difference between a while statement and a do statement?**

do is guaranteed to execute at least once.

1. **What is the difference between static and non-static variables?**

a) The way they are initialized. Static are initalized when the class is loaded. Non-static -- when it's instantiated. b) Non-static belong to the instance of an object while static are class variables. c) Static are accessed using ClassName.varName

1. **How does Java handle integer overflows and underflows?**

It goes down to the MIN\_INT value in case of overflow and to MAX\_INT in case of overflow

**Main method**

1. **Can an application have multiple classes having main method?**

Yes it can. But only one main method will be used as the entrance point for the program.

1. **Can I have multiple main methods in the same class?**

No, if you mean public static void main(String[] args). Yes, if you mean a method with a name "main" and any other signature

1. **What if the main method is declared as private?**

The class will compile. But the method cannot be used as the entrance point

1. **What if the static modifier is removed from the signature of the main method?**

It becomes an instance method. No longer an entrance point but just a valid regular method.

1. **What if I write static public void instead of public static void?**

Nothing happens. Class compiles. Method runs.

1. **What if I do not provide the String array as the argument to the method?**

You just define a static method called "main" with no parameters. It cannot be used as entrance point.

1. **What is the first argument of the String array in main method?**

These are the parameters passed to the program from command line.

1. **If I do not provide any arguments on the command line, then the String array of Main method will be** empty or null?

Array of size 0

1. **Can main method be declared final?**

Yes it can be declared as final.

* © 2021 GitHub, Inc.
* [Terms](https://docs.github.com/en/github/site-policy/github-terms-of-service)
* [Privacy](https://docs.github.com/en/github/site-policy/github-privacy-statement)
* [Security](https://github.com/security)
* [Status](https://www.githubstatus.com/)
* [Docs](https://docs.github.com/)
* [Contact GitHub](https://support.github.com/)
* [Pricing](https://github.com/pricing)
* [API](https://docs.github.com/)
* [Training](https://services.github.com/)
* [Blog](https://github.blog/)
* [About](https://github.com/about)

[Skip to content](https://github.com/svozniuk/java-interviews/blob/master/Java%20Core/Java%20Collections.txt#start-of-content)

* Why GitHub?
* [Team](https://github.com/team)
* [Enterprise](https://github.com/enterprise)
* Explore
* [Marketplace](https://github.com/marketplace)
* Pricing

Top of Form



Bottom of Form

[Sign in](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews%2Fblob%2Fmaster%2FJava%2520Core%2FJava%2520Collections.txt)[Sign up](https://github.com/join?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F%3Cuser-name%3E%2F%3Crepo-name%3E%2Fblob%2Fshow&source=header-repo&source_repo=svozniuk%2Fjava-interviews)

[**svozniuk**](https://github.com/svozniuk)**/**[**java-interviews**](https://github.com/svozniuk/java-interviews)

* [Notifications](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews)
* [Star](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews)[**251**](https://github.com/svozniuk/java-interviews/stargazers)
* [Fork](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews)[**128**](https://github.com/svozniuk/java-interviews/network/members)
* [**Code**](https://github.com/svozniuk/java-interviews)
* [Issues3](https://github.com/svozniuk/java-interviews/issues)
* [Pull requests2](https://github.com/svozniuk/java-interviews/pulls)
* [Actions](https://github.com/svozniuk/java-interviews/actions)
* [Projects](https://github.com/svozniuk/java-interviews/projects)
* [Wiki](https://github.com/svozniuk/java-interviews/wiki)
* [Security](https://github.com/svozniuk/java-interviews/security)
* [Insights](https://github.com/svozniuk/java-interviews/pulse)

 master

[**java-interviews**](https://github.com/svozniuk/java-interviews)**/**[**Java Core**](https://github.com/svozniuk/java-interviews/tree/master/Java%20Core)**/Java Collections.txt**

[Go to file](https://github.com/svozniuk/java-interviews/find/master)

[@svozniuk](https://github.com/svozniuk)

[svozniuk](https://github.com/svozniuk) [Tabs -> spaces; CRLF -> LF;](https://github.com/svozniuk/java-interviews/commit/24f1eb65cdcdb382e62e319dbf6cd7fc14bf4b38) [Fixes](https://github.com/svozniuk/java-interviews/commit/24f1eb65cdcdb382e62e319dbf6cd7fc14bf4b38) [#2](https://github.com/svozniuk/java-interviews/issues/2) [and](https://github.com/svozniuk/java-interviews/commit/24f1eb65cdcdb382e62e319dbf6cd7fc14bf4b38) [#3](https://github.com/svozniuk/java-interviews/issues/3)

Latest commit [24f1eb6](https://github.com/svozniuk/java-interviews/commit/24f1eb65cdcdb382e62e319dbf6cd7fc14bf4b38) on Jul 2, 2015[**History**](https://github.com/svozniuk/java-interviews/commits/master/Java%20Core/Java%20Collections.txt)

**1** contributor

158 lines (97 sloc)  9.77 KB

[Raw](https://github.com/svozniuk/java-interviews/raw/master/Java%20Core/Java%20Collections.txt)[Blame](https://github.com/svozniuk/java-interviews/blame/master/Java%20Core/Java%20Collections.txt)

|  |  |
| --- | --- |
|  | ------------------------------------------------------------------------------------------ |
|  | Collections |
|  | ------------------------------------------------------------------------------------------ |
|  |  |
|  | 1. Draw Collections Framework Class Diagram |
|  |  |
|  | interface Iterable |
|  | interface Collection extends Iterable |
|  | interfaces List, Queue, Set extend Collection |
|  | interface SortedSet extends Set |
|  | interface Map |
|  | interface SortedMap extends Map |
|  | classes ArrayList, Vector, Stack, LinkedList implement List |
|  | classes HashSet, LinkedHashSet implement Set |
|  | class TreeSet implements SortedSet |
|  | classes HashMap, WeakHashMap, LinkedHashMap implement Map |
|  | class TreeMap implements SortedMap |
|  | utility classes Collections and Arrays |
|  |  |
|  | 2. What is HashMap and Map? |
|  |  |
|  | Map is an interface. Contains methods to manipulate Key-Value based collections. The main methods of Map interface are put(K,V), get(K), Collection<V> values(), Set<K> keySet(), containsKey(), containsValue() |
|  | HashMap is one of implementations of the Map interface based on hashcodes of objects used as keys. |
|  |  |
|  | 3. Difference between HashMap and HashTable? Can we make hashmap synchronized? |
|  |  |
|  | Both implement Map interface. HashTable is synchronized. It is recommended to use HashMap wherever possible. HashTable doesn't allow null keys and values. HashMap allows one null key and any number of null values. |
|  | We can make it synchronized |
|  | Map m = Collections.synchronizedMap(new HashMap()); |
|  |  |
|  | 4. Difference between Vector and ArrayList? |
|  |  |
|  | Both implement List interface. ArrayList is not synchronized. |
|  |  |
|  | 5. What is an Iterator? |
|  |  |
|  | It is an interface that contains three methods: next(), boolean hasNext(), void remove() |
|  | It allows to iterate over the collection |
|  | If the class implements iterator then it can be used in foreach loop |
|  |  |
|  | 6. List vs Set vs Map. Purposes and definitions. |
|  |  |
|  | All three are interfaces. |
|  |  |
|  | List -- storing values in specified order. Provides methods to get the element by its position get(i), finding element, ListIterator. Known implementations: ArrayList, Vector, LinkedList. List should be used when the order in which the elements are stored matters. |
|  |  |
|  | Set -- storing only different objects and at most one null element. Known implementations: TreeSet (iterate over the elements in order defined by Comparator, or if the elements implement comparable; provides log(n) performance for basic operations), HashSet -- stores values in buckets defined by their hashcodes. Each bucket is a singly linked list. Provides constant time performance for basic operations. LinkedHashSet |
|  |  |
|  | Map -- for storing key-value pairs. Map cannot contain duplicate keys. Provides three collection views: set of keys, collection of values, set of key-value mappings. Know implementations HashMap, EnumMap, TreeMap, LinkedHashMap, WeakHashMap. |
|  |  |
|  | 7. Pros and cons of ArrayList and LinkedList |
|  |  |
|  | ArrayList -- fast random access. |
|  | LinkedList -- slow random access. Implements Queue interface. Fast deletion of the element. |
|  | If lots of random reads is anticipated use ArrayList. |
|  | If lots of iterations over the whole list and lots of add/delete -- use LinkedList. |
|  |  |
|  | 8. TreeSet vs LinkedHashSet |
|  |  |
|  | LinkedHashSet is backed by LinkedHashMap. LinkedHashMap is backed by doubly linked list to enforce ordering on the elements contained in the Map. |
|  | If the ordering of the elements in the Set matters to you but you don't want to use a comparator you may use LinkedHashSet since it will enforce ordering in which the elements were added to the set. Otherwise use TreeSet |
|  |  |
|  | 9. What are relationships between equals and hash codes? |
|  |  |
|  | See in Java Basics. |
|  |  |
|  | 10. What are the advantages of ArrayList over arrays ? |
|  |  |
|  | 1. ArrayList comes with a number of utility methods (e.g. contains, remove, addAll) |
|  | 2. Type safety |
|  | 3. Dynamic sizing |
|  | On the other hand arrays are a little bit faster and take less memory (packing). Arrays are also able to contain values of primitive types while ArrayList can only contain Objects. |
|  |  |
|  | 11. Principle of storing data in a hashtable |
|  |  |
|  | HashSet. add(element) -> element.hashCode() -> mod bucketsCount -> store |
|  | HashMap. add(key, value) -> key.hashCode() -> mod bucketsCount -> store(value) |
|  |  |
|  | 12. Differences between Hashtable, ConcurrentHashMap and Collections.synchronizedMap() |
|  |  |
|  | ConcurrentHashMap allows concurrent modification of the Map from several threads without the need to block them. Collections.synchronizedMap(map) creates a blocking Map which will degrade performance, albeit ensure consistency (if used properly). |
|  | Use the second option if you need to ensure data consistency, and each thread needs to have an up-to-date view of the map. Use the first if performance is critical, and each thread only inserts data to the map, with reads happening less frequently. |
|  |  |
|  | 13. How are hash codes computed? |
|  |  |
|  | if hashCode() method is defined then it is called to calculate the hashcode |
|  | if its not defined the default implementation in Object class does the following: |
|  |  |
|  | public int hashCode() { |
|  | return VMMemoryManager.getIdentityHashCode(this); |
|  | } |
|  |  |
|  | 14. Is it possible that hash code is not unique? |
|  |  |
|  | It's totally possible. Actually a totally valid hashCode() function could look like this |
|  |  |
|  | int hashCode(){ return 57; } |
|  |  |
|  | 15. Can we put two elements with equal hash code to one hash map? |
|  |  |
|  | Yes we can. The hashcode of objects doesn't matter. Only the hashcode of keys. But even if you want to put keys with the same hashcode it will be ok since it just means that key-value pairs will be put into the same bucket |
|  |  |
|  | 16. Iterator and modification of a List. ConcurentModificationException. |
|  |  |
|  | The iterators returned by this class's iterator method are fail-fast: if the set is modified at any time after the iterator is created, in any way except through the iterator's own remove method, the iterator will throw a ConcurrentModificationException. Thus, in the face of concurrent modification, the iterator fails quickly and cleanly, rather than risking arbitrary, non-deterministic behavior at an undetermined time in the future. |
|  |  |
|  | Note that the fail-fast behavior of an iterator cannot be guaranteed as it is, generally speaking, impossible to make any hard guarantees in the presence of unsynchronized concurrent modification. Fail-fast iterators throw ConcurrentModificationException on a best-effort basis. Therefore, it would be wrong to write a program that depended on this exception for its correctness: the fail-fast behavior of iterators should be used only to detect bugs. |
|  |  |
|  | 17. What is the significance of ListIterator? What is the difference b/w Iterator and ListIterator? |
|  |  |
|  | ListIterator allows to perform iteration both ways (first-->last and last-->first) |
|  | From JavaDoc: ListIterator is an iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list |
|  |  |
|  | 18. What is the Collections API? |
|  |  |
|  | See above |
|  |  |
|  | 19. How can we access elements of a collection? |
|  |  |
|  | Depends on the collection type. In general: using an iterator, getting by index, getting by key, using iterator on a view |
|  |  |
|  | 20. What’s the difference between a queue and a stack? |
|  |  |
|  | In Java Queue is an interface and has a number of implementations. Stack is a class derieved from Vector class. |
|  | The Queue interface contains methods to enqueue and dequeue elements: offer(), poll(). Also contains methods to see the first element in the queue: peek(). The queue ordering is not neccessairly FIFO. It can be based on the comparator (PriorityQueue) |
|  | The Stack contains methods to add and remove elements to/from a stack. If used correctly enforces LIFO policy. Contains methods push(), pop(), peek(). The problem with Java stack implementation is that it contains all the methods from Vector class which can lead to violation of LIFO policy. |
|  |  |
|  | 21. What is the Properties class? |
|  |  |
|  | Property class is designed to hold properties (i.e. pairs <String, String> -- property name (key) and property value). The two main methods of Properties class is getProperty(String name) and setProperty(String name, String value). Properties class extends HashTable |
|  |  |
|  | 22. Which implementation of the List interface provides for the fastest insertion of a new element into the middle of the list? |
|  |  |
|  | LinkedList. It only requires 4 assignments. |
|  | Before: ... <--> A <--> B <--> ... |
|  | After: ... <--> A <--> C <--> B <--> ... . A changes one reference, B changes one reference, C assigns two references. |
|  |  |
|  | 23. How can we use hashset in collection interface? |
|  |  |
|  | I'm not sure I understand this question fully. HashSet implements Set interface which in turn extends the Collection interface. Thus HashSet is a valid class to be used everywhere where Collection interface is required. |
|  |  |
|  | 25. Can you limit the initial capacity of vector in java? |
|  |  |
|  | I'm not sure if limit is correct word to be used in this question. One can just set the initial capacity of the Vector. Well you could say that you are putting lower limit on the size of the Vector. |
|  |  |
|  | 26. What method should the key class of Hashmap override? |
|  |  |
|  | equals() and hashCode(). |
|  |  |
|  | 27. What is the difference between Enumeration and Iterator? |
|  |  |
|  | Enumeration is an interface similar to Iterator. But it doesn't have remove() method. Enumeration acts as Read-only interface, because it has the methods only to traverse and fetch the objects, where as using Iterator we can manipulate the objects also like adding and removing the objects |
|  |  |
|  | 28. Collections class and Arrays class |
|  |  |
|  | The Collections class contains a number of factory methods for creating unmodifiable views or synchronized wrappers of collections, as well as utility methods such as sort, shuffle, reverse |
|  |  |
|  | The Arrays class contains a factory method that take an array as input and return a List based on this array (Arrays.asList()). Also contains methods to sort, search, fill and print arrays |

* © 2021 GitHub, Inc.
* [Terms](https://docs.github.com/en/github/site-policy/github-terms-of-service)
* [Privacy](https://docs.github.com/en/github/site-policy/github-privacy-statement)
* [Security](https://github.com/security)
* [Status](https://www.githubstatus.com/)
* [Docs](https://docs.github.com/)
* [Contact GitHub](https://support.github.com/)
* [Pricing](https://github.com/pricing)
* [API](https://docs.github.com/)
* [Training](https://services.github.com/)
* [Blog](https://github.blog/)
* [About](https://github.com/about)

[Skip to content](https://github.com/svozniuk/java-interviews/blob/master/Java%20Core/Java%20Concurrency.txt#start-of-content)

* Why GitHub?
* [Team](https://github.com/team)
* [Enterprise](https://github.com/enterprise)
* Explore
* [Marketplace](https://github.com/marketplace)
* Pricing

Top of Form



Bottom of Form

[Sign in](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews%2Fblob%2Fmaster%2FJava%2520Core%2FJava%2520Concurrency.txt)[Sign up](https://github.com/join?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F%3Cuser-name%3E%2F%3Crepo-name%3E%2Fblob%2Fshow&source=header-repo&source_repo=svozniuk%2Fjava-interviews)

[**svozniuk**](https://github.com/svozniuk)**/**[**java-interviews**](https://github.com/svozniuk/java-interviews)

* [Notifications](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews)
* [Star](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews)[**251**](https://github.com/svozniuk/java-interviews/stargazers)
* [Fork](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews)[**128**](https://github.com/svozniuk/java-interviews/network/members)
* [**Code**](https://github.com/svozniuk/java-interviews)
* [Issues3](https://github.com/svozniuk/java-interviews/issues)
* [Pull requests2](https://github.com/svozniuk/java-interviews/pulls)
* [Actions](https://github.com/svozniuk/java-interviews/actions)
* [Projects](https://github.com/svozniuk/java-interviews/projects)
* [Wiki](https://github.com/svozniuk/java-interviews/wiki)
* [Security](https://github.com/svozniuk/java-interviews/security)
* [Insights](https://github.com/svozniuk/java-interviews/pulse)

 master

[**java-interviews**](https://github.com/svozniuk/java-interviews)**/**[**Java Core**](https://github.com/svozniuk/java-interviews/tree/master/Java%20Core)**/Java Concurrency.txt**

[Go to file](https://github.com/svozniuk/java-interviews/find/master)

[@svozniuk](https://github.com/svozniuk)

[svozniuk](https://github.com/svozniuk) [Tabs -> spaces; CRLF -> LF;](https://github.com/svozniuk/java-interviews/commit/24f1eb65cdcdb382e62e319dbf6cd7fc14bf4b38) [Fixes](https://github.com/svozniuk/java-interviews/commit/24f1eb65cdcdb382e62e319dbf6cd7fc14bf4b38) [#2](https://github.com/svozniuk/java-interviews/issues/2) [and](https://github.com/svozniuk/java-interviews/commit/24f1eb65cdcdb382e62e319dbf6cd7fc14bf4b38) [#3](https://github.com/svozniuk/java-interviews/issues/3)

Latest commit [24f1eb6](https://github.com/svozniuk/java-interviews/commit/24f1eb65cdcdb382e62e319dbf6cd7fc14bf4b38) on Jul 2, 2015[**History**](https://github.com/svozniuk/java-interviews/commits/master/Java%20Core/Java%20Concurrency.txt)

**1** contributor

275 lines (191 sloc)  15.6 KB

[Raw](https://github.com/svozniuk/java-interviews/raw/master/Java%20Core/Java%20Concurrency.txt)[Blame](https://github.com/svozniuk/java-interviews/blame/master/Java%20Core/Java%20Concurrency.txt)

|  |  |
| --- | --- |
|  | ------------------------------------------------------------------------------------------ |
|  | Concurrency |
|  | ------------------------------------------------------------------------------------------ |
|  | 1. Describe synchronization in respect to multi-threading. What is synchronization. |
|  |  |
|  | Several threads access common data. In order to keep the data in consistent state the access to it has to be synchronized (i.e. some ordering of data access has to be imposed). |
|  |  |
|  | 2. Explain different ways of using thread? |
|  |  |
|  | a) Create a long-running computation in a separate thread so the user interface (or whatever other part of the application) is not blocked. |
|  | b) separate out I/O operations which potentially can take a lot of time (e.g. reading from the network) for the same reason |
|  | c) process incoming requests in parallel (usually using thread pool of some size) |
|  | d) Create thread to do some kind of isolated processing, wait for the processing to finish, kill the thread. Create new threads when needed |
|  |  |
|  | 3. What is the difference between preemptive scheduling and time slicing? |
|  |  |
|  | Not exactly a correct question. |
|  |  |
|  | A scheduler can be preemptive (it's capable to force a process to interrupt its execution and resume it in some time) and non-preemptive (which is unable to interrupt a process and relies on the processes themselves that voluntarily give control to other tasks). Time slicing is a usual technique that is used in a preemptive multitasking system. The scheduler is run every time slice to choose the next process to run (it may happen that it's the same process during few time slices in a row, or it may happen that every time slice a different process is executed ) |
|  |  |
|  | To apply these terms to Java world -- replace the "process" with "thread", since the ideas behind scheduling processes in OS are the same as scheduling threads in an application. |
|  |  |
|  | 4. When a thread is created and started, what is its initial state? |
|  |  |
|  | The thread is then in "RUNNABLE" state. |
|  |  |
|  | 5. Thread vs Runnable, run() vs start() |
|  |  |
|  | See below. The main difference between run() and start() is that the latter creates a separate thread while the former executes the code synchronously |
|  |  |
|  | 6. Describe different ways to create a thread. |
|  |  |
|  | a) |
|  | class MyRunnable extends SomeOtherClass implements Runnable { |
|  | public void run(){ |
|  | // code that has to run in a thread |
|  | } |
|  | } |
|  |  |
|  | MyRunnable r = new MyRunnable(); |
|  | Thread t = new Thread(r); |
|  | r.start(); |
|  |  |
|  | b) |
|  | class MyThread extends Thread { |
|  | public void run(){ |
|  | // code that has to run in a thread |
|  | } |
|  | } |
|  |  |
|  | Thread t = new MyThred(); |
|  | t.start(); |
|  |  |
|  | c) |
|  | class MySomething extends Something { |
|  | public void doSomeStuff() {...} |
|  | } |
|  |  |
|  | new Thread(new Runnable() { |
|  | public void run(){ |
|  | instanceOfMySomething.doSomeStuff(); |
|  | } |
|  | }).start(); |
|  |  |
|  | The main difference between these two approaches is that in case (a) you are able to extend the class you need while still being able to run your code in a separate thread. In case (b) you are already extending from the Thread class which limits your options. In general following one of the good OOP practices (Favor composition over inheritance) option (a) is preferable. Option (c) is also cute since it decouples your class and the fact that its code will be run in a separate thread. In other words you can still call instanceOfMySomething.doSomeStuff() regardless from a new thread or from the same. |
|  |  |
|  | 7. Synchronization of Java blocks and methods |
|  |  |
|  | Methods declared synchronized and statements contained in synchronized blocks. Only one thread is allowed to be executing instructions inside a synchronized block. The main difference between synchronized block and synchronized method is that you can choose which object will be used for synchronization in case of blocks. The methods are always synchronized with "this" |
|  |  |
|  | 8. Explain usage of the couple wait()/notify() |
|  |  |
|  | The mechanism is used to allow one thread to signal another. E.g. Consumer signals that he's waiting for the next object to process, or Producer signals that a new object is ready to be processed. |
|  |  |
|  | wait( ) tells the calling thread to give up the monitor and go to sleep until some other thread enters the same monitor and calls notify( ). |
|  | notify( ) wakes up the a thread (a random one?) that called wait( ) on the same object. |
|  | notifyAll( ) wakes up all the threads that called wait( ) on the same object. |
|  |  |
|  |  |
|  | 9. What does Volatile keyword mean? |
|  |  |
|  | The volatile keyword guarantees that reads and from a variable will see the changes made by the last write to this variable (and of course all other writes that happened earlier). Java documentation states that volatile establishes a "happens-before" relationship between a write to a variable and all subsequent reads from it. |
|  |  |
|  | 10. java.util.concurrent.\*, what utils do you know? |
|  |  |
|  | Synchronization primitives: Semaphore, CyclicBarrier, CountDownLatch, Lock, ReentrantLock |
|  | Threads: Executors, Callable and Future |
|  | Data: Synchronized collections (CopyOnWriteArrayList, ConcurrentHashMap, BlockingQueue) |
|  |  |
|  | 11. ThreadLocal, what for are they needed? Does child thread see the value of parent ThreadLocal? |
|  |  |
|  | Values stored in Thread Local are global to the thread, meaning that they can be accessed from anywhere inside that thread. If a thread calls methods from several classes, then all the methods can see the Thread Local variable set by other methods (because they are executing in same thread). The value need not be passed explicitly. It’s like how you use global variables. |
|  |  |
|  | One possible (and common) use is when you have some object that is not thread-safe, but you want to avoid synchronizing access to that object. Instead, give each thread its own instance of the object. ThreadLocals are one sort of global variables (although slightly less evil because they are restricted to one thread), so you should be careful when using them to avoid unwanted side-effects and memory leaks. Design your APIs so that the ThreadLocal values will always be automatically cleared when they are not anymore needed and that incorrect use of the API won't be possible (for example like this). |
|  |  |
|  | No. The child thread doesn't see the value of parent ThreadLocal unless InheritableThreadLocal<T> is used. |
|  |  |
|  | 12. Recommendations to avoid deadlocks. |
|  |  |
|  | a) We may create a class that will register all locks being held by all the threads. Thus before granting a new lock to a thread we may check whether it will not lead to a deadlock and grant the lock only if it doesn't. |
|  | b) We may devise a policy of acquiring the locks by the threads that will guarantee a deadlock-free program. A simple example of such policy is: no thread is allowed to have more than one lock at a time. |
|  |  |
|  | 13. What is daemon thread and which method is used to create the daemon thread? |
|  |  |
|  | A daemon thread is a thread, that does not prevent the JVM from exiting when the program finishes but the thread is still running. An example for a daemon thread is the garbage collection. The setDaemon() method can be used to change the Thread daemon properties. Normal thread and daemon threads differ in what happens when they exit. When the JVM halts any remaining daemon threads are abandoned: finally blocks are not executed, stacks are not unwound - JVM just exits. Due to this reason daemon threads should be used sparingly and it is dangerous to use them for tasks that might perform any sort of I/O. |
|  |  |
|  | 14. What method must be implemented by all threads? |
|  |  |
|  | run(); by default (in the Thread class itself) it does nothing and returns |
|  |  |
|  | 15. What is the difference between process and thread? |
|  |  |
|  | From Java documentation: A process has a self-contained execution environment. A process generally has a complete, private set of basic run-time resources; in particular, each process has its own memory space. Most implementations of the Java virtual machine run as a single process |
|  |  |
|  | Threads are sometimes called lightweight processes. Both processes and threads provide an execution environment, but creating a new thread requires fewer resources than creating a new process. Threads exist within a process - every process has at least one. Threads share the process's resources, including memory and open files. This makes for efficient, but potentially problematic, communication. |
|  |  |
|  | 16. What are the states associated in the thread? |
|  |  |
|  |  |
|  |  |
|  | 17. When you will synchronize a piece of your code? |
|  |  |
|  | Whenever there will be a need: most probably when there will be concurrent access to some data |
|  |  |
|  | 18. What is deadlock? Example of deadlock. |
|  |  |
|  | Deadlock is a state in which some threads of an application (at least two threads) are mutually blocked (A waits for resource X held by B, while B waits for resource Y held by A). Neither can continue in this case. Note that only part of application can be in a deadlock state while other thread continue their execution. |
|  |  |
|  | public class DeadlockTest { |
|  | public static void main(String[] args) { |
|  |  |
|  | String s1 = "S1"; |
|  | String s2 = "S2"; |
|  |  |
|  | Thread t1 = new Thread(new DeadlockCause(s1, s2)); |
|  | Thread t2 = new Thread(new DeadlockCause(s2, s1)); |
|  |  |
|  | t1.start(); |
|  | t2.start(); |
|  |  |
|  | } |
|  | } |
|  |  |
|  | class DeadlockCause implements Runnable { |
|  |  |
|  | private final Object firstLock; |
|  | private final Object secondLock; |
|  |  |
|  | public DeadlockCause(Object o1, Object o2) { |
|  | firstLock = o1; |
|  | secondLock = o2; |
|  | } |
|  |  |
|  | @Override |
|  | public void run() { |
|  | synchronized(firstLock){ |
|  | System.out.println(Thread.currentThread().getName() + " holds the lock on " + firstLock); |
|  | try { |
|  | Thread.sleep(1000); |
|  | } catch (InterruptedException ex) { |
|  | Logger.getLogger(DeadlockCause.class.getName()).log(Level.SEVERE, null, ex); |
|  | } |
|  |  |
|  | System.out.println(Thread.currentThread().getName() + " tries to get the lock on " + secondLock); |
|  | synchronized(secondLock){ |
|  | System.out.println(Thread.currentThread().getName() + " holds the lock on " + secondLock); |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | 19. Are there any global variables in Java, which can be accessed by other part of your program? |
|  |  |
|  | No, there are no global variables in Java. |
|  |  |
|  | 20. What are Callable and FutureTask interfaces? |
|  |  |
|  | Motivation to create: with Thread and Runnable you cannot return a value as the result of executing the task. Additionally you have to process all the exception inside the run() method because its declaration is public void run() and doesn't allow exceptions to be thrown. |
|  | The Callable has the method public T call() throws Exception. When an ExecutorService gets submitted a Callable<T> it returns a Future<T>. The get() method on this Future<T> instance has to be called in order to get the result |
|  |  |
|  | Example: |
|  |  |
|  | ExecutorService executor = Executors.newFixedThreadPool(); |
|  | Future<Integer> future = executor.submit(new Callable<Integer>(){ |
|  | public Integer call() throws Exception { |
|  | Integer result = 0; |
|  | // do some stuff here |
|  | return result; |
|  | } |
|  | }); |
|  |  |
|  | executor.shutdown(); // stop accepting new tasks |
|  |  |
|  | try { |
|  | System.out.println("The result is: " + future.get()); |
|  | } catch (InterruptedException ex) { |
|  | ... |
|  | } |
|  |  |
|  |  |
|  | 21. What is Executors framework? |
|  |  |
|  | See |
|  | http://docs.oracle.com/javase/tutorial/essential/concurrency/exinter.html |
|  |  |
|  | 22. What is AtomicInteger |
|  |  |
|  | AtomicInteger is a class from java.util.concurrent package that provides thread-safe implementation of an Integer. |
|  | More information on AtomicXXX can be found at http://docs.oracle.com/javase/6/docs/api/java/util/concurrent/atomic/package-summary.html |
|  |  |
|  |  |
|  | ------------------------------------------------------------------------------------------ |
|  | Advanced Concurrency Questions |
|  | ------------------------------------------------------------------------------------------ |
|  |  |
|  | The following questions have been taken from http://www.rsdn.ru/forum/java/3622844.all and translated into English |
|  | Good set of answers can be found here http://swein2.blogspot.ch/2011/10/concurrency.html |
|  |  |
|  | ------------------------------------------------------------------------------------------ |
|  |  |
|  | Differences in JMM before Java 5 and after Java 5? |
|  | What is the difference between Runnable and Callable? |
|  | What is the difference between java.util.concurrent.Atomic\*.compareAndSwap() and java.util.concurrent.Atomic\*.weakCompareAndSwap()? |
|  | What is the difference between Collections.synchronizedMap(new HashMap()) and ConcurrentHashMap? |
|  | What is the difference between Thread.isInterrupted() and Thread.interrupted()? |
|  | What is the difference between CountDownLatch and CyclicBarrier? |
|  | What is the difference between ReenetrantLock and synchronized{} |
|  | Can you list all the reasons when InterruptedException can be thrown? |
|  | In SynchronousQueue what is unique for BlockingQueue? |
|  | In the String class all the fields are final. Could we remove the final keyword, since the class doesn't have any setters and thus the fields cannot be changed. |
|  | What happens when you call Thread.interrupt()? |
|  | What are advantages of ScheduledThreadPool over java.util.Timer? |
|  | Thread.getState() returns an instance of Thread.State. What are the possible values? |
|  | ReentrantLock created with a true flag (fair) one of the methods to get the lock is not fair. Which one? When to use or not use it? |
|  | Which of the following calls creates a happens-before relationship? Thread.sleep(), Thread.join(), Thread.yield(), Thread.start(), Thread.run(), Thread.isAlive(), Thread.getState()? |
|  | Some of the following methods are deprecated, some have never been implemented. Which of them fall into which category? Thread.interrupt(), Thread.stop(), Thread.yield(), Thread.suspend(), Thread.resume(), Thread.join(), Thread.start(), Thread.run(), Thread.sleep() |
|  | What are ordering, visibility, atomicity, happens-before, mutual exclusion? Which of the listed properties are retained when using volatile, AtomicInteger and synchronized{}? How? |
|  | What do you know about asynchronous method calls? Do they exist in Java? If yes, how they are implemented? If no, how would you implement them? |
|  | What is rendez-vous? Which Java classes can help? |
|  | What is a "green thread"? Are they used in Java (in HotSpon JVM.6)? |
|  | What is cooperative multitasking? Is it used in Java? If not then which type of multitasking is used in Java? What are the advantages. |
|  | What is false sharing? Could it happen in Java? If yes, how could you fight it? If no, how the JVM developers have overcome it? |
|  |  |
|  | Explain safe publication. |
|  | Explain recursive parallelism. |
|  | Explain iterative parallelism. |
|  | Tell something about Reactor/Proactor patterns. |
|  | What does reentrant property mean? |
|  | List the ways to fight priority inversion. Name the types of systems where they are extremely dangerous. |
|  | Name the ways to a) avoid b) overcome a deadlock (imagine you're writing a RDBMS kernel) |
|  |  |
|  | What is monitor? |
|  | What is private mutex? |
|  | What is priority inheritance? |
|  | What is back-off protocol (exponential back-off)? |
|  | What is task-stealing? |
|  | What is spin lock? |
|  | What is sequential consistency? |
|  | What is ABA problem? |
|  | What is "test and set"? |
|  | What is "test and set and set"? |
|  | What is "sense reversing barrier"? |
|  | What is pipeline? |
|  | What is poison message? |
|  | What is mutual exclusion? Examples. |
|  | What is condition waiting? Examples. |
|  |  |
|  | Implement a simple nonblocking stack (only push() and pop()) |
|  | Implement a simple copy-on-write ArrayList (void add(int index, T item), T get(int index), void remove(int index), int size()) |
|  | Implement a simple buffer for multiple producers / multiple consumers using synchronized{} |
|  | Implement a simple buffer for multiple producers / multiple consumers using ReentrantLock |
|  |  |
|  |  |
|  |  |
|  |  |

* © 2021 GitHub, Inc.
* [Terms](https://docs.github.com/en/github/site-policy/github-terms-of-service)
* [Privacy](https://docs.github.com/en/github/site-policy/github-privacy-statement)
* [Security](https://github.com/security)
* [Status](https://www.githubstatus.com/)
* [Docs](https://docs.github.com/)
* [Contact GitHub](https://support.github.com/)
* [Pricing](https://github.com/pricing)
* [API](https://docs.github.com/)
* [Training](https://services.github.com/)
* [Blog](https://github.blog/)
* [About](https://github.com/about)

[Skip to content](https://github.com/svozniuk/java-interviews/blob/master/Java%20Core/Java%20Exceptions.txt#start-of-content)

* Why GitHub?
* [Team](https://github.com/team)
* [Enterprise](https://github.com/enterprise)
* Explore
* [Marketplace](https://github.com/marketplace)
* Pricing

Top of Form



Bottom of Form

[Sign in](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews%2Fblob%2Fmaster%2FJava%2520Core%2FJava%2520Exceptions.txt)[Sign up](https://github.com/join?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F%3Cuser-name%3E%2F%3Crepo-name%3E%2Fblob%2Fshow&source=header-repo&source_repo=svozniuk%2Fjava-interviews)

[**svozniuk**](https://github.com/svozniuk)**/**[**java-interviews**](https://github.com/svozniuk/java-interviews)

* [Notifications](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews)
* [Star](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews)[**251**](https://github.com/svozniuk/java-interviews/stargazers)
* [Fork](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews)[**128**](https://github.com/svozniuk/java-interviews/network/members)
* [**Code**](https://github.com/svozniuk/java-interviews)
* [Issues3](https://github.com/svozniuk/java-interviews/issues)
* [Pull requests2](https://github.com/svozniuk/java-interviews/pulls)
* [Actions](https://github.com/svozniuk/java-interviews/actions)
* [Projects](https://github.com/svozniuk/java-interviews/projects)
* [Wiki](https://github.com/svozniuk/java-interviews/wiki)
* [Security](https://github.com/svozniuk/java-interviews/security)
* [Insights](https://github.com/svozniuk/java-interviews/pulse)

 master

[**java-interviews**](https://github.com/svozniuk/java-interviews)**/**[**Java Core**](https://github.com/svozniuk/java-interviews/tree/master/Java%20Core)**/Java Exceptions.txt**

[Go to file](https://github.com/svozniuk/java-interviews/find/master)

[@svozniuk](https://github.com/svozniuk)

[svozniuk](https://github.com/svozniuk) [Tabs -> spaces; CRLF -> LF;](https://github.com/svozniuk/java-interviews/commit/24f1eb65cdcdb382e62e319dbf6cd7fc14bf4b38) [Fixes](https://github.com/svozniuk/java-interviews/commit/24f1eb65cdcdb382e62e319dbf6cd7fc14bf4b38) [#2](https://github.com/svozniuk/java-interviews/issues/2) [and](https://github.com/svozniuk/java-interviews/commit/24f1eb65cdcdb382e62e319dbf6cd7fc14bf4b38) [#3](https://github.com/svozniuk/java-interviews/issues/3)

Latest commit [24f1eb6](https://github.com/svozniuk/java-interviews/commit/24f1eb65cdcdb382e62e319dbf6cd7fc14bf4b38) on Jul 2, 2015[**History**](https://github.com/svozniuk/java-interviews/commits/master/Java%20Core/Java%20Exceptions.txt)

**1** contributor

101 lines (60 sloc)  4.29 KB

[Raw](https://github.com/svozniuk/java-interviews/raw/master/Java%20Core/Java%20Exceptions.txt)[Blame](https://github.com/svozniuk/java-interviews/blame/master/Java%20Core/Java%20Exceptions.txt)

|  |  |
| --- | --- |
|  | ------------------------------------------------------------------------------------------ |
|  | Exceptions |
|  | ------------------------------------------------------------------------------------------ |
|  |  |
|  | 1. What are Checked and UnChecked Exception? |
|  |  |
|  | Checked -- exceptions that the program has to handle with and to be able to succesfully recover from. |
|  | Unchecked (Runtime) -- usually signal of a programmatic error. Potentially can happen anywhere |
|  | Error -- something really bad happened to the JVM or system. Usually cannot be recovered from. |
|  |  |
|  | 2. What are runtime exceptions? |
|  |  |
|  | See above. |
|  |  |
|  | 3. What is the difference between error and an exception? |
|  |  |
|  | See above. |
|  |  |
|  | 4. How to create custom exceptions? |
|  |  |
|  | Extend from Exception class or from RuntimeException class. |
|  |  |
|  | 5. If I want an object of my class to be thrown as an exception object, what should I do? |
|  |  |
|  | Extend from Exception class (or any of its subclasses) |
|  |  |
|  | 6. If my class already extends from some other class what should I do if I want an instance of my class to be thrown as an exception object? |
|  |  |
|  | Nothing. There is no Exception interface. |
|  |  |
|  | 7. How does an exception permeate through the code? |
|  |  |
|  | It's either caught or it is going to be passed to the next level in call stack. |
|  | If it's caught it can be rethrown as another Exception which has to be processed in a higher level. |
|  |  |
|  | 8. What are the different ways to handle exceptions? |
|  |  |
|  | Declare that the methods throws an exception or catch it. |
|  |  |
|  | 9. What is the basic difference between the 2 approaches to exception handling...1> try catch block and 2> specifying the candidate exceptions in the throws clause? |
|  |  |
|  | The main difference is in who is responsibile for handling the exception. |
|  |  |
|  | 10. When should you use which approach? |
|  |  |
|  | As I understand one should handle the exception if he/she has all the knowledge and resources to do so. And the exception has to be handled as soon as this knowledge is aquired. However if you are writing a library you may consider declaring the exceptions that are thrown and force the user of the library to decide how to react to a specific situation. The downside of this approach is that declared exceptions become the part of the API which has to be frozen after being published. |
|  |  |
|  | 11. Is it necessary that each try block must be followed by a catch block? |
|  |  |
|  | No. It can be followed by either catch block or finally block or both. |
|  |  |
|  | 12. If I write return at the end of the try block, will the finally block still execute? |
|  |  |
|  | Yes it will. |
|  |  |
|  | 13. If I write System.exit (0); at the end of the try block, will the finally block still execute? |
|  |  |
|  | No it won't. It also won't execute if the thread where the exception is thrown is terminated. |
|  |  |
|  | 14. How does a try statement determine which catch clause should be used to handle an exception? |
|  |  |
|  | The first that has a matching exception type declared. |
|  |  |
|  | 15. What is the catch or declare rule for method declarations? |
|  |  |
|  | blah-blah-blah. If you are using a checked exception you either have to catch it or declare that it could be thrown by a method. |
|  |  |
|  | 16. Pros and cons of using exceptions vs. returning "error value" |
|  |  |
|  | Exceptions provide clearer understanding and better code |
|  | Exceptions can be grouped together to be handled |
|  | Exceptions can be propagated up in the call stack to be handled where they are supposed to be handled. |
|  | Exception handling could be slow. |
|  | In some sense Exceptions are unignorable error codes. |
|  |  |
|  | For example, exceptions are intrusive (eg if they're not handled the program dies). If you have an error that can be safely ignored, being forced to catch and handle can be painful. However, they're definitely appropriate if the result of quietly ignoring an error is something that will bite you later. |
|  |  |
|  | 17. Cases when the finally block isn't executed |
|  |  |
|  | System.exit() in try or catch block. |
|  | Error in one of the blocks (e.g. OutOfMemoryError) |
|  | Thread is terminated, while the whole application still continues. |
|  |  |
|  | 18. What is exception handling mechanism |
|  |  |
|  | Blah-blah-blah. |
|  |  |
|  | 19. Bundled exceptions |
|  |  |
|  | IllegalArgumentException |
|  | ConcurrentModificationException |
|  | IllegalStateException |
|  | UnsupportedOperationException |
|  | ArithmeticException |
|  | NullPointerException |
|  | ArrayIndexOutOfBoundsException |
|  | IOException |
|  |  |
|  | 20. How to avoid catch block? |
|  |  |
|  | Declare that method throws the exception. You can still use try/finally. |

* © 2021 GitHub, Inc.
* [Terms](https://docs.github.com/en/github/site-policy/github-terms-of-service)
* [Privacy](https://docs.github.com/en/github/site-policy/github-privacy-statement)
* [Security](https://github.com/security)
* [Status](https://www.githubstatus.com/)
* [Docs](https://docs.github.com/)
* [Contact GitHub](https://support.github.com/)
* [Pricing](https://github.com/pricing)
* [API](https://docs.github.com/)
* [Training](https://services.github.com/)
* [Blog](https://github.blog/)
* [About](https://github.com/about)

[Skip to content](https://github.com/svozniuk/java-interviews/blob/master/Java%20Core/Java%20Generics.txt#start-of-content)

* Why GitHub?
* [Team](https://github.com/team)
* [Enterprise](https://github.com/enterprise)
* Explore
* [Marketplace](https://github.com/marketplace)
* Pricing

Top of Form



Bottom of Form

[Sign in](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews%2Fblob%2Fmaster%2FJava%2520Core%2FJava%2520Generics.txt)[Sign up](https://github.com/join?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F%3Cuser-name%3E%2F%3Crepo-name%3E%2Fblob%2Fshow&source=header-repo&source_repo=svozniuk%2Fjava-interviews)

[**svozniuk**](https://github.com/svozniuk)**/**[**java-interviews**](https://github.com/svozniuk/java-interviews)

* [Notifications](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews)
* [Star](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews)[**251**](https://github.com/svozniuk/java-interviews/stargazers)
* [Fork](https://github.com/login?return_to=%2Fsvozniuk%2Fjava-interviews)[**128**](https://github.com/svozniuk/java-interviews/network/members)
* [**Code**](https://github.com/svozniuk/java-interviews)
* [Issues3](https://github.com/svozniuk/java-interviews/issues)
* [Pull requests2](https://github.com/svozniuk/java-interviews/pulls)
* [Actions](https://github.com/svozniuk/java-interviews/actions)
* [Projects](https://github.com/svozniuk/java-interviews/projects)
* [Wiki](https://github.com/svozniuk/java-interviews/wiki)
* [Security](https://github.com/svozniuk/java-interviews/security)
* [Insights](https://github.com/svozniuk/java-interviews/pulse)

 master

[**java-interviews**](https://github.com/svozniuk/java-interviews)**/**[**Java Core**](https://github.com/svozniuk/java-interviews/tree/master/Java%20Core)**/Java Generics.txt**

[Go to file](https://github.com/svozniuk/java-interviews/find/master)

[@svozniuk](https://github.com/svozniuk)

[svozniuk](https://github.com/svozniuk) [Tabs -> spaces; CRLF -> LF;](https://github.com/svozniuk/java-interviews/commit/24f1eb65cdcdb382e62e319dbf6cd7fc14bf4b38) [Fixes](https://github.com/svozniuk/java-interviews/commit/24f1eb65cdcdb382e62e319dbf6cd7fc14bf4b38) [#2](https://github.com/svozniuk/java-interviews/issues/2) [and](https://github.com/svozniuk/java-interviews/commit/24f1eb65cdcdb382e62e319dbf6cd7fc14bf4b38) [#3](https://github.com/svozniuk/java-interviews/issues/3)

Latest commit [24f1eb6](https://github.com/svozniuk/java-interviews/commit/24f1eb65cdcdb382e62e319dbf6cd7fc14bf4b38) on Jul 2, 2015[**History**](https://github.com/svozniuk/java-interviews/commits/master/Java%20Core/Java%20Generics.txt)

**1** contributor

375 lines (239 sloc)  15.9 KB

[Raw](https://github.com/svozniuk/java-interviews/raw/master/Java%20Core/Java%20Generics.txt)[Blame](https://github.com/svozniuk/java-interviews/blame/master/Java%20Core/Java%20Generics.txt)

|  |  |
| --- | --- |
|  | ------------------------------------------------------------------------------------------ |
|  | Generics |
|  | ------------------------------------------------------------------------------------------ |
|  |  |
|  | Very good collection of Q&A about Java Generics can be found |
|  | http://www.angelikalanger.com/GenericsFAQ/JavaGenericsFAQ.html |
|  |  |
|  |  |
|  | Questions |
|  | ------------------------------- |
|  |  |
|  | 1. What is Generics in Java ? What are advantages of using Generics? |
|  |  |
|  | They allow "a type or method to operate on objects of various types while providing compile-time type safety." A common use of this feature is when using a Java Collection that can hold objects of any type, to specify the specific type of object stored in it. |
|  |  |
|  | A class is generic if it declares one or more type variables. |
|  |  |
|  | 2. How Generics works in Java ? What is type erasure ? |
|  |  |
|  | The type safety is ensured during the compile time. In run time the information about type arguments is erased. The type arguments are replaced by their narrowest superclass. For unbounded parameters -- Object. For bounded -- the declared bound. |
|  |  |
|  | 3. What is Bounded and Unbounded wildcards in Generics ? |
|  |  |
|  | The wildcard type is signified by "?" in Java. Classes parametrized by wildcard are usually read as "class of some type", e.g. List<?> is read List of some type. |
|  |  |
|  | A wildcard parameterized type is an instantiation of a generic type where at least one type argument is a wildcard. Examples of wildcard parameterized types are Collection<?>, List<? extends Number>, Comparator<? super String> and Pair<String,?>.A wildcard parameterized type denotes a family of types comprising concrete instantiations of a generic type. The kind of the wildcard being used determines which concrete parameterized types belong to the family. For instance, the wildcard parameterized type Collection<?> denotes the family of all instantiations of the Collection interface regardless of the type argument. The wildcard parameterized type List<? extends Number> denotes the family of all list types where the element type is a subtype of Number. The wildcard parameterized type Comparator<? super String> is the family of all instantiations of the Comparator interface for type argument types that are supertypes of String. |
|  |  |
|  | A wildcard parameterized type is not a concrete type that could appear in a new expression. A wildcard parameterized type is similar to an interface type in the sense that reference variables of a wildcard parameterized type can be declared, but no objects of the wildcard parameterized type can be created. The reference variables of a wildcard parameterized type can refer to an object that is of a type that belongs to the family of types that the wildcard parameterized type denotes. |
|  |  |
|  | 4. What is wildcard prameter |
|  |  |
|  | In generic code, the question mark (?), called the wildcard, represents an unknown type. The wildcard can be used in a variety of situations: as the type of a parameter, field, or local variable; sometimes as a return type (though it is better programming practice to be more specific). The wildcard is never used as a type argument for a generic method invocation, a generic class instance creation, or a supertype. |
|  |  |
|  | 5. What is difference between List<? extends T> and List <? super T> ? |
|  |  |
|  | The first list may contain elements of type T or any of its subtypes. |
|  | The second list may contain elements of type T or any of its supertypes. |
|  |  |
|  | For any type X such that X is a subtype of T or T itself |
|  | List<X> is a subtype of List<? extends T> |
|  |  |
|  | For any type Y such that Y is a supertype of T or T itself |
|  | List<Y> is a subtype of List<? super T> |
|  |  |
|  | List<?> is supertype of any parametrized list. |
|  |  |
|  | 6. How to write a generic method which accepts generic argument and return Generic Type? |
|  |  |
|  | Generic method in a generic class |
|  |  |
|  | public class SomeClass<T>{ |
|  | private T value; |
|  |  |
|  | // T is already declared |
|  | // since it's type parameter of the class |
|  | public T doSomething(){...} |
|  |  |
|  | // U has to be declared since it's not type parameter of the class |
|  | public <U> List<U> createSingleElementList(U arg){ |
|  | List<U> result = new ArrayList<U>(); |
|  | result.add(arg); |
|  | return result; |
|  | } |
|  | } |
|  |  |
|  | Static generic method |
|  |  |
|  | public static <T> T doSomething(T value){...} |
|  |  |
|  | 7. How to write parametrized class in Java using Generics ? |
|  |  |
|  | class SomeClass<T> extends SomeSuperClass implements SomeInterface |
|  | class SomeClass<T extends Comparable<? super T>> |
|  | class SomeClass<K,V> |
|  |  |
|  | 8. Can you pass List<String> to a method which accepts List<Object> |
|  |  |
|  | No. You cannot do this since there List<Object> is not a superclass of List<String> |
|  |  |
|  | 9. Can we use Generics with Array? |
|  |  |
|  | No. One cannot use generic types with arrays. See below other limitations of generic types. |
|  |  |
|  | 10. What are limitations of Generics |
|  |  |
|  | http://docs.oracle.com/javase/tutorial/java/generics/restrictions.html |
|  |  |
|  | Cannot Instantiate Generic Types with Primitive Types |
|  | Cannot Create Instances of Type Parameters |
|  | Cannot Declare Static Fields Whose Types are Type Parameters |
|  | Cannot Use Casts or instanceof with Parameterized Types |
|  | but you can use (obj instanceOf SomeClass<?>) |
|  | Cannot Create Arrays of Parameterized Types |
|  | Cannot Create, Catch, or Throw Objects of Parameterized Types |
|  | Cannot Overload a Method Where the Formal Parameter Types of Each Overload Erase to the Same Raw Type |
|  |  |
|  | 11. How can you suppress unchecked warning in Java ? |
|  |  |
|  | @SuppressWarnings("unchecked") |
|  |  |
|  | 12. Difference between List<Object> and raw type List in Java? |
|  |  |
|  | List<Object> explicitly states that it contains objects. The type checks are on. Using raw types disable checking even outside of their own declarations. This would happen if we used raw List. |
|  |  |
|  | Another thing that differs significantly is that List<Object> is not covariant with any other parametrized list, while List is. |
|  |  |
|  | Example: |
|  |  |
|  | public static void printList(List l){ |
|  | for (Object obj : l) { |
|  | System.out.println(obj); |
|  | } |
|  | } |
|  |  |
|  | public static void printObjectList(List<Object> l){ |
|  | for (Object obj : l) { |
|  | System.out.println(obj); |
|  | } |
|  | } |
|  |  |
|  | public static void main(String[] args) { |
|  | List<Object> objList = Arrays.asList((Object)new Integer(5), (Object)"x"); |
|  | List<String> stringList = Arrays.asList("A","B","C"); |
|  | List<Integer> intList = Arrays.asList(1, 2, 3); |
|  |  |
|  | printList(objList); // valid |
|  | printList(stringList); // valid |
|  | printList(intList); // valid |
|  |  |
|  | printObjList(objList); // valid |
|  | printObjList(stringList); // invalid. compiler error |
|  | printObjList(intList); // invalid. compiler error |
|  | } |
|  |  |
|  | 13. Difference between List<?> and List<Object> in Java? |
|  |  |
|  | List<Object> -- only list of objects is allowed. |
|  | List<?> -- any parametrized list and the raw List type is allowed |
|  |  |
|  | 14. Difference between List<String> and raw type List. |
|  |  |
|  | List<String> |
|  | Only accepts String arguments in methods like add(), set() |
|  | Returns String values in get() |
|  | Takes care casting the stored values to correct class |
|  | Performs the type checks |
|  |  |
|  | 15. Difference between List<?> and raw type List. |
|  |  |
|  | The raw type and the unbounded wildcard parameterized type have a lot in common. Both act as kind of a supertype of all instantiations of the corresponding generic type. Both are so-called reifiable types. Reifiable types can be used in instanceof expressions and as the component type of arrays, where non-reifiable types (such as concrete and bounded wildcard parameterized type) are not permitted. |
|  | In other words, the raw type and the unbounded wildcard parameterized type are semantically equivalent. The only difference is that the compiler applies stricter rules to the unbounded wildcard parameterized type than to the corresponding raw type. Certain operations performed on the raw type yield "unchecked" warnings. The same operations, when performed on the corresponding unbounded wildcard parameterized type, are rejected as errors. |
|  |  |
|  | 16. Is <T> List<? extends T> x() a useful signature? |
|  |  |
|  | From docs.oracle.com: Using a wildcard as a return type should be avoided because it forces programmers using the code to deal with wildcards. In this specific case the returning list could be thought as of read-only. But the client code would still be able to add null, to clear the list or to remove the elements from the list. Trying to add element would result in compiler errors regarding the wildcard capture errors |
|  | In this specific case one cannot be sure about which exact type of list will be returned from the method x(). Is it going to be List<T> or maybe List<S> (where S extends T)? |
|  |  |
|  | 17. What restrictions are placed on method overloading? |
|  |  |
|  | Two methods with different generics cannot overload each other e.g. this is not allowed (again because of type erasure): |
|  |  |
|  | void print(List<String> strings); |
|  | void print(List<Double> doubles); |
|  |  |
|  |  |
|  | 18. Examples of a valid generic type that cannot be expressed with the Java type system and will lead to compiler warnings. |
|  |  |
|  | from http://stackoverflow.com/questions/12254897/a-bug-in-java-type-system |
|  |  |
|  | interface IProducer<T extends Comparable<T>> { |
|  | public Map<Integer, T> getResults(); |
|  | } |
|  |  |
|  | interface IEvaluator { |
|  | public <T extends Comparable<T>> double evaluate(Map<Integer, T> results, |
|  | Map<Integer, Double> groundTruth); |
|  | } |
|  |  |
|  | List<IProducer<?>> producerImplementations = lookUpProducers(); |
|  |  |
|  | // dynamically load evaluators |
|  | List<IEvaluator> evaluatorImplementations = lookUpEvaluators(); |
|  |  |
|  | // pick a producer |
|  | IProducer<?> producer = producerImplementations.get(0); |
|  |  |
|  | // pick an evaluator |
|  | IEvaluator evaluator = evaluatorImplementations.get(0); |
|  |  |
|  | // evaluate the result against the ground truth |
|  | Map<Integer, ?> data = producer.getResults(); // this type works |
|  | double score = evaluator.evaluate(data, groundTruth); // but now this call does not |
|  |  |
|  | You cannot express a type for data in Java type system. It is just not expressible enough. |
|  | But one can still make the following call and it will work |
|  |  |
|  | double score = evaluator.evaluate(producer.getResults(), groundTruth); |
|  |  |
|  | 19. Examples where the java compiler will/will not infere the generic type? (examples of code where the compiler will infere the wrong type) |
|  |  |
|  | No idea |
|  |  |
|  | 20. What generic type information is not erased and can be retrieved from the byte code? |
|  |  |
|  | Example: |
|  |  |
|  | public class TypeReference<T> { |
|  |  |
|  | private final Type type; |
|  |  |
|  | public TypeReference() { |
|  | Type superclass = getClass().getGenericSuperclass(); |
|  | if (superclass instanceof Class) { |
|  | throw new RuntimeException("Missing type parameter."); |
|  | } |
|  | this.type = ((ParameterizedType) superclass).getActualTypeArguments()[0]; |
|  | } |
|  | } |
|  |  |
|  | You are able to get the information about the type used to parametrize this class via getActualTypeArguments (reflection) |
|  |  |
|  | 21. What does the Class Enum<E extends Enum<E>> ensure? |
|  |  |
|  | http://stackoverflow.com/questions/211143/java-enum-definition?lq=1 |
|  |  |
|  | It means that the type argument for enum has to derive from an enum which itself has the same type argument. How can this happen? By making the type argument the new type itself. So if I've got an enum called StatusCode, it would be equivalent to: |
|  |  |
|  | public class StatusCode extends Enum<StatusCode> |
|  |  |
|  | Now if you check the constraints, we've got Enum<StatusCode> - so E=StatusCode. Let's check: does E extend Enum<StatusCode>? Yes! We're okay. |
|  |  |
|  | You may well be asking yourself what the point of this is :) Well, it means that the API for Enum can refer to itself - for instance, being able to say that Enum<E> implements Comparable<E>. The base class is able to do the comparisons (in the case of enums) but it can make sure that it only compares the right kind of enums with each other. |
|  |  |
|  | 22. If the compiler erases all type parameters at compile time, why should you use generics? |
|  |  |
|  | While the compiler erases all type parameters it still performs all the checks on generic code at compile time. Additionally generics enable you to implement generic algorithms. |
|  |  |
|  | 23. Super type tokens |
|  |  |
|  | http://gafter.blogspot.ch/2006/12/super-type-tokens.html |
|  |  |
|  |  |
|  | Write yourself |
|  | ------------------------------- |
|  |  |
|  | 1. Write a program to implement LRU cache using Generics ? |
|  |  |
|  | DONE |
|  |  |
|  | 2. Write a generic method to count the number of elements in a collection that have a specific property (for example, odd integers, prime numbers, palindromes). |
|  |  |
|  | DONE |
|  |  |
|  | 3. Write a generic method to exchange the positions of two different elements in an array. |
|  |  |
|  | DONE |
|  |  |
|  | 4. Write a generic method to find the maximal element in the range [begin, end) of a list. |
|  |  |
|  | DONE |
|  |  |
|  | 5. How do you invoke the following method to find the first integer in a list that is relatively prime (gcd(a,b) = 1) to a list of specified integers? |
|  |  |
|  | public static <T> |
|  | int findFirst(List<T> list, int begin, int end, UnaryPredicate<T> p) |
|  |  |
|  |  |
|  | DONE |
|  |  |
|  |  |
|  | Problems |
|  | ------------------------------- |
|  | 1. What is the following class converted to after type erasure? |
|  |  |
|  | public class Pair<K, V> { |
|  |  |
|  | public Pair(K key, V value) { |
|  | this.key = key; |
|  | this.value = value; |
|  | } |
|  |  |
|  | public K getKey(); { return key; } |
|  | public V getValue(); { return value; } |
|  |  |
|  | public void setKey(K key) { this.key = key; } |
|  | public void setValue(V value) { this.value = value; } |
|  |  |
|  | private K key; |
|  | private V value; |
|  | } |
|  |  |
|  | Just replace both K and V with Object |
|  |  |
|  | 2. What is the following method converted to after type erasure? |
|  |  |
|  | public static <T extends Comparable<T>> |
|  | int findFirstGreaterThan(T[] at, T elem) { |
|  | // ... |
|  | } |
|  |  |
|  | it is converted to the following |
|  |  |
|  | public static Comparable |
|  | int findFirstGreaterThan(Comparable[] at, Comparable elem) { |
|  | // ... |
|  | } |
|  |  |
|  | 3. Will the following method compile? If not, why? |
|  |  |
|  | public static void print(List<? super Number> list) { |
|  | for (Number n : list) |
|  | System.out.print(n + " "); |
|  | System.out.println(); |
|  | } |
|  |  |
|  | No. ? super Number allows List<Object> to be passed as a parameter to this method. The for(Number n: list) has to be changed to for(Object n: list) |
|  | Another change that would make this method compilable -- change parameter type to List<? extends Number> |
|  |  |
|  | 4. Will the following class compile? If not, why? |
|  |  |
|  | public class Singleton<T> { |
|  |  |
|  | public static T getInstance() { |
|  | if (instance == null) |
|  | instance = new Singleton<T>(); |
|  |  |
|  | return instance; |
|  | } |
|  |  |
|  | private static T instance = null; |
|  | } |
|  |  |
|  | No, it won't. The method is static. Should have been declared public static <T> T getInstance(). The field is also static. Type parameter T is not known in static context. |
|  |  |
|  | 5. Given the following classes: |
|  |  |
|  | class Shape { /\* ... \*/ } |
|  | class Circle extends Shape { /\* ... \*/ } |
|  | class Rectangle extends Shape { /\* ... \*/ } |
|  |  |
|  | class Node<T> { /\* ... \*/ } |
|  |  |
|  | Will the following code compile? If not, why? |
|  |  |
|  | Node<Circle> nc = new Node<>(); |
|  | Node<Shape> ns = nc; |
|  |  |
|  | No. Generic types are not covariant. |
|  |  |
|  | 6. Consider this class: |
|  |  |
|  | class Node<T> implements Comparable<T> { |
|  | public int compareTo(T obj) { /\* ... \*/ } |
|  | // ... |
|  | } |
|  |  |
|  | Will the following code compile? If not, why? |
|  |  |
|  | Node<String> node = new Node<>(); |
|  | Comparable<String> comp = node; |
|  |  |
|  | Yes. Node<String> implements Comparable<String> |
|  |  |
|  | 7. Will the following class compile? If not, why? |
|  |  |
|  | public final class Algorithm { |
|  | public static T max(T x, T y) { |
|  | return x > y ? x : y; |
|  | } |
|  | } |
|  |  |
|  | Not this class won't compile |
|  | 1. If T is the type parameter then the method should have been declared like |
|  | public static <T> T max(T x, T y) |
|  | 2. Even if the method was declare correctly the binary operator ">" is not defined for classes |

* © 2021 GitHub, Inc.
* [Terms](https://docs.github.com/en/github/site-policy/github-terms-of-service)
* [Privacy](https://docs.github.com/en/github/site-policy/github-privacy-statement)
* [Security](https://github.com/security)
* [Status](https://www.githubstatus.com/)
* [Docs](https://docs.github.com/)
* [Contact GitHub](https://support.github.com/)
* [Pricing](https://github.com/pricing)
* [API](https://docs.github.com/)
* [Training](https://services.github.com/)
* [Blog](https://github.blog/)
* [About](https://github.com/about)

[**Udemy for Business**](https://business.udemy.com/request-demo/?mx_pg=blog_page_development&locale=en_US&ref=header_nav) [**Browse Udemy courses**](https://www.udemy.com/)

* [**Development**](https://blog.udemy.com/development/)

* [**IT & Software**](https://blog.udemy.com/it-software/)

* [**Data Science**](https://blog.udemy.com/data-science/)

* [**Soft Skills**](https://blog.udemy.com/soft-skills/)

* [**HR and L&D**](https://blog.udemy.com/hr-and-ld/)

* [**Business**](https://blog.udemy.com/business/)

* [**Office Productivity**](https://blog.udemy.com/office-productivity/)

* [**Marketing**](https://blog.udemy.com/marketing/)

* [**Design**](https://blog.udemy.com/design/)

Top of Form

Bottom of Form

1. [**Blog Home**](https://blog.udemy.com/)
2. [**Development**](https://blog.udemy.com/development/)
3. **Java Interview Questions: How to crack the TOP 15 questions**

**Development**

**Java Interview Questions: How to crack the TOP 15 questions**

https://secure.gravatar.com/avatar/7beacb2c62f9275d855469f01f5b1a34?s=50&d=blank&r=g

**Kasia Mikoluk**

[**Share this article**](https://blog.udemy.com/)

[](https://www.udemy.com/course/java-programming-for-beginners/?tc=blog.javainterviewquestions&couponCode=half-off-for-blog)*Updated on February 27, 2014*

“What is the difference between Throw and Throws in Java Exception Handling?”

You stare at the person asking you that question and think, I should know this! But at that moment, you can’t quite remember, and then kick yourself mentally for not preparing for your interview.

Good thing that’s just a fake scenario and you’re definitely going to be able to answer that question and more when the time comes!

The typical company hiring a Java programmer is looking for someone who can program well beyond the level taught in an introductory Java class. Interviewers ask questions that aren’t necessarily stumpers, but are meant to elicit a candidate’s deeper knowledge of the subject matter. While we would suggest you to [learn Java thoroughly](https://www.udemy.com/course/learn-to-program-with-java) before facing an interview, this article will give you a special sneak peek into the **Top 15 Java Interview Questions** that one might confront during a job interview.

Let’s start off with the easier questions and head out to the slightly tougher ones.

**1. What is JVM? Why is Java called the ‘Platform Independent Programming Language’?**

[](https://blog.udemy.com/wp-content/uploads/2013/08/htmlprojects.jpg)JVM, or the Java Virtual Machine, is an interpreter which accepts ‘Bytecode’ and executes it.

Java has been termed as a ‘Platform Independent Language’ as it primarily works on the notion of ‘compile once, run everywhere’. Here’s a sequential step establishing the Platform independence feature in Java:

* The Java Compiler outputs Non-Executable Codes called ‘Bytecode’.
* Bytecode is a highly optimized set of computer instruction which could be executed by the Java Virtual Machine (JVM).
* The translation into Bytecode makes a program easier to be executed across a wide range of platforms, since all we need is a JVM designed for that particular platform.
* JVMs for various platforms might vary in configuration, those they would all understand the same set of Bytecode, thereby making the Java Program ‘Platform Independent’.

**2. What is the Difference between JDK and JRE?**

|  |
| --- |
| When asked typical [**Java Interview Questions**](https://www.udemy.com/course/java-programming-for-beginners/?tc=blog.javainterviewquestions&couponCode=half-off-for-blog) most startup Java developers get confused with JDK and JRE. And eventually, they settle for ‘anything would do man, as long as my program runs!!’ Not quite right if you aspire to make a living and career out of Programming. |

The “JDK” is the Java Development Kit. I.e., the JDK is bundle of software that you can use to develop Java based software.

The “JRE” is the Java Runtime Environment. I.e., the JRE is an implementation of the Java Virtual Machine which actually executes Java programs.

Typically, each JDK contains one (or more) JRE’s along with the various development tools like the Java source compilers, bundling and deployment tools, debuggers, development libraries, etc.

**3. What does the ‘static’ keyword mean?**

We are sure you must be well-acquainted with the [Java Basics](https://www.udemy.com/course/java-fundamentals-i-and-ii/?tc=blog.javainterviewquestions&couponCode=half-off-for-blog). Now that we are settled with the initial concepts, let’s look into the Language specific offerings.

Static variable is associated with a class and not objects of that class. For example:

|  |
| --- |
| public class ExplainStatic {  public static String name = "Look I am a static variable";  } |

We have another class where-in we intend to access this static variable just defined.

|  |
| --- |
| public class Application {  public static void main(String[] args) {  System.out.println(ExplainStatic.name)  }  } |

We don’t create object of the class ExplainStatic to access the static variable. We directly use the class name itself: ExplainStatic.name

**Top courses in Java**

[**Develop Minecraft Plugins (Java)**](https://www.udemy.com/course/develop-minecraft-plugins-java-programming/)

Stephen King

**4.7**  (2,100)

**Bestseller**

[**Java 9 New Features In Simple Way : JShell, JPMS and More**](https://www.udemy.com/course/java-9-new-features-in-simple-way-jshell-jpms-and-more/)

DURGASOFT DURGA

**4.9**  (446)

[**The Complete Core Java Course : Learn to Code**](https://www.udemy.com/course/core-java-from-scratch/)

Dezlearn Education

**4.4**  (740)

[**Java 7 & Java 8 new features with Lambdas & Streams**](https://www.udemy.com/course/java-7-8-new-featuresenhancements-zero-to-master/)

Eazy Bytes

**5**  (38)

**Highest Rated**

[**Artificial Intelligence I: Meta-Heuristics and Games in Java**](https://www.udemy.com/course/artificial-intelligence-games-in-java/)

Holczer Balazs

**4.7**  (621)

[**Java Lambda & Streams [Examples With Selenium WebDriver]**](https://www.udemy.com/course/java-8-and-beyond-for-testers/)

Vinoth Selvaraj

**4.7**  (286)

[**Selenium WebDriver with Java -Basics to Advanced+Frameworks**](https://www.udemy.com/course/selenium-real-time-examplesinterview-questions/)

Rahul Shetty

**4.6**  (59,673)

**Bestseller**

[**Java Programming for Complete Beginners - Java 16**](https://www.udemy.com/course/java-programming-tutorial-for-beginners/)

in28Minutes Official

**4.5**  (15,658)

[**Design Patterns in Java**](https://www.udemy.com/course/design-patterns-java/)

Dmitri Nesteruk

**4.5**  (5,199)

[**Learn Core JAVA Programming - Beginner to Master**](https://www.udemy.com/course/java-se-programming/)

Abdul Bari

**4.6**  (1,751)

[**Clean Code with Java: Learn Simple Design, Refactoring & TDD**](https://www.udemy.com/course/java-clean-code-with-refactoring-and-tdd/)

in28Minutes Official

**4.5**  (252)

[**Java 8 New Features In Simple Way**](https://www.udemy.com/course/java-8-new-features-in-simple-way/)

DURGASOFT DURGA

**4.6**  (6,924)

[**Develop Minecraft Plugins (Java)**](https://www.udemy.com/course/develop-minecraft-plugins-java-programming/)

Stephen King

**4.7**  (2,100)

**Bestseller**

[**Java 9 New Features In Simple Way : JShell, JPMS and More**](https://www.udemy.com/course/java-9-new-features-in-simple-way-jshell-jpms-and-more/)

DURGASOFT DURGA

**4.9**  (446)

[**The Complete Core Java Course : Learn to Code**](https://www.udemy.com/course/core-java-from-scratch/)

Dezlearn Education

**4.4**  (740)

[**Java 7 & Java 8 new features with Lambdas & Streams**](https://www.udemy.com/course/java-7-8-new-featuresenhancements-zero-to-master/)

Eazy Bytes

**5**  (38)

**Highest Rated**

[**Artificial Intelligence I: Meta-Heuristics and Games in Java**](https://www.udemy.com/course/artificial-intelligence-games-in-java/)

Holczer Balazs

**4.7**  (621)

[**Java Lambda & Streams [Examples With Selenium WebDriver]**](https://www.udemy.com/course/java-8-and-beyond-for-testers/)

Vinoth Selvaraj

**4.7**  (286)

[**Selenium WebDriver with Java -Basics to Advanced+Frameworks**](https://www.udemy.com/course/selenium-real-time-examplesinterview-questions/)

Rahul Shetty

**4.6**  (59,673)

**Bestseller**

[**Java Programming for Complete Beginners - Java 16**](https://www.udemy.com/course/java-programming-tutorial-for-beginners/)

in28Minutes Official

**4.5**  (15,658)

[**Design Patterns in Java**](https://www.udemy.com/course/design-patterns-java/)

Dmitri Nesteruk

**4.5**  (5,199)

[**Learn Core JAVA Programming - Beginner to Master**](https://www.udemy.com/course/java-se-programming/)

Abdul Bari

**4.6**  (1,751)

[**Clean Code with Java: Learn Simple Design, Refactoring & TDD**](https://www.udemy.com/course/java-clean-code-with-refactoring-and-tdd/)

in28Minutes Official

**4.5**  (252)

[**Java 8 New Features In Simple Way**](https://www.udemy.com/course/java-8-new-features-in-simple-way/)

DURGASOFT DURGA

**4.6**  (6,924)

[**Develop Minecraft Plugins (Java)**](https://www.udemy.com/course/develop-minecraft-plugins-java-programming/)

Stephen King

**4.7**  (2,100)

**Bestseller**

[**Java 9 New Features In Simple Way : JShell, JPMS and More**](https://www.udemy.com/course/java-9-new-features-in-simple-way-jshell-jpms-and-more/)

DURGASOFT DURGA

**4.9**  (446)

[**The Complete Core Java Course : Learn to Code**](https://www.udemy.com/course/core-java-from-scratch/)

Dezlearn Education

**4.4**  (740)

[**Java 7 & Java 8 new features with Lambdas & Streams**](https://www.udemy.com/course/java-7-8-new-featuresenhancements-zero-to-master/)

Eazy Bytes

**5**  (38)

**Highest Rated**

[**Artificial Intelligence I: Meta-Heuristics and Games in Java**](https://www.udemy.com/course/artificial-intelligence-games-in-java/)

Holczer Balazs

**4.7**  (621)

Next

[**More Java Courses**](https://www.udemy.com/topic/java/)

**4. What are the Data Types supported by Java? What is Autoboxing and Unboxing?**

This is one of the most common and fundamental Java interview questions**.** This is something you should have right at your finger-tips when asked. The eight Primitive Data types supported by Java are:

* **Byte :**8-bit signed two’s complement integer. It has a minimum value of -128 and a maximum value of 127 (inclusive)
* **Short :**16-bit signed two’s complement integer. It has a minimum value of -32,768 and a maximum value of 32,767 (inclusive).
* **Int :**32-bit signed two’s complement integer. It has a minimum value of -2,147,483,648 and a maximum value of 2,147,483,647 (inclusive)
* **Long :**64-bit signed two’s complement integer. It has a minimum value of -9,223,372,036,854,775,808 and a maximum value of 9,223,372,036,854,775,807 (inclusive)
* **Float**
* **Double**

**Autoboxing**: The Java compiler brings about an automatic transformation of primitive type (int, float, double etc.) into their object equivalents or wrapper type (Integer, Float, Double,etc) for the ease of compilation.

**Unboxing**: The automatic transformation of wrapper types into their primitive equivalent is known as Unboxing.

**5. What is the difference between STRINGBUFFER and STRING?**

**String object is immutable.** i.e , the value stored in the String object cannot be changed. Consider the following code snippet:

|  |
| --- |
| String myString = “Hello”;  myString = myString + ” Guest”; |

When you print the contents of myString the output will be “Hello Guest”. Although we made use of the same object (myString), internally a new object was created in the process. That’s a performance issue.

**StringBuffer/StringBuilder objects are mutable:**StringBuffer/StringBuilder objects are mutable; we can make changes to the value stored in the object. What this effectively means is that string operations such as *append* would be more efficient if performed using StringBuffer/StringBuilder objects than String objects.

|  |
| --- |
| String str = “Be Happy With Your Salary.''  str += “Because Increments are a myth";  StringBuffer strbuf = new StringBuffer();  strbuf.append(str);  System.out.println(strbuf); |

The Output of the code snippet would be: **Be Happy With Your Salary. Because Increments are a myth.**

**6. What is Function Over-Riding and Over-Loading in Java?**

This is a very important concept in [OOP (Object Oriented Programming)](https://www.udemy.com/course/object-oriented-programming-design-hitesh-choudhary/?tc=blog.javainterviewquestions) and is a must-know for every Java Programmer.

**Over-Riding:** An override is a type of function which occurs in a class which inherits from another class. An override function “replaces” a function inherited from the base class, but does so in such a way that it is called even when an instance of its class is pretending to be a different type through polymorphism. That probably was a little over the top. The code snippet below should explain things better.

|  |
| --- |
| public class Car {  public static void main (String [] args) {  Car a = new Car();  Car b = new Ferrari(); //Car ref, but a Ferrari object  a.start(); // Runs the Car version of start()  b.start(); // Runs the Ferrari version of start()  }  }  class Car {  public void start() {  System.out.println("This is a Generic start to any Car");  }  }  class Ferrari extends Car {  public void start() {  System.out.println("Lets start the Ferrari and go out for a cool Party.");  }  } |

**Over-Loading:**Overloading is the action of defining multiple methods with the same name, but with different parameters. It is unrelated to either overriding or polymorphism. Functions in Java could be overloaded by two mechanisms ideally:

* Varying the number of arguments.
* Varying the Data Type.

|  |
| --- |
| class CalculateArea{  void Area(int length){System.out.println(length\*2);}  void Area(int length , int width){System.out.println(length\*width);}  public static void main(String args[]){  CalculateArea obj=new CalculateArea();  obj.Area(10);   // Area of a Square  obj.Area(20,20);  // Area of a Rectangle  }  } |

**7. What is Constructors, Constructor Overloading in Java and Copy-Constructor?**

Constructors form the basics of OOPs, for starters.

**Constructor:**The sole purpose of having Constructors is to create an instance of a class. They are invoked while creating an object of a class. Here are a few salient features of Java Constructors:

* Constructors can be public, private, or protected.
* If a constructor with arguments has been defined in a class, you can no longer use a default no-argument constructor – you have to write one.
* They are called only once when the class is being instantiated.
* They must have the same name as the class itself.
* They do not return a value and you do not have to specify the keyword void.
* If you do not create a constructor for the class, Java helps you by using a so called default no-argument constructor.

|  |
| --- |
| public class Boss{  String name;  Boss(String input) { //This is the constructor  name = "Our Boss is also known as : " + input;  }  public static void main(String args[]) {  Boss p1 = new Boss("Super-Man");  }  } |

**Constructor overloading:**passing different number and type of variables as arguments all of which are private variables of the class. Example snippet could be as follows:

|  |
| --- |
| public class Boss{  String name;  Boss(String input) { //This is the constructor  name = "Our Boss is also known as : " + input;  }  Boss() {  name = "Our Boss is a nice man. We don’t call him names.”;  }  public static void main(String args[]) {  Boss p1 = new Boss("Super-Man");  Boss p2 = new Boss();  }  } |

**Copy Constructor:**A copy constructor is a type of constructor which constructs the object of the class from another object of the same class. The copy constructor accepts a reference to its own class as a parameter.

**Java students also learn**

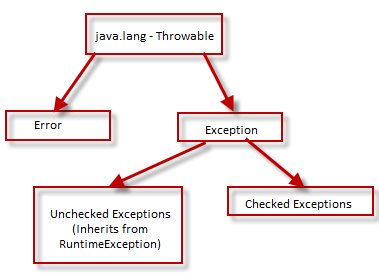
[**Spring Framework**](https://www.udemy.com/topic/spring-framework/) [**Android Development**](https://www.udemy.com/topic/android-development/) [**Selenium WebDriver**](https://www.udemy.com/topic/selenium-webdriver/) [**Spring Boot**](https://www.udemy.com/topic/spring-boot/) [**Python**](https://www.udemy.com/topic/python/) [**Object Oriented Programming**](https://www.udemy.com/topic/object-oriented-programming/) [**JavaScript**](https://www.udemy.com/topic/javascript/) [**Spring MVC**](https://www.udemy.com/topic/spring-mvc/) [**Web Development**](https://www.udemy.com/topic/web-development/) [**Hibernate**](https://www.udemy.com/topic/hibernate/) [**Programming Fundamentals**](https://www.udemy.com/topic/programming-fundamentals/) [**Java EE**](https://www.udemy.com/topic/java-ee/) [**Data Structures**](https://www.udemy.com/topic/data-structures/) [**Selenium Testing Framework**](https://www.udemy.com/topic/selenium-testing-framework/)

|  |
| --- |
| **Note: Java Doesn’t support Copy Constructor. Nevertheless folks from**[**C/C++ background**](https://www.udemy.com/course/c-2008-fundamentals/?tc=blog.javainterviewquestions&couponCode=half-off-for-blog)**often get confused when asked about Java Copy Constructors.** |

**8. What is Java Exception Handling? What is the difference between Errors, Unchecked Exception and Checked Exception?**

Anything that’s not Normal is an exception. Exceptions are the customary way in Java to indicate to a calling method that an abnormal condition has occurred.

In Java, exceptions are objects. When you throw an exception, you throw an object. You can’t throw just any object as an exception, however — only those objects whose classes descend from Throwable. Throwable serves as the base class for an entire family of classes, declared in java.lang, that your program can instantiate and throw. Here’s a hierarchical Exception class structure:

[](https://blog.udemy.com/wp-content/uploads/2013/06/java-8.png)

* An **Unchecked Exception** inherits from RuntimeException (which extends from Exception). The JVM treats RuntimeException differently as there is no requirement for the application-code to deal with them explicitly.
* A **Checked Exception** inherits from the Exception-class. The client code has to handle the checked exceptions either in a try-catch clause or has to be thrown for the Super class to catch the same. A Checked Exception thrown by a lower class (sub-class) enforces a contract on the invoking class (super-class) to catch or throw it.
* **Errors (members of the Error family)** are usually thrown for more serious problems, such as OutOfMemoryError (OOM), that may not be so easy to handle.

Exception handling needs special attention while designing large applications. So we would suggest you to spend some time [brushing up your Java skills](https://www.udemy.com/course/learn-java-from-scratch/?tc=blog.javainterviewquestions&couponCode=half-off-for-blog).

**9. What is the difference between Throw and Throws in Java Exception Handling (remember this queston?)**

**Throws:**A throws clause lists the types of exceptions that a method might throw, thereby warning the invoking method – ‘Dude. You need to handle this list of exceptions I might throw.’ Except those of type Error or RuntimeException, all other Exceptions or any of their subclasses, must be declared in the throws clause, if the method in question doesn’t implement a try…catch block. It is therefore the onus of the next-on-top method to take care of the mess.

|  |
| --- |
| public void myMethod() throws PRException  {..}  This means the super function calling the function should be equipped to handle this exception.  public void Callee()  {  try{  myMethod();  }catch(PRException ex)  {  ...handle Exception....  }  } |

**Using the Throw:** If the user wants to throw an explicit Exception, often customized, we use the Throw. The Throw clause can be used in any part of code where you feel a specific exception needs to be thrown to the calling method.

|  |
| --- |
| try{  if(age>100){throw new AgeBarException(); //Customized ExceptioN  }else{  ....}  }  }catch(AgeBarException ex){  ...handle Exception.....  } |

**10. What is the Difference between byte stream and Character streams?**

Every [Java Programmer](https://www.udemy.com/intermediate-advanced-java-programming/?tc=blog.javainterviewquestions) deals with File Operations. To generate User reports, send attachments through mails and spill out data files from Java programs. And a sound knowledge on File Operation becomes even more important while dealing with Java questions.

**byte stream :** For reading and writing binary data, byte stream is incorporated. Programs use byte streams to perform byte input and output.

* Performing InputStream operations or OutputStream operations means generally having a loop that reads the input stream and writes the output stream one byte at a time.
* You can use buffered I/O streams for an overhead reduction (overhead generated by each such request often triggers disk access, network activity, or some other operation that is relatively expensive).

**Character streams:**Character streams work with the characters rather than the byte. In Java, characters are stored by following the Unicode (allows a unique number for every character) conventions. In such kind of storage, characters become the platform independent, program independent, language independent.

**11. What are FileInputStream and FileOutputStream ? Explain with an example to read and write into files.**

**[](https://blog.udemy.com/wp-content/uploads/2014/01/shutterstock_128187719.jpg)FileInputStream :**It contains the input byte from a file and implements an input stream.

**FileOutputStream :** It uses for writing data to a file and also implements an output stream.

|  |
| --- |
| public class FileHandling {  public static void main(String [ ] args) throws IOException  {  FileInputStream inputStream = new FileInputStream ("Input.txt") ;  FileOutputStream outputStream = new FileOutputStream("Output.txt",true) ;  byte[] buffer = new byte[1024];  //For larger files we specify a buffer size which defines the chunks size for data  int bytesRead;  while ((bytesRead = inputStream.read(buffer)) != -1)  outputStream.write(buffer, 0, bytesRead);  inputStream.close() ;  outputStream.close() ;  }  } |

**12. What are FileReader and FileWriter ? Explain with an example to read and write into files.**

**FileReader :** The FileReader class makes it possible to read the contents of a file as a stream of characters. It works much like the FileInputStream, except the FileInputStream reads bytes, whereas the FileReader reads characters. The FileReader is intended to read text, in other words. One character may correspond to one or more bytes depending on the character encoding scheme.The FileReader object also lets web applications asynchronously read the contents of files (or raw data buffers) stored on the user’s computer, using File or Blob objects to specify the file or data to read.

[](https://blog.udemy.com/wp-content/uploads/2014/02/shutterstock_126226676.jpg)**FileWriter :**This class is used to write to character files. Creation of a FileWriter is not dependent on the file already existing. FileWriter will create the file before opening it for output when you create the object. Its write() methods allow you to write character(s) or Strings to a file. FileWriters are usually wrapped by higher-level Writer objects such as BufferedWriters or PrintWriters, which provide better performance and higher-level, more flexible methods to write data.

**Empower your team. Lead the industry.**

Get a subscription to a library of online courses and digital learning tools for your organization with Udemy for Business.

[**Request a demo**](https://business.udemy.com/request-demo/?mx_pg=blog_page_development&locale=en_US&ref=demo_module)

Usage of FileWriter can be explained as follows :

|  |
| --- |
| File file = new File("fileWrite2.txt");  FileWriter fw = new FileWriter(file);  for(int i=0;i<10;i++){  fw.write("Soham is Just Awesome : "+i);  fw.flush();  }  fw.close(); |

Usage of FileWriter and FileReader used in conjunction is as follows:

|  |
| --- |
| int c;  FileReader fread = new FileReader("xanadu.txt");  FileWriter fwrite = new FileWriter("characteroutput.txt");  while ((c = fread.read()) != -1)  fwrite.write(c); |

**13. What is the difference between ArrayList and LinkedList ?**

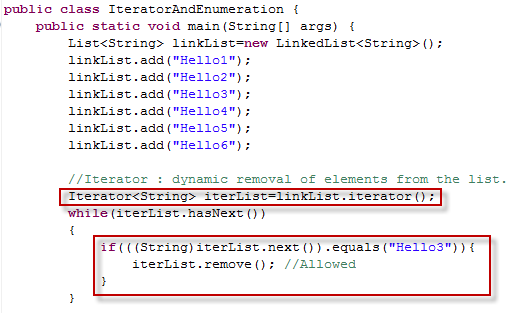
Please pay special attention as this is probably one of the most widely asked interview questions.

We aren’t going to state the properties of each in this question. What we are looking for are the differences. The prime areas where the two stand apart are as follows :

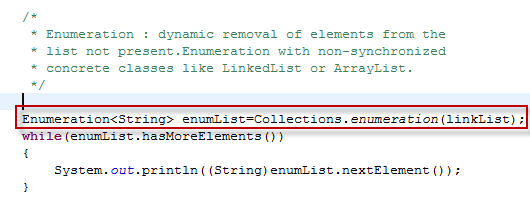
|  |  |
| --- | --- |
| **Arraylist** | **Linklist** |
| Random access. | Sequential access.The control traverses from the first node to reach the indexed node. |
| Only objects can be added. | The LinkedList is implemented using nodes linked to each other. Each node contains a previous node link, next node link, and value, which contains the actual data |

**14. Explain the difference between ITERATOR AND ENUMERATION INTERFACE with example.**

* Iterators allow the caller to remove elements from the underlying collection during the iteration with well-defined semantics.
* Iterator actually adds one method that Enumeration doesn’t have: remove ().

[](https://blog.udemy.com/wp-content/uploads/2013/06/14.png)

Whereas in case of Enumeration:

[](https://blog.udemy.com/wp-content/uploads/2013/06/14b1.png)

**15. What is the use of the ‘SimpleDateFormat’ and how can you use it to display the current system date in ‘yyyy/MM/DD HH:mm:ss’ format?**

SimpleDateFormat is one such concrete class which is widely used by Java developers for parsing and formatting of dates. This is also used to convert Dates to String and vice-versa.

|  |
| --- |
| Literally every Enterprise level Java Application invariably uses the SimpleDateFormat for handling user dates. We ofcourse aren’t expecting Java interviewees to be absolutely spectacular with the syntaxes. But a basic know-how of this class is mandatory. |

|  |
| --- |
| public class CurrentSystemDate {  public static void main(String[] args) {  SimpleDateFormat sysForm = new SimpleDateFormat("yyyy/MM/DD HH:mm:ss");  Date curdate= new Date();  System.out.println(sysForm.format(curdate));  }  } |

The best way to brush up your Java knowledge is to [open up an eclipse](https://www.udemy.com/programming-java/?tc=blog.javainterviewquestions&couponCode=half-off-for-blog) and write loads of codes and sample programs. Get into the habit of remembering syntaxes and applying the best coding standards possible.

And there’s your top 15! Best of luck on your interview.

**You might also like:**

1. [Some Advanced Java Programming](https://www.udemy.com/course/advanced-java-programming/?tc=blog.javainterviewquestions&couponCode=half-off-for-blog)

2. [Java Programming using Eclipse](https://www.udemy.com/programming-java/?tc=blog.javainterviewquestions&couponCode=half-off-for-blog)

3. [Servlets and JSPs: Creating Web Applications With Java](https://www.udemy.com/javawebtut/?tc=blog.javainterviewquestions&couponCode=half-off-for-blog)

4. [Java Essentials for Android](https://www.udemy.com/course/java-essentials-for-android/?tc=blog.javainterviewquestions&couponCode=half-off-for-blog)

*Page Last Updated: October 2015*

https://secure.gravatar.com/avatar/7beacb2c62f9275d855469f01f5b1a34?s=50&d=blank&r=g

**Kasia Mikoluk**

**Recent Articles by Kasia Mikoluk**

* [How to Use the Excel SUMPRODUCT Function](https://blog.udemy.com/excel-sumproduct/)
* [Java Projects: Ideas to Teach You Java Programming in 2021](https://blog.udemy.com/java-projects/)
* [Java vs. Python: Which Is the Best Programming Language for You?](https://blog.udemy.com/python-vs-java/)

**Recommended Articles**

[**How to Convert a Java String to Integer**](https://blog.udemy.com/java-string-to-integer/)

Udemy Editor

[**OOPs Concepts with Examples in Java**](https://blog.udemy.com/oops-concepts-with-examples/)

Udemy Editor

[**Java Xmx and Various Memory Management Options in Java**](https://blog.udemy.com/java-xmx/)

Udemy Editor

[**An Introduction To Java: Your Top 5 Questions Answered**](https://blog.udemy.com/introduction-to-java/)

Kasia Mikoluk

[**How to Convert Integers to Strings in Java**](https://blog.udemy.com/java-integer-to-string/)

Udemy Editor

[**How to Use the for each Loop in Java with Arrays**](https://blog.udemy.com/for-each-loop-java/)

Udemy Editor

[**Java vs. Python: Which Is the Best Programming Language for You?**](https://blog.udemy.com/python-vs-java/)

Tim Buchalka

[**Understanding the Java Dictionary Class**](https://blog.udemy.com/java-dictionary/)

Udemy Editor

[**Bubble Sort: Java Array Sorting with a For Loop**](https://blog.udemy.com/bubble-sort-java/)

Kasia Mikoluk

[**How to Use the Java Collections Sort Method**](https://blog.udemy.com/java-collections-sort/)

Udemy Editor

[**How to Convert a Java String to Integer**](https://blog.udemy.com/java-string-to-integer/)

Udemy Editor

[**OOPs Concepts with Examples in Java**](https://blog.udemy.com/oops-concepts-with-examples/)

Udemy Editor

[**Java Xmx and Various Memory Management Options in Java**](https://blog.udemy.com/java-xmx/)

Udemy Editor

[**An Introduction To Java: Your Top 5 Questions Answered**](https://blog.udemy.com/introduction-to-java/)

Kasia Mikoluk

[**How to Convert Integers to Strings in Java**](https://blog.udemy.com/java-integer-to-string/)

Udemy Editor

[**How to Use the for each Loop in Java with Arrays**](https://blog.udemy.com/for-each-loop-java/)

Udemy Editor

[**Java vs. Python: Which Is the Best Programming Language for You?**](https://blog.udemy.com/python-vs-java/)

Tim Buchalka

[**Understanding the Java Dictionary Class**](https://blog.udemy.com/java-dictionary/)

Udemy Editor

[**Bubble Sort: Java Array Sorting with a For Loop**](https://blog.udemy.com/bubble-sort-java/)

Kasia Mikoluk

[**How to Use the Java Collections Sort Method**](https://blog.udemy.com/java-collections-sort/)

Udemy Editor

[**How to Convert a Java String to Integer**](https://blog.udemy.com/java-string-to-integer/)

Udemy Editor

[**OOPs Concepts with Examples in Java**](https://blog.udemy.com/oops-concepts-with-examples/)

Udemy Editor

[**Java Xmx and Various Memory Management Options in Java**](https://blog.udemy.com/java-xmx/)

Udemy Editor

Next

**Teach the World Online**

Create an online video course, reach students across the globe, and earn money.

[**Teach on Udemy**](https://www.udemy.com/teaching/?ref=teach_header)

* [Udemy for Business](https://business.udemy.com/?locale=en_US&ref=footer&mx_pg=clp-unknown&mx_pg=blog_page_development&locale=en_US&ref=footer)
* [Teach on Udemy](https://www.udemy.com/teaching/?ref=teach_footer)
* [Udemy app](https://www.udemy.com/mobile/)
* [About us](https://about.udemy.com/?locale=en-us)
* [Careers](https://about.udemy.com/careers/?locale=en-us)
* [Blog](https://blog.udemy.com/)
* [Help and Support](https://support.udemy.com/hc/en-us)
* [Affiliate](https://www.udemy.com/affiliate/)
* [Sitemap](https://www.udemy.com/sitemap/)
* [Popular courses](https://www.udemy.com/popular-courses/)

[© 2021 Udemy, Inc.](https://blog.udemy.com/)

[Terms](https://www.udemy.com/terms/) [Privacy Policy and Cookie Policy](https://www.udemy.com/terms/privacy/)

⇧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [Javatpoint Logo](https://www.javatpoint.com/)  Top of Form   |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | |  |   Bottom of Form |

* [Home](https://www.javatpoint.com/)
* [Interview Questions](https://www.javatpoint.com/interview-questions-and-answers)
* [Java](https://www.javatpoint.com/java-tutorial)
* [C](https://www.javatpoint.com/c-programming-language-tutorial)
* [SQL](https://www.javatpoint.com/sql-tutorial)
* [Android](https://www.javatpoint.com/android-tutorial)
* [HTML](https://www.javatpoint.com/html-tutorial)
* [XML](https://www.javatpoint.com/xml-tutorial)
* [JavaScript](https://www.javatpoint.com/javascript-tutorial)
* [Ajax](https://www.javatpoint.com/ajax-tutorial)
* [Cloud](https://www.javatpoint.com/cloud-computing-tutorial)
* [Python](https://www.javatpoint.com/python-tutorial)
* [Web Services](https://www.javatpoint.com/web-services-tutorial)
* [Quiz](https://www.javatpoint.com/examaccess)
* [Projects](https://www.javatpoint.com/free-java-projects)
* [Comment](https://www.javatpoint.com/comment.jsp)
* [Forum](https://www.javatpoint.com/forum)
* [Training](https://www.javatpoint.com/training)

**All Interview**

[Interview Tips](https://www.javatpoint.com/interview-tips)[JOB/HR Interview](https://www.javatpoint.com/job-interview-questions)[Competency Interview](https://www.javatpoint.com/competency-based-interview-questions)[Business Analyst](https://www.javatpoint.com/business-analyst-interview-questions)[Behavioral Interview](https://www.javatpoint.com/behavioral-interview-questions)

**Company Interview**

[Company Interview](https://www.javatpoint.com/company-interview-questions-and-recruitment-process)[IBM Interview](https://www.javatpoint.com/ibm-interview-questions)[Infosys Interview](https://www.javatpoint.com/infosys-interview-questions)[Capgemini Interview](https://www.javatpoint.com/capgemini-interview-questions)[Cognizant Interview](https://www.javatpoint.com/cognizant-interview-questions)[Wipro Interview](https://www.javatpoint.com/wipro-interview-questions)[Accenture Interview](https://www.javatpoint.com/accenture-interview-questions)[iGate Interview](https://www.javatpoint.com/igate-interview-questions)[TCS Interview](https://www.javatpoint.com/tcs-interview-questions)[HCL Interview](https://www.javatpoint.com/hcl-interview-questions)[Adobe Interview](https://www.javatpoint.com/adobe-interview-questions)[Microsoft Interview](https://www.javatpoint.com/microsoft-interview-questions)[DXC Technology Interview](https://www.javatpoint.com/dxc-technology-interview-questions)[Ericsson Interview](https://www.javatpoint.com/ericsson-interview-questions)[EXL Service Interview](https://www.javatpoint.com/exl-interview-questions)[IndiaMART Interview](https://www.javatpoint.com/indiamart-interview-questions)[Intuit Interview](https://www.javatpoint.com/intuit-interview-questions)[SpaceX Interview](https://www.javatpoint.com/spacex-interview-questions)[Sapient Interview](https://www.javatpoint.com/sapient-interview-questions)[Amazon Interview](https://www.javatpoint.com/amazon-interview-questions)

**Technical Interview**

[C Interview](https://www.javatpoint.com/c-interview-questions)[C++ Interview](https://www.javatpoint.com/cpp-interview-questions)[Data Structure Interview](https://www.javatpoint.com/data-structure-interview-questions)[Linux Interview](https://www.javatpoint.com/linux-interview-questions)[Unix Interview](https://www.javatpoint.com/unix-interview-questions)[Shell Scripting](https://www.javatpoint.com/shell-scripting-interview-questions)[Networking Interview](https://www.javatpoint.com/networking-interview-questions)[CCNA Interview](https://www.javatpoint.com/ccna-interview-questions)[Android Interview](https://www.javatpoint.com/android-interview-questions)[Cloud Computing](https://www.javatpoint.com/cloud-computing-interview-questions)[Hadoop Interview](https://www.javatpoint.com/hadoop-interview-questions)[Testing/QTP Interview](https://www.javatpoint.com/software-testing-interview-questions)[Selenium Interview](https://www.javatpoint.com/selenium-interview-questions)[Web Services Interview](https://www.javatpoint.com/web-services-interview-questions)[OS Interview](https://www.javatpoint.com/operating-system-interview-questions)[Excel Interview](https://www.javatpoint.com/excel-interview-questions)[SEO Interview](https://www.javatpoint.com/seo-interview-questions)[Digital Marketing](https://www.javatpoint.com/digital-marketing-interview-questions-and-answers)[Python Interview](https://www.javatpoint.com/python-interview-questions)[Django Interview](https://www.javatpoint.com/django-interview-questions-and-answers)[Pascal Interview](https://www.javatpoint.com/pascal-interview-questions-and-answers)[Ruby Interview](https://www.javatpoint.com/ruby-interview-questions)[Ruby on Rails Interview](https://www.javatpoint.com/ruby-on-rails-interview-questions)[Memcached Interview](https://www.javatpoint.com/memcached-interview-questions-and-answers)[Go Interview](https://www.javatpoint.com/go-programming-interview-questions)[OpenStack Interview](https://www.javatpoint.com/openstack-interview-questions-and-answers)[Scala Interview](https://www.javatpoint.com/scala-interview-questions)[Control Systems](https://www.javatpoint.com/control-systems-interview-questions)[Electrical Machines](https://www.javatpoint.com/electrical-machines-interview-questions)[Power System](https://www.javatpoint.com/power-system-interview-questions)[Digital Electronics](https://www.javatpoint.com/digital-electronics-interview-questions)[Robotics Interview](https://www.javatpoint.com/robotics-interview-questions)[TypeScript Interview](https://www.javatpoint.com/typescript-interview-questions)[Swift Interview](https://www.javatpoint.com/swift-interview-questions)[Blockchain Interview](https://www.javatpoint.com/blockchain-interview-questions)[Bitcoin Interview](https://www.javatpoint.com/bitcoin-interview-questions)[AWS Interview](https://www.javatpoint.com/aws-interview-questions)[Informatica Interview](https://www.javatpoint.com/informatica-interview-questions)[QA Interview](https://www.javatpoint.com/qa-interview-questions)[React Interview](https://www.javatpoint.com/react-interview-questions)[GIT Interview](https://www.javatpoint.com/git-interview-questions)[J2EE Interview](https://www.javatpoint.com/j2ee-interview-questions)[Deep Learning](https://www.javatpoint.com/deep-learning-interview-questions)[Salesforce Interview](https://www.javatpoint.com/salesforce-interview-questions)[TestNG Interview](https://www.javatpoint.com/testng-interview-questions)[Agile Interview](https://www.javatpoint.com/agile-interview-questions)[Machine Learning](https://www.javatpoint.com/machine-learning-interview-questions)[API Testing](https://www.javatpoint.com/api-testing-interview-questions)[Algorithm](https://www.javatpoint.com/algorithm-interview-questions)[Java Design Pattern](https://www.javatpoint.com/java-design-pattern-interview-questions)[Teradata](https://www.javatpoint.com/teradata-interview-questions)[TensorFlow](https://www.javatpoint.com/tensorflow-interview-questions)[LINQ](https://www.javatpoint.com/linq-interview-questions)[Mainframe](https://www.javatpoint.com/mainframe-interview-questions)[Laravel](https://www.javatpoint.com/laravel-interview-questions)[JIRA](https://www.javatpoint.com/jira-interview-questions)[JCL](https://www.javatpoint.com/jcl-interview-questions)[Pytorch](https://www.javatpoint.com/pytorch-interview-questions)[ETL Testing](https://www.javatpoint.com/etl-testing-interview-questions)[Linked List](https://www.javatpoint.com/linked-list-interview-questions)[DataStage](https://www.javatpoint.com/datastage-interview-questions)[Tableau](https://www.javatpoint.com/tableau-interview-questions)[SAP ABAP](https://www.javatpoint.com/sap-abap-interview-questions)[DAA](https://www.javatpoint.com/daa-interview-questions)[Computer Graphics](https://www.javatpoint.com/computer-graphics-interview-questions)[Data Science](https://www.javatpoint.com/data-science-interview-questions)[Software Engineering](https://www.javatpoint.com/software-engineering-interview-questions)[Power BI](https://www.javatpoint.com/power-bi-interview-questions)[Xamarin](https://www.javatpoint.com/xamarin-interview-questions)[Ansible](https://www.javatpoint.com/ansible-interview-questions)[Data Warehouse](https://www.javatpoint.com/data-warehouse-interview-questions)[Ionic](https://www.javatpoint.com/ionic-interview-questions)[Devops](https://www.javatpoint.com/devops-interview-questions)[React Native](https://www.javatpoint.com/react-native-interview-questions)[Python Pandas](https://www.javatpoint.com/python-pandas-interview-questions)[HTTP Interview](https://www.javatpoint.com/http-interview-questions)[Kali Linux Interview](https://www.javatpoint.com/kali-linux-interview-questions)[PowerShell Interview](https://www.javatpoint.com/powershell-interview-questions)[SharePoint Interview](https://www.javatpoint.com/sharepoint-interview-questions)[Talend Interview](https://www.javatpoint.com/talend-interview-questions)[Microsoft Azure Interview](https://www.javatpoint.com/microsoft-azure-interview-questions)[R Interview](https://www.javatpoint.com/r-interview-questions)[Flutter Interview](https://www.javatpoint.com/flutter-interview-questions)[MATLAB Interview](https://www.javatpoint.com/matlab-interview-questions)[ES6 Interview](https://www.javatpoint.com/es6-interview-questions)[RPA Interview](https://www.javatpoint.com/rpa-interview-questions)[Desktop Support Interview](https://www.javatpoint.com/desktop-support-interview-questions)[Angular 8 Interview](https://www.javatpoint.com/angular-8-interview-questions)[GraphQL Interview](https://www.javatpoint.com/graphql-interview-questions)[Vue.js Interview](https://www.javatpoint.com/vue-js-interview-questions)[Automation Anywhere Interview](https://www.javatpoint.com/automation-anywhere-interview-questions)[AI Interview](https://www.javatpoint.com/artificial-intelligence-interview-questions)[UiPath Interview](https://www.javatpoint.com/uipath-interview-questions)[Blue Prism Interview](https://www.javatpoint.com/blue-prism-interview-questions)[SSIS Interview](https://www.javatpoint.com/ssis-interview-questions)[Mobile Computing Interview](https://www.javatpoint.com/mobile-computing-interview-questions)[ITIL Interview](https://www.javatpoint.com/itil-interview-questions)[SAS Interview](https://www.javatpoint.com/sas-interview-questions)[Elasticsearch Interview](https://www.javatpoint.com/elasticsearch-interview-questions)[Entity Framework Interview](https://www.javatpoint.com/entity-framework-interview-questions)[RxJS Interview](https://www.javatpoint.com/rxjs-interview-questions)[Accounting Interview](https://www.javatpoint.com/accounting-interview-questions)[Electron.js Interview](https://www.javatpoint.com/electronjs-interview-questions)[Knockout JS Interview](https://www.javatpoint.com/knockout-js-interview-questions)[Top Angular Interview](https://www.javatpoint.com/top-angular-interview-questions)[IAS Interview Question](https://www.javatpoint.com/ias-interview-questions)[Array Interview Questions in Java](https://www.javatpoint.com/array-interview-questions-in-java)[Banking Interview](https://www.javatpoint.com/banking-interview-questions)[Interview Questions for Freshers](https://www.javatpoint.com/interview-questions-for-freshers)[PowerPoint Interview](https://www.javatpoint.com/powerpoint-interview-questions)[BPO Interview](https://www.javatpoint.com/bpo-interview-questions)[Civil Engineering Interview](https://www.javatpoint.com/civil-engineering-interview-questions)[Apache Spark Interview](https://www.javatpoint.com/apache-spark-interview-questions)[Data Mining Interview](https://www.javatpoint.com/data-mining-interview-questions)[Java Support Interview](https://www.javatpoint.com/java-support-interview-questions)[Kubernetes Interview](https://www.javatpoint.com/kubernetes-interview-questions)[Web API Interview](https://www.javatpoint.com/web-api-interview-questions)[Scrum Master Interview](https://www.javatpoint.com/scrum-master-interview-questions)[System Design Interview](https://www.javatpoint.com/system-design-interview-questions)[SSB Interview Questions](https://www.javatpoint.com/ssb-interview-questions)[Teachers Interview Questions](https://www.javatpoint.com/teachers-interview-questions)[Finance Interview Questions](https://www.javatpoint.com/finance-interview-questions)[Technical Support Interview](https://www.javatpoint.com/technical-support-interview-questions)[Active Directory Interview](https://www.javatpoint.com/active-directory-interview-questions)[Appium Interview](https://www.javatpoint.com/appium-interview-questions)[Cucumber Interview](https://www.javatpoint.com/cucumber-interview-questions)[Full-Stack Developer](https://www.javatpoint.com/full-stack-developer-interview-questions-java)

**Web Interview**

[HTML Interview](https://www.javatpoint.com/html-interview-questions)[CSS Interview](https://www.javatpoint.com/css-interview-questions)[JavaScript Interview](https://www.javatpoint.com/javascript-interview-questions)[jQuery Interview](https://www.javatpoint.com/jquery-interview-questions)[AngularJS Interview](https://www.javatpoint.com/angularjs-interview-questions)[Angular Interview](https://www.javatpoint.com/angular-interview-questions)[Angular 7 Interview](https://www.javatpoint.com/angular-7-interview-questions)[Node.js Interview](https://www.javatpoint.com/node-js-interview-questions)[AJAX Interview](https://www.javatpoint.com/ajax-interview-questions)[XML Interview](https://www.javatpoint.com/xml-interview-questions)[Dojo Interview](https://www.javatpoint.com/dojo-interview-questions-and-answers)[Backbone.js Interview](https://www.javatpoint.com/backbone-js-interview-questions-and-answers)[Ember.js Interview](https://www.javatpoint.com/ember-js-interview-questions)[XHTML Interview](https://www.javatpoint.com/xhtml-interview-questions)[XSLT Interview](https://www.javatpoint.com/xslt-interview-questions)[XPath Interview](https://www.javatpoint.com/xpath-interview-questions)[XQuery Interview](https://www.javatpoint.com/xquery-interview-questions)[XForms Interview](https://www.javatpoint.com/xforms-interview-questions)[SASS Interview](https://www.javatpoint.com/sass-interview-questions-and-answers)[CoffeeScript Interview](https://www.javatpoint.com/coffeescript-interview-questions-and-answers)[LESS Interview](https://www.javatpoint.com/less-interview-questions-and-answers)[Perl Interview](https://www.javatpoint.com/perl-interview-questions)[Pure.CSS Interview](https://www.javatpoint.com/pure-css-interview-questions)[Materialize Interview](https://www.javatpoint.com/materialize-css-interview-questions)[Framework7 Interview](https://www.javatpoint.com/framework7-interview-questions)[SVG Interview](https://www.javatpoint.com/svg-interview-questions)

**PHP Interview**

[PHP Interview](https://www.javatpoint.com/php-interview-questions)[Wordpress Interview](https://www.javatpoint.com/wordpress-interview-questions)[Joomla Interview](https://www.javatpoint.com/joomla-interview-questions-and-answers)[Drupal Interview](https://www.javatpoint.com/drupal-interview-questions-and-answers)[Magento Interview](https://www.javatpoint.com/magento-interview-questions-and-answers)[CodeIgniter Interview](https://www.javatpoint.com/codeigniter-interview-questions-and-answers)[Phalcon Interview](https://www.javatpoint.com/phalcon-interview-questions)

**.Net Interview**

[.Net Interview](https://www.javatpoint.com/dot-net-interview-questions)[C# Interview](https://www.javatpoint.com/c-sharp-interview-questions)[ASP.NET Interview](https://www.javatpoint.com/asp-dot-net-interview-questions)[ADO.NET Interview](https://www.javatpoint.com/ado-dot-net-interview-questions)[WCF Interview](https://www.javatpoint.com/wcf-interview-questions)[WPF Interview](https://www.javatpoint.com/wpf-interview-questions)[Silverlight Interview](https://www.javatpoint.com/silverlight-interview-questions-and-answers)[F# Interview](https://www.javatpoint.com/fsharp-interview-questions)

**Java Interview**

[**Java 1 Interview** [open link](https://www.javatpoint.com/corejava-interview-questions)](https://www.javatpoint.com/corejava-interview-questions)[Java 2 Interview](https://www.javatpoint.com/corejava-interview-questions-2)[Exception & String](https://www.javatpoint.com/corejava-interview-questions-3)[Multithreading](https://www.javatpoint.com/java-multithreading-interview-questions)[Collections Interview](https://www.javatpoint.com/java-collections-interview-questions)[JDBC Interview](https://www.javatpoint.com/jdbc-interview-questions)[Servlet Interview](https://www.javatpoint.com/servletinterview)[JSP Interview](https://www.javatpoint.com/jspinterview)[EJB Interview](https://www.javatpoint.com/ejb-interview-questions)[Struts Interview](https://www.javatpoint.com/struts-interview-questions)[Hibernate Interview](https://www.javatpoint.com/hibernate-interview-questions)[Spring Interview](https://www.javatpoint.com/spring-interview-questions)[Spring Boot Interview](https://www.javatpoint.com/spring-boot-interview-questions)[Spring MVC Interview](https://www.javatpoint.com/spring-mvc-interview-questions)[JPA Interview](https://www.javatpoint.com/jpa-interview-questions)[Maven Interview](https://www.javatpoint.com/maven-interview-questions)[JUnit Interview](https://www.javatpoint.com/junit-interview-questions)[JSF Interview](https://www.javatpoint.com/jsf-interview-questions)[PrimeFaces Interview](https://www.javatpoint.com/primefaces-interview-questions)[RichFaces Interview](https://www.javatpoint.com/richfaces-interview-questions)[GWT Interview](https://www.javatpoint.com/gwt-interview-questions)[JDB Interview](https://www.javatpoint.com/jdb-interview-questions)[JOGL Interview](https://www.javatpoint.com/jogl-interview-questions)[jBPM Interview](https://www.javatpoint.com/jbpm-interview-questions)[Jenkins Interview](https://www.javatpoint.com/jenkins-interview-questions)[Java Constructor Interview Questions](https://www.javatpoint.com/java-constructor-interview-questions)[Java Technical Architect Interview Questions](https://www.javatpoint.com/java-technical-architect-interview-questions)

**Database Interview**

[DBMS Interview](https://www.javatpoint.com/dbms-interview-questions)[SQL Interview](https://www.javatpoint.com/sql-interview-questions)[PL/SQL Interview](https://www.javatpoint.com/pl-sql-interview-questions)[Oracle Interview](https://www.javatpoint.com/oracle-interview-questions)[MySql Interview](https://www.javatpoint.com/mysql-interview-questions)[SQL Server Interview](https://www.javatpoint.com/sql-server-interview-questions)[MongoDB Interview](https://www.javatpoint.com/mongodb-interview-questions)[Cassandra Interview](https://www.javatpoint.com/cassandra-interview-questions-and-answers)[DB2 Interview](https://www.javatpoint.com/db2-interview-questions-and-answers)[Access Interview](https://www.javatpoint.com/microsoft-access-interview-questions-and-answers)[SQLite Interview](https://www.javatpoint.com/sqlite-interview-questions)[CouchDB Interview](https://www.javatpoint.com/couchdb-interview-questions)[Neo4j Interview](https://www.javatpoint.com/neo4j-interview-questions-and-answers)[MariaDB Interview](https://www.javatpoint.com/mariadb-interview-questions)[PouchDB Interview](https://www.javatpoint.com/pouchdb-interview-questions)[Redis Interview](https://www.javatpoint.com/redis-interview-questions-and-answers)

[ADS BY ADRECOVER](https://adrecover.com/?utm_campaign=ByAdRecover&utm_source=www.javatpoint.com&utm_medium=ByAdRecover-160)

[ADVERTISEMENT BY ADRECOVER](https://adrecover.com/?utm_campaign=ByAdRecover&utm_source=www.javatpoint.com&utm_medium=ByAdRecover-728)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 300 Core Java Interview Questions | Set 1  **90% assurance of interview questions**  There is the list of 300 core Java interview questions. If there is any core Java interview question that has been asked to you, kindly post it in the ask question section. We assure that you will get here the 90% frequently asked interview questions and answers.  The answers to the Core Java interview questions are short and to the point. The core Java interview questions are categorized in Basics of Java interview questions, OOPs interview questions, String Handling interview questions, Multithreading interview questions, collection interview questions, JDBC interview questions, etc.  [**1**](https://www.javatpoint.com/corejava-interview-questions) [**2**](https://www.javatpoint.com/corejava-interview-questions-2) [**3**](https://www.javatpoint.com/java-multithreading-interview-questions) [**4**](https://www.javatpoint.com/java-collections-interview-questions) [**5**](https://www.javatpoint.com/jdbc-interview-questions)  Core Java: Basics of Java Interview Questions  1) What is Java?  [Java](https://www.javatpoint.com/java-tutorial) is the high-level, [object-oriented](https://www.javatpoint.com/java-oops-concepts), robust, secure programming language, platform-independent, high performance, Multithreaded, and portable programming language. It was developed by [**James Gosling**](https://www.javatpoint.com/james-gosling-father-of-java) in June 1991. It can also be known as the platform as it provides its own JRE and API.  2) What are the differences between C++ and Java?  The differences between [C++](https://www.javatpoint.com/cpp-tutorial) and Java are given in the following table.   |  |  |  | | --- | --- | --- | | **Comparison Index** | **C++** | **Java** | | **Platform-independent** | C++ is platform-dependent. | Java is platform-independent. | | **Mainly used for** | C++ is mainly used for system programming. | Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications. | | **Design Goal** | C++ was designed for systems and applications programming. It was an extension of [C programming language](https://www.javatpoint.com/c-programming-language-tutorial). | Java was designed and created as an interpreter for printing systems but later extended as a support network computing. It was designed with a goal of being easy to use and accessible to a broader audience. | | **Goto** | C++ supports the [goto](https://www.javatpoint.com/cpp-goto-statement) statement. | Java doesn't support the goto statement. | | **Multiple inheritance** | C++ supports multiple inheritance. | Java doesn't support multiple inheritance through class. It can be achieved by [interfaces in java](https://www.javatpoint.com/interface-in-java). | | **Operator Overloading** | C++ supports [operator overloading](https://www.javatpoint.com/cpp-overloading). | Java doesn't support operator overloading. | | **Pointers** | C++ supports [pointers](https://www.javatpoint.com/cpp-pointers). You can write pointer program in C++. | Java supports pointer internally. However, you can't write the pointer program in java. It means java has restricted pointer support in Java. | | **Compiler and Interpreter** | C++ uses compiler only. C++ is compiled and run using the compiler which converts source code into machine code so, C++ is platform dependent. | Java uses compiler and interpreter both. Java source code is converted into bytecode at compilation time. The interpreter executes this bytecode at runtime and produces output. Java is interpreted that is why it is platform independent. | | **Call by Value and Call by reference** | C++ supports both call by value and call by reference. | Java supports call by value only. There is no call by reference in java. | | **Structure and Union** | C++ supports structures and unions. | Java doesn't support structures and unions. | | **Thread Support** | C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support. | Java has built-in [thread](https://www.javatpoint.com/multithreading-in-java) support. | | **Documentation comment** | C++ doesn't support documentation comment. | Java supports documentation comment (/\*\* ... \*/) to create documentation for java source code. | | **Virtual Keyword** | C++ supports virtual keyword so that we can decide whether or not override a function. | Java has no virtual keyword. We can override all non-static methods by default. In other words, non-static methods are virtual by default. | | **unsigned right shift >>>** | C++ doesn't support >>> operator. | Java supports unsigned right shift >>> operator that fills zero at the top for the negative numbers. For positive numbers, it works same like >> operator. | | **Inheritance Tree** | C++ creates a new inheritance tree always. | Java uses a single inheritance tree always because all classes are the child of Object class in java. The object class is the root of the [inheritance](https://www.javatpoint.com/inheritance-in-java) tree in java. | | **Hardware** | C++ is nearer to hardware. | Java is not so interactive with hardware. | | **Object-oriented** | C++ is an object-oriented language. However, in C language, single root hierarchy is not possible. | Java is also an [object-oriented](https://www.javatpoint.com/java-oops-concepts) language. However, everything (except fundamental types) is an object in Java. It is a single root hierarchy as everything gets derived from java.lang.Object. |   3) List the features of Java Programming language.  There are the following features in Java Programming Language.   * **Simple:** Java is easy to learn. The syntax of Java is based on C++ which makes easier to write the program in it. * **Object-Oriented:** Java follows the object-oriented paradigm which allows us to maintain our code as the combination of different type of objects that incorporates both data and behavior. * **Portable:** Java supports read-once-write-anywhere approach. We can execute the Java program on every machine. Java program (.java) is converted to bytecode (.class) which can be easily run on every machine. * **Platform Independent:** Java is a platform independent programming language. It is different from other programming languages like C and C++ which needs a platform to be executed. Java comes with its platform on which its code is executed. Java doesn't depend upon the operating system to be executed. * **Secured:** Java is secured because it doesn't use explicit pointers. Java also provides the concept of ByteCode and Exception handling which makes it more secured. * **Robust:** Java is a strong programming language as it uses strong memory management. The concepts like Automatic garbage collection, Exception handling, etc. make it more robust. * **Architecture Neutral:** Java is architectural neutral as it is not dependent on the architecture. In C, the size of data types may vary according to the architecture (32 bit or 64 bit) which doesn't exist in Java. * **Interpreted:** Java uses the Just-in-time (JIT) interpreter along with the compiler for the program execution. * **High Performance:** Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g., C++). * **Multithreaded:** We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications, etc. * **Distributed:** Java is distributed because it facilitates users to create distributed applications in Java. RMI and EJB are used for creating distributed applications. This feature of Java makes us able to access files by calling the methods from any machine on the internet. * **Dynamic:** Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.   4) What do you understand by Java virtual machine?  [Java Virtual Machine](https://www.javatpoint.com/jvm-java-virtual-machine) is a virtual machine that enables the computer to run the Java program. JVM acts like a run-time engine which calls the main method present in the Java code. JVM is the specification which must be implemented in the computer system. The Java code is compiled by JVM to be a Bytecode which is machine independent and close to the native code.  5) What is the difference between JDK, JRE, and JVM?  JVM  JVM is an acronym for Java Virtual Machine; it is an abstract machine which provides the runtime environment in which Java bytecode can be executed. It is a specification which specifies the working of Java Virtual Machine. Its implementation has been provided by Oracle and other companies. Its implementation is known as JRE.  JVMs are available for many hardware and software platforms (so JVM is platform dependent). It is a runtime instance which is created when we run the Java class. There are three notions of the JVM: specification, implementation, and instance.  JRE  JRE stands for Java Runtime Environment. It is the implementation of JVM. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.  JDK  JDK is an acronym for Java Development Kit. It is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools. JDK is an implementation of any one of the below given Java Platforms released by Oracle Corporation:   * Standard Edition Java Platform * Enterprise Edition Java Platform * Micro Edition Java Platform   [More Details.](https://www.javatpoint.com/difference-between-jdk-jre-and-jvm)  6) How many types of memory areas are allocated by JVM?  Many types:   1. **Class(Method) Area:** Class Area stores per-class structures such as the runtime constant pool, field, method data, and the code for methods. 2. **Heap:** It is the runtime data area in which the memory is allocated to the objects 3. **Stack:** Java Stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return. Each thread has a private JVM stack, created at the same time as the thread. A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes. 4. **Program Counter Register:** PC (program counter) register contains the address of the Java virtual machine instruction currently being executed. 5. **Native Method Stack:** It contains all the native methods used in the application.   [More Details.](https://www.javatpoint.com/internal-details-of-jvm)  7) What is JIT compiler?  **Just-In-Time(JIT) compiler:** It is used to improve the performance. JIT compiles parts of the bytecode that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here the term “compiler” refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.  8) What is the platform?  A platform is the hardware or software environment in which a piece of software is executed. There are two types of platforms, software-based and hardware-based. Java provides the software-based platform.  9) What are the main differences between the Java platform and other platforms?  There are the following differences between the Java platform and other platforms.   * Java is the software-based platform whereas other platforms may be the hardware platforms or software-based platforms. * Java is executed on the top of other hardware platforms whereas other platforms can only have the hardware components.   10) What gives Java its 'write once and run anywhere' nature?  The bytecode. Java compiler converts the Java programs into the class file (Byte Code) which is the intermediate language between source code and machine code. This bytecode is not platform specific and can be executed on any computer.  11) What is classloader?  Classloader is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the classloader. There are three built-in classloaders in Java.   1. **Bootstrap ClassLoader**: This is the first classloader which is the superclass of Extension classloader. It loads the *rt.jar* file which contains all class files of Java Standard Edition like java.lang package classes, java.net package classes, java.util package classes, java.io package classes, java.sql package classes, etc. 2. **Extension ClassLoader**: This is the child classloader of Bootstrap and parent classloader of System classloader. It loads the jar files located inside *$JAVA\_HOME/jre/lib/ext* directory. 3. **System/Application ClassLoader**: This is the child classloader of Extension classloader. It loads the class files from the classpath. By default, the classpath is set to the current directory. You can change the classpath using "-cp" or "-classpath" switch. It is also known as Application classloader.   12) Is Empty .java file name a valid source file name?  Yes, Java allows to save our java file by **.java** only, we need to compile it by **javac .java** and run by **java classname** Let's take a simple example:   1. //save by .java only 2. **class** A{ 3. **public** **static** **void** main(String args[]){ 4. System.out.println("Hello java"); 5. } 6. } 7. //compile by javac .java 8. //run by     java A   compile it by **javac .java**  run it by **java A**  13) Is delete, next, main, exit or null keyword in java?  No.  14) If I don't provide any arguments on the command line, then what will the value stored in the String array passed into the main() method, empty or NULL?  It is empty, but not null.  15) What if I write static public void instead of public static void?  The program compiles and runs correctly because the order of specifiers doesn't matter in Java.  16) What is the default value of the local variables?  The local variables are not initialized to any default value, neither primitives nor object references.  17) What are the various access specifiers in Java?  In Java, access specifiers are the keywords which are used to define the access scope of the method, class, or a variable. In Java, there are four access specifiers given below.   * **Public** The classes, methods, or variables which are defined as public, can be accessed by any class or method. * **Protected** Protected can be accessed by the class of the same package, or by the sub-class of this class, or within the same class. * **Default** Default are accessible within the package only. By default, all the classes, methods, and variables are of default scope. * **Private** The private class, methods, or variables defined as private can be accessed within the class only.   18) What is the purpose of static methods and variables?  The methods or variables defined as static are shared among all the objects of the class. The static is the part of the class and not of the object. The static variables are stored in the class area, and we do not need to create the object to access such variables. Therefore, static is used in the case, where we need to define variables or methods which are common to all the objects of the class.  For example, In the class simulating the collection of the students in a college, the name of the college is the common attribute to all the students. Therefore, the college name will be defined as **static**.  19) What are the advantages of Packages in Java?  There are various advantages of defining packages in Java.   * Packages avoid the name clashes. * The Package provides easier access control. * We can also have the hidden classes that are not visible outside and used by the package. * It is easier to locate the related classes.   20) What is the output of the following Java program?   1. **class** Test 2. { 3. **public** **static** **void** main (String args[]) 4. { 5. System.out.println(10 + 20 + "Javatpoint"); 6. System.out.println("Javatpoint" + 10 + 20); 7. } 8. }   The output of the above code will be  30Javatpoint  Javatpoint1020  **Explanation**  In the first case, 10 and 20 are treated as numbers and added to be 30. Now, their sum 30 is treated as the string and concatenated with the string **Javatpoint**. Therefore, the output will be **30Javatpoint**.  In the second case, the string Javatpoint is concatenated with 10 to be the string **Javatpoint10** which will then be concatenated with 20 to be **Javatpoint1020**.  21) What is the output of the following Java program?   1. **class** Test 2. { 3. **public** **static** **void** main (String args[]) 4. { 5. System.out.println(10 \* 20 + "Javatpoint"); 6. System.out.println("Javatpoint" + 10 \* 20); 7. } 8. }   The output of the above code will be  200Javatpoint  Javatpoint200  **Explanation**  In the first case, The numbers 10 and 20 will be multiplied first and then the result 200 is treated as the string and concatenated with the string **Javatpoint** to produce the output **200Javatpoint**.  In the second case, The numbers 10 and 20 will be multiplied first to be 200 because the precedence of the multiplication is higher than addition. The result 200 will be treated as the string and concatenated with the string **Javatpoint**to produce the output as **Javatpoint200**.  22) What is the output of the following Java program?   1. **class** Test 2. { 3. **public** **static** **void** main (String args[]) 4. { 5. **for**(**int** i=0; 0; i++) 6. { 7. System.out.println("Hello Javatpoint"); 8. } 9. } 10. }   The above code will give the compile-time error because the for loop demands a boolean value in the second part and we are providing an integer value, i.e., 0.  Core Java - OOPs Concepts: Initial OOPs Interview Questions  There is given more than 50 OOPs (Object-Oriented Programming and System) interview questions. However, they have been categorized in many sections such as constructor interview questions, static interview questions, Inheritance Interview questions, Abstraction interview question, Polymorphism interview questions, etc. for better understanding.  23) What is object-oriented paradigm?  It is a programming paradigm based on objects having data and methods defined in the class to which it belongs. Object-oriented paradigm aims to incorporate the advantages of modularity and reusability. Objects are the instances of classes which interacts with one another to design applications and programs. There are the following features of the object-oriented paradigm.   * Follows the bottom-up approach in program design. * Focus on data with methods to operate upon the object's data * Includes the concept like Encapsulation and abstraction which hides the complexities from the user and show only functionality. * Implements the real-time approach like inheritance, abstraction, etc. * The examples of the object-oriented paradigm are C++, Simula, Smalltalk, Python, C#, etc.   24) What is an object?  The Object is the real-time entity having some state and behavior. In Java, Object is an instance of the class having the instance variables as the state of the object and the methods as the behavior of the object. The object of a class can be created by using the **new** keyword.  25) What is the difference between an object-oriented programming language and object-based programming language?  There are the following basic differences between the object-oriented language and object-based language.   * Object-oriented languages follow all the concepts of OOPs whereas, the object-based language doesn't follow all the concepts of OOPs like inheritance and polymorphism. * Object-oriented languages do not have the inbuilt objects whereas Object-based languages have the inbuilt objects, for example, JavaScript has window object. * Examples of object-oriented programming are Java, C#, Smalltalk, etc. whereas the examples of object-based languages are JavaScript, VBScript, etc.   26) What will be the initial value of an object reference which is defined as an instance variable?  All object references are initialized to null in Java.  Core Java - OOPs Concepts: Constructor Interview Questions  27) What is the constructor?  The constructor can be defined as the special type of method that is used to initialize the state of an object. It is invoked when the class is instantiated, and the memory is allocated for the object. Every time, an object is created using the **new** keyword, the default constructor of the class is called. The name of the constructor must be similar to the class name. The constructor must not have an explicit return type.  [More Details.](https://www.javatpoint.com/constructor)  28) How many types of constructors are used in Java?  Based on the parameters passed in the constructors, there are two types of constructors in Java.   * **Default Constructor:** default constructor is the one which does not accept any value. The default constructor is mainly used to initialize the instance variable with the default values. It can also be used for performing some useful task on object creation. A default constructor is invoked implicitly by the compiler if there is no constructor defined in the class. * **Parameterized Constructor:** The parameterized constructor is the one which can initialize the instance variables with the given values. In other words, we can say that the constructors which can accept the arguments are called parameterized constructors.   Java Constructors  29) What is the purpose of a default constructor?  The purpose of the default constructor is to assign the default value to the objects. The java compiler creates a default constructor implicitly if there is no constructor in the class.   1. **class** Student3{ 2. **int** id; 3. String name; 5. **void** display(){System.out.println(id+" "+name);} 7. **public** **static** **void** main(String args[]){ 8. Student3 s1=**new** Student3(); 9. Student3 s2=**new** Student3(); 10. s1.display(); 11. s2.display(); 12. } 13. }   [**Test it Now**](https://www.javatpoint.com/opr/test.jsp?filename=Student3)  Output:  0 null  0 null  **Explanation:** In the above class, you are not creating any constructor, so compiler provides you a default constructor. Here 0 and null values are provided by default constructor.  Java default constructor [More Details.](https://www.javatpoint.com/constructor)  30) Does constructor return any value?  **Ans:** yes, The constructor implicitly returns the current instance of the class (You can't use an explicit return type with the constructor).[More Details.](https://www.javatpoint.com/constructor)  31)Is constructor inherited?  No, The constructor is not inherited.  32) Can you make a constructor final?  No, the constructor can't be final.  33) Can we overload the constructors?  Yes, the constructors can be overloaded by changing the number of arguments accepted by the constructor or by changing the data type of the parameters. Consider the following example.   1. **class** Test 2. { 3. **int** i; 4. **public** Test(**int** k) 5. { 6. i=k; 7. } 8. **public** Test(**int** k, **int** m) 9. { 10. System.out.println("Hi I am assigning the value max(k, m) to i"); 11. **if**(k>m) 12. { 13. i=k; 14. } 15. **else** 16. { 17. i=m; 18. } 19. } 20. } 21. **public** **class** Main 22. { 23. **public** **static** **void** main (String args[]) 24. { 25. Test test1 = **new** Test(10); 26. Test test2 = **new** Test(12, 15); 27. System.out.println(test1.i); 28. System.out.println(test2.i); 29. } 30. }   In the above program, The constructor Test is overloaded with another constructor. In the first call to the constructor, The constructor with one argument is called, and i will be initialized with the value 10. However, In the second call to the constructor, The constructor with the 2 arguments is called, and i will be initialized with the value 15.  34) What do you understand by copy constructor in Java?  There is no copy constructor in java. However, we can copy the values from one object to another like copy constructor in C++.  There are many ways to copy the values of one object into another in java. They are:   * By constructor * By assigning the values of one object into another * By clone() method of Object class   In this example, we are going to copy the values of one object into another using java constructor.   1. //Java program to initialize the values from one object to another 2. **class** Student6{ 3. **int** id; 4. String name; 5. //constructor to initialize integer and string 6. Student6(**int** i,String n){ 7. id = i; 8. name = n; 9. } 10. //constructor to initialize another object 11. Student6(Student6 s){ 12. id = s.id; 13. name =s.name; 14. } 15. **void** display(){System.out.println(id+" "+name);} 17. **public** **static** **void** main(String args[]){ 18. Student6 s1 = **new** Student6(111,"Karan"); 19. Student6 s2 = **new** Student6(s1); 20. s1.display(); 21. s2.display(); 22. } 23. }   [**Test it Now**](https://www.javatpoint.com/opr/test.jsp?filename=Student6)  Output:  111 Karan  111 Karan  35) What are the differences between the constructors and methods?  There are many differences between constructors and methods. They are given below.   |  |  | | --- | --- | | **Java Constructor** | **Java Method** | | A constructor is used to initialize the state of an object. | A method is used to expose the behavior of an object. | | A constructor must not have a return type. | A method must have a return type. | | The constructor is invoked implicitly. | The method is invoked explicitly. | | The Java compiler provides a default constructor if you don't have any constructor in a class. | The method is not provided by the compiler in any case. | | The constructor name must be same as the class name. | The method name may or may not be same as class name. |   Java Constructors vs Methods  36) What is the output of the following Java program?   1. **public** **class** Test 2. { 3. Test(**int** a, **int** b) 4. { 5. System.out.println("a = "+a+" b = "+b); 6. } 7. Test(**int** a, **float** b) 8. { 9. System.out.println("a = "+a+" b = "+b); 10. } 11. **public** **static** **void** main (String args[]) 12. { 13. **byte** a = 10; 14. **byte** b = 15; 15. Test test = **new** Test(a,b); 16. } 17. }   The output of the following program is:  a = 10 b = 15  Here, the data type of the variables a and b, i.e., byte gets promoted to int, and the first parameterized constructor with the two integer parameters is called.  37) What is the output of the following Java program?   1. **class** Test 2. { 3. **int** i; 4. } 5. **public** **class** Main 6. { 7. **public** **static** **void** main (String args[]) 8. { 9. Test test = **new** Test(); 10. System.out.println(test.i); 11. } 12. }   The output of the program is 0 because the variable i is initialized to 0 internally. As we know that a default constructor is invoked implicitly if there is no constructor in the class, the variable i is initialized to 0 since there is no constructor in the class.  38) What is the output of the following Java program?   1. **class** Test 2. { 3. **int** test\_a, test\_b; 4. Test(**int** a, **int** b) 5. { 6. test\_a = a; 7. test\_b = b; 8. } 9. **public** **static** **void** main (String args[]) 10. { 11. Test test = **new** Test(); 12. System.out.println(test.test\_a+" "+test.test\_b); 13. } 14. }   There is a **compiler error** in the program because there is a call to the default constructor in the main method which is not present in the class. However, there is only one parameterized constructor in the class Test. Therefore, no default constructor is invoked by the constructor implicitly.  Core Java - OOPs Concepts: static keyword Interview Questions  39) What is the static variable?  The static variable is used to refer to the common property of all objects (that is not unique for each object), e.g., The company name of employees, college name of students, etc. Static variable gets memory only once in the class area at the time of class loading. Using a static variable makes your program more memory efficient (it saves memory). Static variable belongs to the class rather than the object.   1. //Program of static variable 3. **class** Student8{ 4. **int** rollno; 5. String name; 6. **static** String college ="ITS"; 8. Student8(**int** r,String n){ 9. rollno = r; 10. name = n; 11. } 12. **void** display (){System.out.println(rollno+" "+name+" "+college);} 14. **public** **static** **void** main(String args[]){ 15. Student8 s1 = **new** Student8(111,"Karan"); 16. Student8 s2 = **new** Student8(222,"Aryan"); 18. s1.display(); 19. s2.display(); 20. } 21. }   [**Test it Now**](https://www.javatpoint.com/opr/test.jsp?filename=Student8)  Output:111 Karan ITS  222 Aryan ITS  Static Variable [More Details.](https://www.javatpoint.com/static-keyword-in-java)  40) What is the static method?   * A static method belongs to the class rather than the object. * There is no need to create the object to call the static methods. * A static method can access and change the value of the static variable.   [More Details.](https://www.javatpoint.com/static-keyword-in-java)  41) What are the restrictions that are applied to the Java static methods?  Two main restrictions are applied to the static methods.   * The static method can not use non-static data member or call the non-static method directly. * this and super cannot be used in static context as they are non-static.   42) Why is the main method static?  Because the object is not required to call the static method. If we make the main method non-static, JVM will have to create its object first and then call main() method which will lead to the extra memory allocation.[More Details.](https://www.javatpoint.com/static-keyword-in-java)  43) Can we override the static methods?  No, we can't override static methods.  44) What is the static block?  Static block is used to initialize the static data member. It is executed before the main method, at the time of classloading.   1. **class** A2{ 2. **static**{System.out.println("static block is invoked");} 3. **public** **static** **void** main(String args[]){ 4. System.out.println("Hello main"); 5. } 6. }   [**Test it Now**](https://www.javatpoint.com/opr/test.jsp?filename=A2)  Output: static block is invoked  Hello main  [More Details.](https://www.javatpoint.com/static-keyword-in-java)  45) Can we execute a program without main() method?  Ans) No, It was possible before JDK 1.7 using the static block. Since JDK 1.7, it is not possible. [More Details.](https://www.javatpoint.com/static-keyword-in-java)  46) What if the static modifier is removed from the signature of the main method?  Program compiles. However, at runtime, It throws an error "NoSuchMethodError."  47) What is the difference between static (class) method and instance method?   |  |  | | --- | --- | | **static or class method** | **instance method** | | 1)A method that is declared as static is known as the static method. | A method that is not declared as static is known as the instance method. | | 2)We don't need to create the objects to call the static methods. | The object is required to call the instance methods. | | 3)Non-static (instance) members cannot be accessed in the static context (static method, static block, and static nested class) directly. | Static and non-static variables both can be accessed in instance methods. | | 4)For example: public static int cube(int n){ return n\*n\*n;} | For example: public void msg(){...}. |   48) Can we make constructors static?  As we know that the static context (method, block, or variable) belongs to the class, not the object. Since Constructors are invoked only when the object is created, there is no sense to make the constructors static. However, if you try to do so, the compiler will show the compiler error.  49) Can we make the abstract methods static in Java?  In Java, if we make the abstract methods static, It will become the part of the class, and we can directly call it which is unnecessary. Calling an undefined method is completely useless therefore it is not allowed.  50) Can we declare the static variables and methods in an abstract class?  Yes, we can declare static variables and methods in an abstract method. As we know that there is no requirement to make the object to access the static context, therefore, we can access the static context declared inside the abstract class by using the name of the abstract class. Consider the following example.   1. **abstract** **class** Test 2. { 3. **static** **int** i = 102; 4. **static** **void** TestMethod() 5. { 6. System.out.println("hi !! I am good !!"); 7. } 8. } 9. **public** **class** TestClass **extends** Test 10. { 11. **public** **static** **void** main (String args[]) 12. { 13. Test.TestMethod(); 14. System.out.println("i = "+Test.i); 15. } 16. }   **Output**  hi !! I am good !!  i = 102  Core Java - OOPs Concepts: Inheritance Interview Questions  51) What is **this** keyword in java?  The **this** keyword is a reference variable that refers to the current object. There are the various uses of this keyword in Java. It can be used to refer to current class properties such as instance methods, variable, constructors, etc. It can also be passed as an argument into the methods or constructors. It can also be returned from the method as the current class instance.  java this keyword [More Details.](https://www.javatpoint.com/this-keyword)  52) What are the main uses of this keyword?  There are the following uses of **this** keyword.   * **this** can be used to refer to the current class instance variable. * **this** can be used to invoke current class method (implicitly) * **this()** can be used to invoke the current class constructor. * **this** can be passed as an argument in the method call. * **this** can be passed as an argument in the constructor call. * **this** can be used to return the current class instance from the method.   53) Can we assign the reference to **this** variable?  No, this cannot be assigned to any value because it always points to the current class object and this is the final reference in Java. However, if we try to do so, the compiler error will be shown. Consider the following example.   1. **public** **class** Test 2. { 3. **public** Test() 4. { 5. **this** = **null**; 6. System.out.println("Test class constructor called"); 7. } 8. **public** **static** **void** main (String args[]) 9. { 10. Test t = **new** Test(); 11. } 12. }   **Output**  Test.java:5: error: cannot assign a value to final variable this  this = null;  ^  1 error  54) Can **this** keyword be used to refer static members?  Yes, It is possible to use this keyword to refer static members because this is just a reference variable which refers to the current class object. However, as we know that, it is unnecessary to access static variables through objects, therefore, it is not the best practice to use this to refer static members. Consider the following example.   1. **public** **class** Test 2. { 3. **static** **int** i = 10; 4. **public** Test () 5. { 6. System.out.println(**this**.i); 7. } 8. **public** **static** **void** main (String args[]) 9. { 10. Test t = **new** Test(); 11. } 12. }   **Output**  10  55) How can constructor chaining be done using this keyword?  Constructor chaining enables us to call one constructor from another constructor of the class with respect to the current class object. We can use this keyword to perform constructor chaining within the same class. Consider the following example which illustrates how can we use this keyword to achieve constructor chaining.   1. **public** **class** Employee 2. { 3. **int** id,age; 4. String name, address; 5. **public** Employee (**int** age) 6. { 7. **this**.age = age; 8. } 9. **public** Employee(**int** id, **int** age) 10. { 11. **this**(age); 12. **this**.id = id; 13. } 14. **public** Employee(**int** id, **int** age, String name, String address) 15. { 16. **this**(id, age); 17. **this**.name = name; 18. **this**.address = address; 19. } 20. **public** **static** **void** main (String args[]) 21. { 22. Employee emp = **new** Employee(105, 22, "Vikas", "Delhi"); 23. System.out.println("ID: "+emp.id+" Name:"+emp.name+" age:"+emp.age+" address: "+emp.address); 24. } 26. }   **Output**  ID: 105 Name:Vikas age:22 address: Delhi  56) What are the advantages of passing this into a method instead of the current class object itself?  As we know, that this refers to the current class object, therefore, it must be similar to the current class object. However, there can be two main advantages of passing this into a method instead of the current class object.   * this is a final variable. Therefore, this cannot be assigned to any new value whereas the current class object might not be final and can be changed. * this can be used in the synchronized block.   57) What is the Inheritance?  Inheritance is a mechanism by which one object acquires all the properties and behavior of another object of another class. It is used for Code Reusability and Method Overriding. The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also. Inheritance represents the IS-A relationship which is also known as a parent-child relationship.  There are five types of inheritance in Java.   * Single-level inheritance * Multi-level inheritance * Multiple Inheritance * Hierarchical Inheritance * Hybrid Inheritance   Multiple inheritance is not supported in Java through class.  [More Details.](https://www.javatpoint.com/inheritance-in-java)  58) Why is Inheritance used in Java?  There are various advantages of using inheritance in Java that is given below.   * Inheritance provides code reusability. The derived class does not need to redefine the method of base class unless it needs to provide the specific implementation of the method. * Runtime polymorphism cannot be achieved without using inheritance. * We can simulate the inheritance of classes with the real-time objects which makes OOPs more realistic. * Inheritance provides data hiding. The base class can hide some data from the derived class by making it private. * Method overriding cannot be achieved without inheritance. By method overriding, we can give a specific implementation of some basic method contained by the base class.   59) Which class is the superclass for all the classes?  The object class is the superclass of all other classes in Java.  60) Why is multiple inheritance not supported in java?  To reduce the complexity and simplify the language, multiple inheritance is not supported in java. Consider a scenario where A, B, and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class.  Since the compile-time errors are better than runtime errors, Java renders compile-time error if you inherit 2 classes. So whether you have the same method or different, there will be a compile time error.   1. **class** A{ 2. **void** msg(){System.out.println("Hello");} 3. } 4. **class** B{ 5. **void** msg(){System.out.println("Welcome");} 6. } 7. **class** C **extends** A,B{//suppose if it were 9. Public Static **void** main(String args[]){ 10. C obj=**new** C(); 11. obj.msg();//Now which msg() method would be invoked? 12. } 13. }   [**Test it Now**](https://www.javatpoint.com/opr/test.jsp?filename=C)  Compile Time Error  61) What is aggregation?  Aggregation can be defined as the relationship between two classes where the aggregate class contains a reference to the class it owns. Aggregation is best described as a **has-a** relationship. For example, The aggregate class Employee having various fields such as age, name, and salary also contains an object of Address class having various fields such as Address-Line 1, City, State, and pin-code. In other words, we can say that Employee (class) has an object of Address class. Consider the following example.  **Address.java**   1. **public** **class** Address { 2. String city,state,country; 4. **public** Address(String city, String state, String country) { 5. **this**.city = city; 6. **this**.state = state; 7. **this**.country = country; 8. } 10. }   **Employee.java**   1. **public** **class** Emp { 2. **int** id; 3. String name; 4. Address address; 6. **public** Emp(**int** id, String name,Address address) { 7. **this**.id = id; 8. **this**.name = name; 9. **this**.address=address; 10. } 12. **void** display(){ 13. System.out.println(id+" "+name); 14. System.out.println(address.city+" "+address.state+" "+address.country); 15. } 17. **public** **static** **void** main(String[] args) { 18. Address address1=**new** Address("gzb","UP","india"); 19. Address address2=**new** Address("gno","UP","india"); 21. Emp e=**new** Emp(111,"varun",address1); 22. Emp e2=**new** Emp(112,"arun",address2); 24. e.display(); 25. e2.display(); 27. } 28. }   **Output**  111 varun  gzb UP india  112 arun  gno UP india  62) What is composition?  Holding the reference of a class within some other class is known as composition. When an object contains the other object, if the contained object cannot exist without the existence of container object, then it is called composition. In other words, we can say that composition is the particular case of aggregation which represents a stronger relationship between two objects. Example: A class contains students. A student cannot exist without a class. There exists composition between class and students.  63) What is the difference between aggregation and composition?  Aggregation represents the weak relationship whereas composition represents the strong relationship. For example, the bike has an indicator (aggregation), but the bike has an engine (composition).  64) Why does Java not support pointers?  The pointer is a variable that refers to the memory address. They are not used in Java because they are unsafe(unsecured) and complex to understand.  65) What is super in java?  The **super** keyword in Java is a reference variable that is used to refer to the immediate parent class object. Whenever you create the instance of the subclass, an instance of the parent class is created implicitly which is referred by super reference variable. The super() is called in the class constructor implicitly by the compiler if there is no super or this.   1. **class** Animal{ 2. Animal(){System.out.println("animal is created");} 3. } 4. **class** Dog **extends** Animal{ 5. Dog(){ 6. System.out.println("dog is created"); 7. } 8. } 9. **class** TestSuper4{ 10. **public** **static** **void** main(String args[]){ 11. Dog d=**new** Dog(); 12. } 13. }   [**Test it Now**](https://www.javatpoint.com/opr/test.jsp?filename=TestSuper4)  Output:  animal is created  dog is created  [More Details.](https://www.javatpoint.com/super-keyword)  66) How can constructor chaining be done by using the super keyword?   1. **class** Person 2. { 3. String name,address; 4. **int** age; 5. **public** Person(**int** age, String name, String address) 6. { 7. **this**.age = age; 8. **this**.name = name; 9. **this**.address = address; 10. } 11. } 12. **class** Employee **extends** Person 13. { 14. **float** salary; 15. **public** Employee(**int** age, String name, String address, **float** salary) 16. { 17. **super**(age,name,address); 18. **this**.salary = salary; 19. } 20. } 21. **public** **class** Test 22. { 23. **public** **static** **void** main (String args[]) 24. { 25. Employee e = **new** Employee(22, "Mukesh", "Delhi", 90000); 26. System.out.println("Name: "+e.name+" Salary: "+e.salary+" Age: "+e.age+" Address: "+e.address); 27. } 28. }   **Output**  Name: Mukesh Salary: 90000.0 Age: 22 Address: Delhi  67) What are the main uses of the super keyword?  There are the following uses of super keyword.   * super can be used to refer to the immediate parent class instance variable. * super can be used to invoke the immediate parent class method. * super() can be used to invoke immediate parent class constructor.   68) What are the differences between this and super keyword?  There are the following differences between this and super keyword.   * The super keyword always points to the parent class contexts whereas this keyword always points to the current class context. * The super keyword is primarily used for initializing the base class variables within the derived class constructor whereas this keyword primarily used to differentiate between local and instance variables when passed in the class constructor. * The super and this must be the first statement inside constructor otherwise the compiler will throw an error.   69) What is the output of the following Java program?   1. **class** Person 2. { 3. **public** Person() 4. { 5. System.out.println("Person class constructor called"); 6. } 7. } 8. **public** **class** Employee **extends** Person 9. { 10. **public** Employee() 11. { 12. System.out.println("Employee class constructor called"); 13. } 14. **public** **static** **void** main (String args[]) 15. { 16. Employee e = **new** Employee(); 17. } 18. }   **Output**  Person class constructor called  Employee class constructor called  **Explanation**  The super() is implicitly invoked by the compiler if no super() or this() is included explicitly within the derived class constructor. Therefore, in this case, The Person class constructor is called first and then the Employee class constructor is called.  70) Can you use this() and super() both in a constructor?  No, because this() and super() must be the first statement in the class constructor.  **Example:**   1. **public** **class** Test{ 2. Test() 3. { 4. **super**(); 5. **this**(); 6. System.out.println("Test class object is created"); 7. } 8. **public** **static** **void** main(String []args){ 9. Test t = **new** Test(); 10. } 11. }   Output:  Test.java:5: error: call to this must be first statement in constructor  71)What is object cloning?  The object cloning is used to create the exact copy of an object. The clone() method of the Object class is used to clone an object. The **java.lang.Cloneable** interface must be implemented by the class whose object clone we want to create. If we don't implement Cloneable interface, clone() method generates CloneNotSupportedException.   1. **protected** Object clone() **throws** CloneNotSupportedException   [More Details.](https://www.javatpoint.com/object-cloning)  Core Java - OOPs Concepts: Method Overloading Interview Questions  72) What is method overloading?  Method overloading is the polymorphism technique which allows us to create multiple methods with the same name but different signature. We can achieve method overloading in two ways.   * By Changing the number of arguments * By Changing the data type of arguments   Method overloading increases the readability of the program. Method overloading is performed to figure out the program quickly.  [More Details.](https://www.javatpoint.com/method-overloading-in-java)  73) Why is method overloading not possible by changing the return type in java?  In Java, method overloading is not possible by changing the return type of the program due to avoid the ambiguity.   1. **class** Adder{ 2. **static** **int** add(**int** a,**int** b){**return** a+b;} 3. **static** **double** add(**int** a,**int** b){**return** a+b;} 4. } 5. **class** TestOverloading3{ 6. **public** **static** **void** main(String[] args){ 7. System.out.println(Adder.add(11,11));//ambiguity 8. }}   [**Test it Now**](https://www.javatpoint.com/opr/test.jsp?filename=TestOverloading3)  Output:  Compile Time Error: method add(int, int) is already defined in class Adder  [More Details.](https://www.javatpoint.com/method-overloading-in-java)  74) Can we overload the methods by making them static?  No, We cannot overload the methods by just applying the static keyword to them(number of parameters and types are the same). Consider the following example.   1. **public** **class** Animal 2. { 3. **void** consume(**int** a) 4. { 5. System.out.println(a+" consumed!!"); 6. } 7. **static** **void** consume(**int** a) 8. { 9. System.out.println("consumed static "+a); 10. } 11. **public** **static** **void** main (String args[]) 12. { 13. Animal a = **new** Animal(); 14. a.consume(10); 15. Animal.consume(20); 16. } 17. }   **Output**  Animal.java:7: error: method consume(int) is already defined in class Animal  static void consume(int a)  ^  Animal.java:15: error: non-static method consume(int) cannot be referenced from a static context  Animal.consume(20);  ^  2 errors  75) Can we overload the main() method?  Yes, we can have any number of main methods in a Java program by using method overloading.  [More Details.](https://www.javatpoint.com/method-overloading-in-java)  76) What is method overloading with type promotion?  By Type promotion is method overloading, we mean that one data type can be promoted to another implicitly if no exact matching is found.  Java Method Overloading with Type Promotion  As displayed in the above diagram, the byte can be promoted to short, int, long, float or double. The short datatype can be promoted to int, long, float or double. The char datatype can be promoted to int, long, float or double and so on. Consider the following example.   1. **class** OverloadingCalculation1{ 2. **void** sum(**int** a,**long** b){System.out.println(a+b);} 3. **void** sum(**int** a,**int** b,**int** c){System.out.println(a+b+c);} 5. **public** **static** **void** main(String args[]){ 6. OverloadingCalculation1 obj=**new** OverloadingCalculation1(); 7. obj.sum(20,20);//now second int literal will be promoted to long 8. obj.sum(20,20,20); 9. } 10. }   [**Test it Now**](https://www.javatpoint.com/opr/test.jsp?filename=OverloadingCalculation1)  **Output**  40  60  77) What is the output of the following Java program?   1. **class** OverloadingCalculation3{ 2. **void** sum(**int** a,**long** b){System.out.println("a method invoked");} 3. **void** sum(**long** a,**int** b){System.out.println("b method invoked");} 5. **public** **static** **void** main(String args[]){ 6. OverloadingCalculation3 obj=**new** OverloadingCalculation3(); 7. obj.sum(20,20);//now ambiguity 8. } 9. }   **Output**  OverloadingCalculation3.java:7: error: reference to sum is ambiguous  obj.sum(20,20);//now ambiguity  ^  both method sum(int,long) in OverloadingCalculation3  and method sum(long,int) in OverloadingCalculation3 match  1 error  **Explanation**  There are two methods defined with the same name, i.e., sum. The first method accepts the integer and long type whereas the second method accepts long and the integer type. The parameter passed that are a = 20, b = 20. We can not tell that which method will be called as there is no clear differentiation mentioned between integer literal and long literal. This is the case of ambiguity. Therefore, the compiler will throw an error.  Core Java - OOPs Concepts: Method Overriding Interview Questions  78) What is method overriding:  If a subclass provides a specific implementation of a method that is already provided by its parent class, it is known as Method Overriding. It is used for runtime polymorphism and to implement the interface methods.  **Rules for Method overriding**   * The method must have the same name as in the parent class. * The method must have the same signature as in the parent class. * Two classes must have an IS-A relationship between them.   [More Details.](https://www.javatpoint.com/method-overriding-in-java)  79) Can we override the static method?  No, you can't override the static method because they are the part of the class, not the object.  80) Why can we not override static method?  It is because the static method is the part of the class, and it is bound with class whereas instance method is bound with the object, and static gets memory in class area, and instance gets memory in a heap.  81) Can we override the overloaded method?  Yes.  82) Difference between method Overloading and Overriding.   |  |  | | --- | --- | | **Method Overloading** | **Method Overriding** | | 1) Method overloading increases the readability of the program. | Method overriding provides the specific implementation of the method that is already provided by its superclass. | | 2) Method overloading occurs within the class. | Method overriding occurs in two classes that have IS-A relationship between them. | | 3) In this case, the parameters must be different. | In this case, the parameters must be the same. |   83) Can we override the private methods?  No, we cannot override the private methods because the scope of private methods is limited to the class and we cannot access them outside of the class.  84) Can we change the scope of the overridden method in the subclass?  Yes, we can change the scope of the overridden method in the subclass. However, we must notice that we cannot decrease the accessibility of the method. The following point must be taken care of while changing the accessibility of the method.   * The private can be changed to protected, public, or default. * The protected can be changed to public or default. * The default can be changed to public. * The public will always remain public.   85) Can we modify the throws clause of the superclass method while overriding it in the subclass?  Yes, we can modify the throws clause of the superclass method while overriding it in the subclass. However, there are some rules which are to be followed while overriding in case of exception handling.   * If the superclass method does not declare an exception, subclass overridden method cannot declare the checked exception, but it can declare the unchecked exception. * If the superclass method declares an exception, subclass overridden method can declare same, subclass exception or no exception but cannot declare parent exception.   86) What is the output of the following Java program?   1. **class** Base 2. { 3. **void** method(**int** a) 4. { 5. System.out.println("Base class method called with integer a = "+a); 6. } 8. **void** method(**double** d) 9. { 10. System.out.println("Base class method called with double d ="+d); 11. } 12. } 14. **class** Derived **extends** Base 15. { 16. @Override 17. **void** method(**double** d) 18. { 19. System.out.println("Derived class method called with double d ="+d); 20. } 21. } 23. **public** **class** Main 24. { 25. **public** **static** **void** main(String[] args) 26. { 27. **new** Derived().method(10); 28. } 29. }   **Output**  Base class method called with integer a = 10  **Explanation**  The method() is overloaded in class Base whereas it is derived in class Derived with the double type as the parameter. In the method call, the integer is passed.  87) Can you have virtual functions in Java?  Yes, all functions in Java are virtual by default.  88) What is covariant return type?  Now, since java5, it is possible to override any method by changing the return type if the return type of the subclass overriding method is subclass type. It is known as covariant return type. The covariant return type specifies that the return type may vary in the same direction as the subclass.   1. **class** A{ 2. A get(){**return** **this**;} 3. } 5. **class** B1 **extends** A{ 6. B1 get(){**return** **this**;} 7. **void** message(){System.out.println("welcome to covariant return type");} 9. **public** **static** **void** main(String args[]){ 10. **new** B1().get().message(); 11. } 12. }   [**Test it Now**](https://www.javatpoint.com/opr/test.jsp?filename=B1)  Output: welcome to covariant return type  [More Details.](https://www.javatpoint.com/covariant-return-type)  89) What is the output of the following Java program?   1. **class** Base 2. { 3. **public** **void** baseMethod() 4. { 5. System.out.println("BaseMethod called ..."); 6. } 7. } 8. **class** Derived **extends** Base 9. { 10. **public** **void** baseMethod() 11. { 12. System.out.println("Derived method called ..."); 13. } 14. } 15. **public** **class** Test 16. { 17. **public** **static** **void** main (String args[]) 18. { 19. Base b = **new** Derived(); 20. b.baseMethod(); 21. } 22. }   **Output**  Derived method called ...  **Explanation**  The method of Base class, i.e., baseMethod() is overridden in Derived class. In Test class, the reference variable b (of type Base class) refers to the instance of the Derived class. Here, Runtime polymorphism is achieved between class Base and Derived. At compile time, the presence of method baseMethod checked in Base class, If it presence then the program compiled otherwise the compiler error will be shown. In this case, baseMethod is present in Base class; therefore, it is compiled successfully. However, at runtime, It checks whether the baseMethod has been overridden by Derived class, if so then the Derived class method is called otherwise Base class method is called. In this case, the Derived class overrides the baseMethod; therefore, the Derived class method is called.  Core Java - OOPs Concepts: final keyword Interview Questions  90) What is the final variable?  In Java, the final variable is used to restrict the user from updating it. If we initialize the final variable, we can't change its value. In other words, we can say that the final variable once assigned to a value, can never be changed after that. The final variable which is not assigned to any value can only be assigned through the class constructor.  final keyword in java   1. **class** Bike9{ 2. **final** **int** speedlimit=90;//final variable 3. **void** run(){ 4. speedlimit=400; 5. } 6. **public** **static** **void** main(String args[]){ 7. Bike9 obj=**new**  Bike9(); 8. obj.run(); 9. } 10. }//end of class   [**Test it Now**](https://www.javatpoint.com/opr/test.jsp?filename=Bike9)  Output:Compile Time Error  [More Details.](https://www.javatpoint.com/final-keyword)  91) What is the final method?  If we change any method to a final method, we can't override it. [More Details.](https://www.javatpoint.com/final-keyword)   1. **class** Bike{ 2. **final** **void** run(){System.out.println("running");} 3. } 5. **class** Honda **extends** Bike{ 6. **void** run(){System.out.println("running safely with 100kmph");} 8. **public** **static** **void** main(String args[]){ 9. Honda honda= **new** Honda(); 10. honda.run(); 11. } 12. }   [**Test it Now**](https://www.javatpoint.com/opr/test.jsp?filename=Honda)  Output:Compile Time Error  92) What is the final class?  If we make any class final, we can't inherit it into any of the subclasses.   1. **final** **class** Bike{} 3. **class** Honda1 **extends** Bike{ 4. **void** run(){System.out.println("running safely with 100kmph");} 6. **public** **static** **void** main(String args[]){ 7. Honda1 honda= **new** Honda1(); 8. honda.run(); 9. } 10. }   [**Test it Now**](https://www.javatpoint.com/opr/test.jsp?filename=Honda1)  Output:Compile Time Error  [More Details.](https://www.javatpoint.com/final-keyword)  93) What is the final blank variable?  A final variable, not initialized at the time of declaration, is known as the final blank variable. We can't initialize the final blank variable directly. Instead, we have to initialize it by using the class constructor. It is useful in the case when the user has some data which must not be changed by others, for example, PAN Number. Consider the following example:   1. **class** Student{ 2. **int** id; 3. String name; 4. **final** String PAN\_CARD\_NUMBER; 5. ... 6. }   [More Details.](https://www.javatpoint.com/final-keyword)  94) Can we initialize the final blank variable?  Yes, if it is not static, we can initialize it in the constructor. If it is static blank final variable, it can be initialized only in the static block.[More Details.](https://www.javatpoint.com/final-keyword)  95) Can you declare the main method as final?  Yes, We can declare the main method as public static final void main(String[] args){}.  96) What is the output of the following Java program?   1. **class** Main { 2. **public** **static** **void** main(String args[]){ 3. **final** **int** i; 4. i = 20; 5. System.out.println(i); 6. } 7. }   **Output**  20  **Explanation**  Since i is the blank final variable. It can be initialized only once. We have initialized it to 20. Therefore, 20 will be printed.  97) What is the output of the following Java program?   1. **class** Base 2. { 3. **protected** **final** **void** getInfo() 4. { 5. System.out.println("method of Base class"); 6. } 7. } 9. **public** **class** Derived **extends** Base 10. { 11. **protected** **final** **void** getInfo() 12. { 13. System.out.println("method of Derived class"); 14. } 15. **public** **static** **void** main(String[] args) 16. { 17. Base obj = **new** Base(); 18. obj.getInfo(); 19. } 20. }   **Output**  Derived.java:11: error: getInfo() in Derived cannot override getInfo() in Base  protected final void getInfo()  ^  overridden method is final  1 error  **Explanation**  The getDetails() method is final; therefore it can not be overridden in the subclass.  98) Can we declare a constructor as final?  The constructor can never be declared as final because it is never inherited. Constructors are not ordinary methods; therefore, there is no sense to declare constructors as final. However, if you try to do so, The compiler will throw an error.  99) Can we declare an interface as final?  No, we cannot declare an interface as final because the interface must be implemented by some class to provide its definition. Therefore, there is no sense to make an interface final. However, if you try to do so, the compiler will show an error.  100) What is the difference between the final method and abstract method?  The main difference between the final method and abstract method is that the abstract method cannot be final as we need to override them in the subclass to give its definition.  Next Topic[Java interview Questions part 2](https://www.javatpoint.com/corejava-interview-questions-2)  [Next →](https://www.javatpoint.com/corejava-interview-questions-2)  [**1**](https://www.javatpoint.com/corejava-interview-questions) [**2**](https://www.javatpoint.com/corejava-interview-questions-2) [**3**](https://www.javatpoint.com/java-multithreading-interview-questions) [**4**](https://www.javatpoint.com/java-collections-interview-questions) [**5**](https://www.javatpoint.com/jdbc-interview-questions)   |  |  | | --- | --- | | [**Java Basics Interview Questions**](https://www.javatpoint.com/corejava-interview-questions#corebasicsinterview) | [**Java OOPs Interview Questions**](https://www.javatpoint.com/corejava-interview-questions#oopsinterview) | | [**Java Multithreading Interview Questions**](https://www.javatpoint.com/java-multithreading-interview-questions) | [**Java String & Exception Interview Questions**](https://www.javatpoint.com/corejava-interview-questions-3) | | [**Java Collection Interview Questions**](https://www.javatpoint.com/java-collections-interview-questions) | [**JDBC Interview Questions**](https://www.javatpoint.com/jdbc-interview-questions) | | [**Servlet Interview Questions**](https://www.javatpoint.com/servletinterview) | [**JSP Interview Questions**](https://www.javatpoint.com/jspinterview) | | [**Spring Interview Questions**](https://www.javatpoint.com/spring-interview-questions) | [**Hibernate Interview Questions**](https://www.javatpoint.com/hibernate-interview-questions) | | [**PL/SQL Interview Questions**](https://www.javatpoint.com/pl-sql-interview-questions) | [**SQL Interview Questions**](https://www.javatpoint.com/sql-interview-questions) | | [**Oracle Interview Questions**](https://www.javatpoint.com/oracle-interview-questions) | [**Android Interview Questions**](https://www.javatpoint.com/android-interview-questions) | | [**SQL Server Interview Questions**](https://www.javatpoint.com/sql-server-interview-questions) | [**MySQL Interview Questions**](https://www.javatpoint.com/mysql-interview-questions) |     [ADVERTISEMENT BY ADRECOVER](https://adrecover.com/?utm_campaign=ByAdRecover&utm_source=www.javatpoint.com&utm_medium=ByAdRecover-336) |

**Javatpoint Services**

JavaTpoint offers too many high quality services. Mail us on hr@javatpoint.com, to get more information about given services.

* Website Designing
* Website Development
* Java Development
* PHP Development
* WordPress
* Graphic Designing
* Logo
* Digital Marketing
* On Page and Off Page SEO
* PPC
* Content Development
* Corporate Training
* Classroom and Online Training
* Data Entry

**Training For College Campus**

JavaTpoint offers college campus training on Core Java, Advance Java, .Net, Android, Hadoop, PHP, Web Technology and Python. Please mail your requirement at hr@javatpoint.com.  
Duration: 1 week to 2 week

Like/Subscribe us for latest updates or newsletter[RSS Feed](https://feeds.feedburner.com/javatpointsonoo) [Subscribe to Get Email Alerts](https://feedburner.google.com/fb/a/mailverify?uri=javatpointsonoo) [Facebook Page](https://www.facebook.com/javatpoint) [Twitter Page](https://twitter.com/pagejavatpoint) [YouTube](https://www.youtube.com/channel/UCUnYvQVCrJoFWZhKK3O2xLg) [Blog Page](https://javatpoint.blogspot.com/)

**LEARN TUTORIALS**

[Learn Java](https://www.javatpoint.com/java-tutorial)[Learn Data Structures](https://www.javatpoint.com/data-structure-tutorial)[Learn C Programming](https://www.javatpoint.com/c-programming-language-tutorial)[Learn C++ Tutorial](https://www.javatpoint.com/cpp-tutorial)[Learn C# Tutorial](https://www.javatpoint.com/c-sharp-tutorial)[Learn PHP Tutorial](https://www.javatpoint.com/php-tutorial)[Learn HTML Tutorial](https://www.javatpoint.com/html-tutorial)[Learn JavaScript Tutorial](https://www.javatpoint.com/javascript-tutorial)[Learn jQuery Tutorial](https://www.javatpoint.com/jquery-tutorial)[Learn Spring Tutorial](https://www.javatpoint.com/spring-tutorial)

**OUR WEBSITES**

[Javatpoint.com](https://www.javatpoint.com/)[Hindi100.com](https://www.hindi100.com/)[Lyricsia.com](https://www.lyricsia.com/)[Quoteperson.com](https://www.quoteperson.com/)[Jobandplacement.com](https://www.jobandplacement.com/)

**OUR SERVICES**

Website Development

Android Development

Website Designing

Digital Marketing

Summer Training

Industrial Training

College Campus Training

**CONTACT**

Address: G-13, 2nd Floor, Sec-3

Noida, UP, 201301, India

Contact No: 0120-4256464, 9990449935

[Contact Us](https://www.javatpoint.com/contact-us)[Subscribe Us](https://www.javatpoint.com/subscribe.jsp)[Privacy Policy](https://www.javatpoint.com/privacy-policy)[Sitemap](https://www.javatpoint.com/sitemap.xml)  
[About Me](https://www.javatpoint.com/sonoo-jaiswal)

© Copyright 2011-2018 www.javatpoint.com. All rights reserved. Developed by JavaTpoint.

⇧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [Javatpoint Logo](https://www.javatpoint.com/)  Top of Form   |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | |  |   Bottom of Form |

* [Home](https://www.javatpoint.com/)
* [Interview Questions](https://www.javatpoint.com/interview-questions-and-answers)
* [Java](https://www.javatpoint.com/java-tutorial)
* [C](https://www.javatpoint.com/c-programming-language-tutorial)
* [SQL](https://www.javatpoint.com/sql-tutorial)
* [Android](https://www.javatpoint.com/android-tutorial)
* [HTML](https://www.javatpoint.com/html-tutorial)
* [XML](https://www.javatpoint.com/xml-tutorial)
* [JavaScript](https://www.javatpoint.com/javascript-tutorial)
* [Ajax](https://www.javatpoint.com/ajax-tutorial)
* [Cloud](https://www.javatpoint.com/cloud-computing-tutorial)
* [Python](https://www.javatpoint.com/python-tutorial)
* [Web Services](https://www.javatpoint.com/web-services-tutorial)
* [Quiz](https://www.javatpoint.com/examaccess)
* [Projects](https://www.javatpoint.com/free-java-projects)
* [Comment](https://www.javatpoint.com/comment.jsp)
* [Forum](https://www.javatpoint.com/forum)
* [Training](https://www.javatpoint.com/training)

**All Interview**

[Interview Tips](https://www.javatpoint.com/interview-tips)[JOB/HR Interview](https://www.javatpoint.com/job-interview-questions)[Competency Interview](https://www.javatpoint.com/competency-based-interview-questions)[Business Analyst](https://www.javatpoint.com/business-analyst-interview-questions)[Behavioral Interview](https://www.javatpoint.com/behavioral-interview-questions)

**Company Interview**

[Company Interview](https://www.javatpoint.com/company-interview-questions-and-recruitment-process)[IBM Interview](https://www.javatpoint.com/ibm-interview-questions)[Infosys Interview](https://www.javatpoint.com/infosys-interview-questions)[Capgemini Interview](https://www.javatpoint.com/capgemini-interview-questions)[Cognizant Interview](https://www.javatpoint.com/cognizant-interview-questions)[Wipro Interview](https://www.javatpoint.com/wipro-interview-questions)[Accenture Interview](https://www.javatpoint.com/accenture-interview-questions)[iGate Interview](https://www.javatpoint.com/igate-interview-questions)[TCS Interview](https://www.javatpoint.com/tcs-interview-questions)[HCL Interview](https://www.javatpoint.com/hcl-interview-questions)[Adobe Interview](https://www.javatpoint.com/adobe-interview-questions)[Microsoft Interview](https://www.javatpoint.com/microsoft-interview-questions)[DXC Technology Interview](https://www.javatpoint.com/dxc-technology-interview-questions)[Ericsson Interview](https://www.javatpoint.com/ericsson-interview-questions)[EXL Service Interview](https://www.javatpoint.com/exl-interview-questions)[IndiaMART Interview](https://www.javatpoint.com/indiamart-interview-questions)[Intuit Interview](https://www.javatpoint.com/intuit-interview-questions)[SpaceX Interview](https://www.javatpoint.com/spacex-interview-questions)[Sapient Interview](https://www.javatpoint.com/sapient-interview-questions)[Amazon Interview](https://www.javatpoint.com/amazon-interview-questions)

**Technical Interview**

[C Interview](https://www.javatpoint.com/c-interview-questions)[C++ Interview](https://www.javatpoint.com/cpp-interview-questions)[Data Structure Interview](https://www.javatpoint.com/data-structure-interview-questions)[Linux Interview](https://www.javatpoint.com/linux-interview-questions)[Unix Interview](https://www.javatpoint.com/unix-interview-questions)[Shell Scripting](https://www.javatpoint.com/shell-scripting-interview-questions)[Networking Interview](https://www.javatpoint.com/networking-interview-questions)[CCNA Interview](https://www.javatpoint.com/ccna-interview-questions)[Android Interview](https://www.javatpoint.com/android-interview-questions)[Cloud Computing](https://www.javatpoint.com/cloud-computing-interview-questions)[Hadoop Interview](https://www.javatpoint.com/hadoop-interview-questions)[Testing/QTP Interview](https://www.javatpoint.com/software-testing-interview-questions)[Selenium Interview](https://www.javatpoint.com/selenium-interview-questions)[Web Services Interview](https://www.javatpoint.com/web-services-interview-questions)[OS Interview](https://www.javatpoint.com/operating-system-interview-questions)[Excel Interview](https://www.javatpoint.com/excel-interview-questions)[SEO Interview](https://www.javatpoint.com/seo-interview-questions)[Digital Marketing](https://www.javatpoint.com/digital-marketing-interview-questions-and-answers)[Python Interview](https://www.javatpoint.com/python-interview-questions)[Django Interview](https://www.javatpoint.com/django-interview-questions-and-answers)[Pascal Interview](https://www.javatpoint.com/pascal-interview-questions-and-answers)[Ruby Interview](https://www.javatpoint.com/ruby-interview-questions)[Ruby on Rails Interview](https://www.javatpoint.com/ruby-on-rails-interview-questions)[Memcached Interview](https://www.javatpoint.com/memcached-interview-questions-and-answers)[Go Interview](https://www.javatpoint.com/go-programming-interview-questions)[OpenStack Interview](https://www.javatpoint.com/openstack-interview-questions-and-answers)[Scala Interview](https://www.javatpoint.com/scala-interview-questions)[Control Systems](https://www.javatpoint.com/control-systems-interview-questions)[Electrical Machines](https://www.javatpoint.com/electrical-machines-interview-questions)[Power System](https://www.javatpoint.com/power-system-interview-questions)[Digital Electronics](https://www.javatpoint.com/digital-electronics-interview-questions)[Robotics Interview](https://www.javatpoint.com/robotics-interview-questions)[TypeScript Interview](https://www.javatpoint.com/typescript-interview-questions)[Swift Interview](https://www.javatpoint.com/swift-interview-questions)[Blockchain Interview](https://www.javatpoint.com/blockchain-interview-questions)[Bitcoin Interview](https://www.javatpoint.com/bitcoin-interview-questions)[AWS Interview](https://www.javatpoint.com/aws-interview-questions)[Informatica Interview](https://www.javatpoint.com/informatica-interview-questions)[QA Interview](https://www.javatpoint.com/qa-interview-questions)[React Interview](https://www.javatpoint.com/react-interview-questions)[GIT Interview](https://www.javatpoint.com/git-interview-questions)[J2EE Interview](https://www.javatpoint.com/j2ee-interview-questions)[Deep Learning](https://www.javatpoint.com/deep-learning-interview-questions)[Salesforce Interview](https://www.javatpoint.com/salesforce-interview-questions)[TestNG Interview](https://www.javatpoint.com/testng-interview-questions)[Agile Interview](https://www.javatpoint.com/agile-interview-questions)[Machine Learning](https://www.javatpoint.com/machine-learning-interview-questions)[API Testing](https://www.javatpoint.com/api-testing-interview-questions)[Algorithm](https://www.javatpoint.com/algorithm-interview-questions)[Java Design Pattern](https://www.javatpoint.com/java-design-pattern-interview-questions)[Teradata](https://www.javatpoint.com/teradata-interview-questions)[TensorFlow](https://www.javatpoint.com/tensorflow-interview-questions)[LINQ](https://www.javatpoint.com/linq-interview-questions)[Mainframe](https://www.javatpoint.com/mainframe-interview-questions)[Laravel](https://www.javatpoint.com/laravel-interview-questions)[JIRA](https://www.javatpoint.com/jira-interview-questions)[JCL](https://www.javatpoint.com/jcl-interview-questions)[Pytorch](https://www.javatpoint.com/pytorch-interview-questions)[ETL Testing](https://www.javatpoint.com/etl-testing-interview-questions)[Linked List](https://www.javatpoint.com/linked-list-interview-questions)[DataStage](https://www.javatpoint.com/datastage-interview-questions)[Tableau](https://www.javatpoint.com/tableau-interview-questions)[SAP ABAP](https://www.javatpoint.com/sap-abap-interview-questions)[DAA](https://www.javatpoint.com/daa-interview-questions)[Computer Graphics](https://www.javatpoint.com/computer-graphics-interview-questions)[Data Science](https://www.javatpoint.com/data-science-interview-questions)[Software Engineering](https://www.javatpoint.com/software-engineering-interview-questions)[Power BI](https://www.javatpoint.com/power-bi-interview-questions)[Xamarin](https://www.javatpoint.com/xamarin-interview-questions)[Ansible](https://www.javatpoint.com/ansible-interview-questions)[Data Warehouse](https://www.javatpoint.com/data-warehouse-interview-questions)[Ionic](https://www.javatpoint.com/ionic-interview-questions)[Devops](https://www.javatpoint.com/devops-interview-questions)[React Native](https://www.javatpoint.com/react-native-interview-questions)[Python Pandas](https://www.javatpoint.com/python-pandas-interview-questions)[HTTP Interview](https://www.javatpoint.com/http-interview-questions)[Kali Linux Interview](https://www.javatpoint.com/kali-linux-interview-questions)[PowerShell Interview](https://www.javatpoint.com/powershell-interview-questions)[SharePoint Interview](https://www.javatpoint.com/sharepoint-interview-questions)[Talend Interview](https://www.javatpoint.com/talend-interview-questions)[Microsoft Azure Interview](https://www.javatpoint.com/microsoft-azure-interview-questions)[R Interview](https://www.javatpoint.com/r-interview-questions)[Flutter Interview](https://www.javatpoint.com/flutter-interview-questions)[MATLAB Interview](https://www.javatpoint.com/matlab-interview-questions)[ES6 Interview](https://www.javatpoint.com/es6-interview-questions)[RPA Interview](https://www.javatpoint.com/rpa-interview-questions)[Desktop Support Interview](https://www.javatpoint.com/desktop-support-interview-questions)[Angular 8 Interview](https://www.javatpoint.com/angular-8-interview-questions)[GraphQL Interview](https://www.javatpoint.com/graphql-interview-questions)[Vue.js Interview](https://www.javatpoint.com/vue-js-interview-questions)[Automation Anywhere Interview](https://www.javatpoint.com/automation-anywhere-interview-questions)[AI Interview](https://www.javatpoint.com/artificial-intelligence-interview-questions)[UiPath Interview](https://www.javatpoint.com/uipath-interview-questions)[Blue Prism Interview](https://www.javatpoint.com/blue-prism-interview-questions)[SSIS Interview](https://www.javatpoint.com/ssis-interview-questions)[Mobile Computing Interview](https://www.javatpoint.com/mobile-computing-interview-questions)[ITIL Interview](https://www.javatpoint.com/itil-interview-questions)[SAS Interview](https://www.javatpoint.com/sas-interview-questions)[Elasticsearch Interview](https://www.javatpoint.com/elasticsearch-interview-questions)[Entity Framework Interview](https://www.javatpoint.com/entity-framework-interview-questions)[RxJS Interview](https://www.javatpoint.com/rxjs-interview-questions)[Accounting Interview](https://www.javatpoint.com/accounting-interview-questions)[Electron.js Interview](https://www.javatpoint.com/electronjs-interview-questions)[Knockout JS Interview](https://www.javatpoint.com/knockout-js-interview-questions)[Top Angular Interview](https://www.javatpoint.com/top-angular-interview-questions)[IAS Interview Question](https://www.javatpoint.com/ias-interview-questions)[Array Interview Questions in Java](https://www.javatpoint.com/array-interview-questions-in-java)[Banking Interview](https://www.javatpoint.com/banking-interview-questions)[Interview Questions for Freshers](https://www.javatpoint.com/interview-questions-for-freshers)[PowerPoint Interview](https://www.javatpoint.com/powerpoint-interview-questions)[BPO Interview](https://www.javatpoint.com/bpo-interview-questions)[Civil Engineering Interview](https://www.javatpoint.com/civil-engineering-interview-questions)[Apache Spark Interview](https://www.javatpoint.com/apache-spark-interview-questions)[Data Mining Interview](https://www.javatpoint.com/data-mining-interview-questions)[Java Support Interview](https://www.javatpoint.com/java-support-interview-questions)[Kubernetes Interview](https://www.javatpoint.com/kubernetes-interview-questions)[Web API Interview](https://www.javatpoint.com/web-api-interview-questions)[Scrum Master Interview](https://www.javatpoint.com/scrum-master-interview-questions)[System Design Interview](https://www.javatpoint.com/system-design-interview-questions)[SSB Interview Questions](https://www.javatpoint.com/ssb-interview-questions)[Teachers Interview Questions](https://www.javatpoint.com/teachers-interview-questions)[Finance Interview Questions](https://www.javatpoint.com/finance-interview-questions)[Technical Support Interview](https://www.javatpoint.com/technical-support-interview-questions)[Active Directory Interview](https://www.javatpoint.com/active-directory-interview-questions)[Appium Interview](https://www.javatpoint.com/appium-interview-questions)[Cucumber Interview](https://www.javatpoint.com/cucumber-interview-questions)[Full-Stack Developer](https://www.javatpoint.com/full-stack-developer-interview-questions-java)

**Web Interview**

[HTML Interview](https://www.javatpoint.com/html-interview-questions)[CSS Interview](https://www.javatpoint.com/css-interview-questions)[JavaScript Interview](https://www.javatpoint.com/javascript-interview-questions)[jQuery Interview](https://www.javatpoint.com/jquery-interview-questions)[AngularJS Interview](https://www.javatpoint.com/angularjs-interview-questions)[Angular Interview](https://www.javatpoint.com/angular-interview-questions)[Angular 7 Interview](https://www.javatpoint.com/angular-7-interview-questions)[Node.js Interview](https://www.javatpoint.com/node-js-interview-questions)[AJAX Interview](https://www.javatpoint.com/ajax-interview-questions)[XML Interview](https://www.javatpoint.com/xml-interview-questions)[Dojo Interview](https://www.javatpoint.com/dojo-interview-questions-and-answers)[Backbone.js Interview](https://www.javatpoint.com/backbone-js-interview-questions-and-answers)[Ember.js Interview](https://www.javatpoint.com/ember-js-interview-questions)[XHTML Interview](https://www.javatpoint.com/xhtml-interview-questions)[XSLT Interview](https://www.javatpoint.com/xslt-interview-questions)[XPath Interview](https://www.javatpoint.com/xpath-interview-questions)[XQuery Interview](https://www.javatpoint.com/xquery-interview-questions)[XForms Interview](https://www.javatpoint.com/xforms-interview-questions)[SASS Interview](https://www.javatpoint.com/sass-interview-questions-and-answers)[CoffeeScript Interview](https://www.javatpoint.com/coffeescript-interview-questions-and-answers)[LESS Interview](https://www.javatpoint.com/less-interview-questions-and-answers)[Perl Interview](https://www.javatpoint.com/perl-interview-questions)[Pure.CSS Interview](https://www.javatpoint.com/pure-css-interview-questions)[Materialize Interview](https://www.javatpoint.com/materialize-css-interview-questions)[Framework7 Interview](https://www.javatpoint.com/framework7-interview-questions)[SVG Interview](https://www.javatpoint.com/svg-interview-questions)

**PHP Interview**

[PHP Interview](https://www.javatpoint.com/php-interview-questions)[Wordpress Interview](https://www.javatpoint.com/wordpress-interview-questions)[Joomla Interview](https://www.javatpoint.com/joomla-interview-questions-and-answers)[Drupal Interview](https://www.javatpoint.com/drupal-interview-questions-and-answers)[Magento Interview](https://www.javatpoint.com/magento-interview-questions-and-answers)[CodeIgniter Interview](https://www.javatpoint.com/codeigniter-interview-questions-and-answers)[Phalcon Interview](https://www.javatpoint.com/phalcon-interview-questions)

**.Net Interview**

[.Net Interview](https://www.javatpoint.com/dot-net-interview-questions)[C# Interview](https://www.javatpoint.com/c-sharp-interview-questions)[ASP.NET Interview](https://www.javatpoint.com/asp-dot-net-interview-questions)[ADO.NET Interview](https://www.javatpoint.com/ado-dot-net-interview-questions)[WCF Interview](https://www.javatpoint.com/wcf-interview-questions)[WPF Interview](https://www.javatpoint.com/wpf-interview-questions)[Silverlight Interview](https://www.javatpoint.com/silverlight-interview-questions-and-answers)[F# Interview](https://www.javatpoint.com/fsharp-interview-questions)

**Java Interview**

[Java 1 Interview](https://www.javatpoint.com/corejava-interview-questions)[**Java 2 Interview** [open link](https://www.javatpoint.com/corejava-interview-questions-2)](https://www.javatpoint.com/corejava-interview-questions-2)[Exception & String](https://www.javatpoint.com/corejava-interview-questions-3)[Multithreading](https://www.javatpoint.com/java-multithreading-interview-questions)[Collections Interview](https://www.javatpoint.com/java-collections-interview-questions)[JDBC Interview](https://www.javatpoint.com/jdbc-interview-questions)[Servlet Interview](https://www.javatpoint.com/servletinterview)[JSP Interview](https://www.javatpoint.com/jspinterview)[EJB Interview](https://www.javatpoint.com/ejb-interview-questions)[Struts Interview](https://www.javatpoint.com/struts-interview-questions)[Hibernate Interview](https://www.javatpoint.com/hibernate-interview-questions)[Spring Interview](https://www.javatpoint.com/spring-interview-questions)[Spring Boot Interview](https://www.javatpoint.com/spring-boot-interview-questions)[Spring MVC Interview](https://www.javatpoint.com/spring-mvc-interview-questions)[JPA Interview](https://www.javatpoint.com/jpa-interview-questions)[Maven Interview](https://www.javatpoint.com/maven-interview-questions)[JUnit Interview](https://www.javatpoint.com/junit-interview-questions)[JSF Interview](https://www.javatpoint.com/jsf-interview-questions)[PrimeFaces Interview](https://www.javatpoint.com/primefaces-interview-questions)[RichFaces Interview](https://www.javatpoint.com/richfaces-interview-questions)[GWT Interview](https://www.javatpoint.com/gwt-interview-questions)[JDB Interview](https://www.javatpoint.com/jdb-interview-questions)[JOGL Interview](https://www.javatpoint.com/jogl-interview-questions)[jBPM Interview](https://www.javatpoint.com/jbpm-interview-questions)[Jenkins Interview](https://www.javatpoint.com/jenkins-interview-questions)[Java Constructor Interview Questions](https://www.javatpoint.com/java-constructor-interview-questions)[Java Technical Architect Interview Questions](https://www.javatpoint.com/java-technical-architect-interview-questions)

**Database Interview**

[DBMS Interview](https://www.javatpoint.com/dbms-interview-questions)[SQL Interview](https://www.javatpoint.com/sql-interview-questions)[PL/SQL Interview](https://www.javatpoint.com/pl-sql-interview-questions)[Oracle Interview](https://www.javatpoint.com/oracle-interview-questions)[MySql Interview](https://www.javatpoint.com/mysql-interview-questions)[SQL Server Interview](https://www.javatpoint.com/sql-server-interview-questions)[MongoDB Interview](https://www.javatpoint.com/mongodb-interview-questions)[Cassandra Interview](https://www.javatpoint.com/cassandra-interview-questions-and-answers)[DB2 Interview](https://www.javatpoint.com/db2-interview-questions-and-answers)[Access Interview](https://www.javatpoint.com/microsoft-access-interview-questions-and-answers)[SQLite Interview](https://www.javatpoint.com/sqlite-interview-questions)[CouchDB Interview](https://www.javatpoint.com/couchdb-interview-questions)[Neo4j Interview](https://www.javatpoint.com/neo4j-interview-questions-and-answers)[MariaDB Interview](https://www.javatpoint.com/mariadb-interview-questions)[PouchDB Interview](https://www.javatpoint.com/pouchdb-interview-questions)[Redis Interview](https://www.javatpoint.com/redis-interview-questions-and-answers)

[ADS BY ADRECOVER](https://adrecover.com/?utm_campaign=ByAdRecover&utm_source=www.javatpoint.com&utm_medium=ByAdRecover-160)

[ADVERTISEMENT BY ADRECOVER](https://adrecover.com/?utm_campaign=ByAdRecover&utm_source=www.javatpoint.com&utm_medium=ByAdRecover-728)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 300 Java interview questions | Set 2  [**1**](https://www.javatpoint.com/corejava-interview-questions) [**2**](https://www.javatpoint.com/corejava-interview-questions-2) [**3**](https://www.javatpoint.com/java-multithreading-interview-questions) [**4**](https://www.javatpoint.com/java-collections-interview-questions) [**5**](https://www.javatpoint.com/jdbc-interview-questions)  Core Java - OOPs: Polymorphism Interview Questions  101) What is the difference between compile-time polymorphism and runtime polymorphism?  There are the following differences between compile-time polymorphism and runtime polymorphism.   |  |  |  | | --- | --- | --- | | **SN** | **compile-time polymorphism** | **Runtime polymorphism** | | 1 | In compile-time polymorphism, call to a method is resolved at compile-time. | In runtime polymorphism, call to an overridden method is resolved at runtime. | | 2 | It is also known as static binding, early binding, or overloading. | It is also known as dynamic binding, late binding, overriding, or dynamic method dispatch. | | 3 | Overloading is a way to achieve compile-time polymorphism in which, we can define multiple methods or constructors with different signatures. | Overriding is a way to achieve runtime polymorphism in which, we can redefine some particular method or variable in the derived class. By using overriding, we can give some specific implementation to the base class properties in the derived class. | | 4 | It provides fast execution because the type of an object is determined at compile-time. | It provides slower execution as compare to compile-time because the type of an object is determined at run-time. | | 5 | Compile-time polymorphism provides less flexibility because all the things are resolved at compile-time. | Run-time polymorphism provides more flexibility because all the things are resolved at runtime. |   102) What is Runtime Polymorphism?  Runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time. In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.   1. **class** Bike{ 2. **void** run(){System.out.println("running");} 3. } 4. **class** Splendor **extends** Bike{ 5. **void** run(){System.out.println("running safely with 60km");} 6. **public** **static** **void** main(String args[]){ 7. Bike b = **new** Splendor();//upcasting 8. b.run(); 9. } 10. }   [**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=Splender)  Output:  running safely with 60km.  In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.  [More details.](https://www.javatpoint.com/runtime-polymorphism-in-java)  103) Can you achieve Runtime Polymorphism by data members?  No, because method overriding is used to achieve runtime polymorphism and data members cannot be overridden. We can override the member functions but not the data members. Consider the example given below.   1. **class** Bike{ 2. **int** speedlimit=90; 3. } 4. **class** Honda3 **extends** Bike{ 5. **int** speedlimit=150; 6. **public** **static** **void** main(String args[]){ 7. Bike obj=**new** Honda3(); 8. System.out.println(obj.speedlimit);//90 9. }   [**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=Honda3)  Output:  90  [More details.](https://www.javatpoint.com/runtime-polymorphism-in-java)  104) What is the difference between static binding and dynamic binding?  In case of the static binding, the type of the object is determined at compile-time whereas, in the dynamic binding, the type of the object is determined at runtime.  **Static Binding**   1. **class** Dog{ 2. **private** **void** eat(){System.out.println("dog is eating...");} 4. **public** **static** **void** main(String args[]){ 5. Dog d1=**new** Dog(); 6. d1.eat(); 7. } 8. }   **Dynamic Binding**   1. **class** Animal{ 2. **void** eat(){System.out.println("animal is eating...");} 3. } 5. **class** Dog **extends** Animal{ 6. **void** eat(){System.out.println("dog is eating...");} 8. **public** **static** **void** main(String args[]){ 9. Animal a=**new** Dog(); 10. a.eat(); 11. } 12. }   [More details.](https://www.javatpoint.com/static-binding-and-dynamic-binding)  105) What is the output of the following Java program?   1. **class** BaseTest 2. { 3. **void** print() 4. { 5. System.out.println("BaseTest:print() called"); 6. } 7. } 8. **public** **class** Test **extends** BaseTest 9. { 10. **void** print() 11. { 12. System.out.println("Test:print() called"); 13. } 14. **public** **static** **void** main (String args[]) 15. { 16. BaseTest b = **new** Test(); 17. b.print(); 18. } 19. }   **Output**  Test:print() called  **Explanation**  It is an example of Dynamic method dispatch. The type of reference variable b is determined at runtime. At compile-time, it is checked whether that method is present in the Base class. In this case, it is overridden in the child class, therefore, at runtime the derived class method is called.  106) What is Java instanceOf operator?  The instanceof in Java is also known as type comparison operator because it compares the instance with type. It returns either true or false. If we apply the instanceof operator with any variable that has a null value, it returns false. Consider the following example.   1. **class** Simple1{ 2. **public** **static** **void** main(String args[]){ 3. Simple1 s=**new** Simple1(); 4. System.out.println(s **instanceof** Simple1);//true 5. } 6. }   [**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=Simple1)  **Output**  true  An object of subclass type is also a type of parent class. For example, if Dog extends Animal then object of Dog can be referred by either Dog or Animal class.  Core Java - OOPs Concepts: Abstraction Interview Questions  107) What is the abstraction?  Abstraction is a process of hiding the implementation details and showing only functionality to the user. It displays just the essential things to the user and hides the internal information, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery. Abstraction enables you to focus on what the object does instead of how it does it. Abstraction lets you focus on what the object does instead of how it does it.  In Java, there are two ways to achieve the abstraction.   * Abstract Class * Interface   [More details.](https://www.javatpoint.com/abstract-class-in-java)  108) What is the difference between abstraction and encapsulation?  Abstraction hides the implementation details whereas encapsulation wraps code and data into a single unit.  [More details.](https://www.javatpoint.com/abstract-class-in-java)  109) What is the abstract class?  A class that is declared as abstract is known as an abstract class. It needs to be extended and its method implemented. It cannot be instantiated. It can have abstract methods, non-abstract methods, constructors, and static methods. It can also have the final methods which will force the subclass not to change the body of the method. Consider the following example.   1. **abstract** **class** Bike{ 2. **abstract** **void** run(); 3. } 4. **class** Honda4 **extends** Bike{ 5. **void** run(){System.out.println("running safely");} 6. **public** **static** **void** main(String args[]){ 7. Bike obj = **new** Honda4(); 8. obj.run(); 9. } 10. }   [**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=Honda4)  **Output**  running safely  [More details.](https://www.javatpoint.com/abstract-class-in-java)  110) Can there be an abstract method without an abstract class?  No, if there is an abstract method in a class, that class must be abstract.  111) Is the following program written correctly? If yes then what will be the output of the program?   1. **abstract** **class** Calculate 2. { 3. **abstract** **int** multiply(**int** a, **int** b); 4. } 6. **public** **class** Main 7. { 8. **public** **static** **void** main(String[] args) 9. { 10. **int** result = **new** Calculate() 11. { 12. @Override 13. **int** multiply(**int** a, **int** b) 14. { 15. **return** a\*b; 16. } 17. }.multiply(12,32); 18. System.out.println("result = "+result); 19. } 20. }   Yes, the program is written correctly. The Main class provides the definition of abstract method multiply declared in abstract class Calculation. The output of the program will be:  **Output**  384  112) Can you use abstract and final both with a method?  No, because we need to override the abstract method to provide its implementation, whereas we can't override the final method.  113) Is it possible to instantiate the abstract class?  No, the abstract class can never be instantiated even if it contains a constructor and all of its methods are implemented.  114) What is the interface?  The interface is a blueprint for a class that has static constants and abstract methods. It can be used to achieve full abstraction and multiple inheritance. It is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java. In other words, you can say that interfaces can have abstract methods and variables. Java Interface also represents the IS-A relationship. It cannot be instantiated just like the abstract class. However, we need to implement it to define its methods. Since Java 8, we can have the default, static, and private methods in an interface.  [More details.](https://www.javatpoint.com/interface-in-java)  115) Can you declare an interface method static?  No, because methods of an interface are abstract by default, and we can not use static and abstract together.  116) Can the Interface be final?  No, because an interface needs to be implemented by the other class and if it is final, it can't be implemented by any class.  117) What is a marker interface?  A Marker interface can be defined as the interface which has no data member and member functions. For example, Serializable, Cloneable are marker interfaces. The marker interface can be declared as follows.   1. **public** **interface** Serializable{ 2. }   118) What are the differences between abstract class and interface?   |  |  | | --- | --- | | **Abstract class** | **Interface** | | An abstract class can have a method body (non-abstract methods). | The interface has only abstract methods. | | An abstract class can have instance variables. | An interface cannot have instance variables. | | An abstract class can have the constructor. | The interface cannot have the constructor. | | An abstract class can have static methods. | The interface cannot have static methods. | | You can extend one abstract class. | You can implement multiple interfaces. | | The abstract class **can provide the implementation of the interface**. | The Interface **can't provide the implementation of the abstract class**. | | The **abstract keyword** is used to declare an abstract class. | The **interface keyword** is used to declare an interface. | | An **abstract class** can extend another Java class and implement multiple Java interfaces. | An **interface** can extend another Java interface only. | | An **abstract class** can be extended using keyword **extends** | An **interface class** can be implemented using keyword **implements** | | A Java**abstract class** can have class members like private, protected, etc. | Members of a Java interface are public by default. | | **Example:** public abstract class Shape{ public abstract void draw(); } | **Example:** public interface Drawable{ void draw(); } |   119) Can we define private and protected modifiers for the members in interfaces?  No, they are implicitly public.  120) When can an object reference be cast to an interface reference?  An object reference can be cast to an interface reference when the object implements the referenced interface.  121) How to make a read-only class in Java?  A class can be made read-only by making all of the fields private. The read-only class will have only getter methods which return the private property of the class to the main method. We cannot modify this property because there is no setter method available in the class. Consider the following example.   1. //A Java class which has only getter methods. 2. **public** **class** Student{ 3. //private data member 4. **private** String college="AKG"; 5. //getter method for college 6. **public** String getCollege(){ 7. **return** college; 8. } 9. }   122) How to make a write-only class in Java?  A class can be made write-only by making all of the fields private. The write-only class will have only setter methods which set the value passed from the main method to the private fields. We cannot read the properties of the class because there is no getter method in this class. Consider the following example.   1. //A Java class which has only setter methods. 2. **public** **class** Student{ 3. //private data member 4. **private** String college; 5. //getter method for college 6. **public** **void** setCollege(String college){ 7. **this**.college=college; 8. } 9. }   123) What are the advantages of Encapsulation in Java?  There are the following advantages of Encapsulation in Java?   * By providing only the setter or getter method, you can make the class read-only or write-only. In other words, you can skip the getter or setter methods. * It provides you the control over the data. Suppose you want to set the value of id which should be greater than 100 only, you can write the logic inside the setter method. You can write the logic not to store the negative numbers in the setter methods. * It is a way to achieve data hiding in Java because other class will not be able to access the data through the private data members. * The encapsulate class is easy to test. So, it is better for unit testing. * The standard IDE's are providing the facility to generate the getters and setters. So, it is easy and fast to create an encapsulated class in Java.   Core Java - OOPs Concepts: Package Interview Questions  124) What is the package?  A package is a group of similar type of classes, interfaces, and sub-packages. It provides access protection and removes naming collision. The packages in Java can be categorized into two forms, inbuilt package, and user-defined package. There are many built-in packages such as Java, lang, awt, javax, swing, net, io, util, sql, etc. Consider the following example to create a package in Java.   1. //save as Simple.java 2. **package** mypack; 3. **public** **class** Simple{ 4. **public** **static** **void** main(String args[]){ 5. System.out.println("Welcome to package"); 6. } 7. }   package in java [More details.](https://www.javatpoint.com/package)  125) What are the advantages of defining packages in Java?  By defining packages, we can avoid the name conflicts between the same class names defined in different packages. Packages also enable the developer to organize the similar classes more effectively. For example, one can clearly understand that the classes present in java.io package are used to perform io related operations.  126) How to create packages in Java?  If you are using the programming IDEs like Eclipse, NetBeans, MyEclipse, etc. click on **file->new->project** and eclipse will ask you to enter the name of the package. It will create the project package containing various directories such as src, etc. If you are using an editor like notepad for java programming, use the following steps to create the package.   * Define a package **package\_name**. Create the class with the name **class\_name** and save this file with **your\_class\_name.java**. * Now compile the file by running the following command on the terminal.   1. javac -d . your\_class\_name.java   The above command creates the package with the name **package\_name** in the present working directory.   * Now, run the class file by using the absolute class file name, like following.   1. java package\_name.class\_name   127) How can we access some class in another class in Java?  There are two ways to access a class in another class.   * **By using the fully qualified name:** To access a class in a different package, either we must use the fully qualified name of that class, or we must import the package containing that class. * **By using the relative path**, We can use the path of the class that is related to the package that contains our class. It can be the same or subpackage.   128) Do I need to import java.lang package any time? Why?  No. It is by default loaded internally by the JVM.  129) Can I import same package/class twice? Will the JVM load the package twice at runtime?  One can import the same package or the same class multiple times. Neither compiler nor JVM complains about it. However, the JVM will internally load the class only once no matter how many times you import the same class.  130) What is the static import?  By static import, we can access the static members of a class directly, and there is no to qualify it with the class name.  [More details.](https://www.javatpoint.com/static-import-in-java)  Java: Exception Handling Interview Questions  There is given a list of exception handling interview questions with answers. If you know any exception handling interview question, kindly post it in the comment section.  131) How many types of exception can occur in a Java program?  There are mainly two types of exceptions: checked and unchecked. Here, an error is considered as the unchecked exception. According to Oracle, there are three types of exceptions:   * **Checked Exception:** Checked exceptions are the one which are checked at compile-time. For example, SQLException, ClassNotFoundException, etc. * **Unchecked Exception:** Unchecked exceptions are the one which are handled at runtime because they can not be checked at compile-time. For example, ArithmaticException, NullPointerException, ArrayIndexOutOfBoundsException, etc. * **Error:** Error cause the program to exit since they are not recoverable. For Example, OutOfMemoryError, AssertionError, etc.   132) What is Exception Handling?  Exception Handling is a mechanism that is used to handle runtime errors. It is used primarily to handle checked exceptions. Exception handling maintains the normal flow of the program. There are mainly two types of exceptions: checked and unchecked. Here, the error is considered as the unchecked exception.  [More details.](https://www.javatpoint.com/exception-handling-and-checked-and-unchecked-exception)  133) Explain the hierarchy of Java Exception classes?  The java.lang.Throwable class is the root class of Java Exception hierarchy which is inherited by two subclasses: Exception and Error. A hierarchy of Java Exception classes are given below:  hierarchy of exception handling  134) What is the difference between Checked Exception and Unchecked Exception?  1) Checked Exception  The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions, e.g., IOException, SQLException, etc. Checked exceptions are checked at compile-time.  2) Unchecked Exception  The classes that extend RuntimeException are known as unchecked exceptions, e.g., ArithmeticException, NullPointerException, etc. Unchecked exceptions are not checked at compile-time.  [More details.](https://www.javatpoint.com/exception-handling-and-checked-and-unchecked-exception)  135) What is the base class for Error and Exception?  The Throwable class is the base class for Error and Exception.  136) Is it necessary that each try block must be followed by a catch block?  It is not necessary that each try block must be followed by a catch block. It should be followed by either a catch block OR a finally block. So whatever exceptions are likely to be thrown should be declared in the throws clause of the method. Consider the following example.   1. **public** **class** Main{ 2. **public** **static** **void** main(String []args){ 3. **try**{ 4. **int** a = 1; 5. System.out.println(a/0); 6. } 7. **finally** 8. { 9. System.out.println("rest of the code..."); 10. } 11. } 12. }   **Output:**  Exception in thread main java.lang.ArithmeticException:/ by zero  rest of the code...  137) What is the output of the following Java program?   1. **public** **class** ExceptionHandlingExample { 2. **public** **static** **void** main(String args[]) 3. { 4. **try** 5. { 6. **int** a = 1/0; 7. System.out.println("a = "+a); 8. } 9. **catch**(Exception e){System.out.println(e);} 10. **catch**(ArithmeticException ex){System.out.println(ex);} 11. } 12. }   **Output**  ExceptionHandlingExample.java:10: error: exception ArithmeticException has already been caught  catch(ArithmeticException ex){System.out.println(ex);}  ^  1 error  **Explanation**  ArithmaticException is the subclass of Exception. Therefore, it can not be used after Exception. Since Exception is the base class for all the exceptions, therefore, it must be used at last to handle the exception. No class can be used after this.  138) What is finally block?  The "finally" block is used to execute the important code of the program. It is executed whether an exception is handled or not. In other words, we can say that finally block is the block which is always executed. Finally block follows try or catch block. If you don't handle the exception, before terminating the program, JVM runs finally block, (if any). The finally block is mainly used to place the cleanup code such as closing a file or closing a connection. Here, we must know that for each try block there can be zero or more catch blocks, but only one finally block. The finally block will not be executed if program exits(either by calling System.exit() or by causing a fatal error that causes the process to abort).  java finally [More details.](https://www.javatpoint.com/finally-block-in-exception-handling)  139) Can finally block be used without a catch?  Yes, According to the definition of finally block, it must be followed by a try or catch block, therefore, we can use try block instead of catch.[More details.](https://www.javatpoint.com/finally-block-in-exception-handling)  140) Is there any case when finally will not be executed?  Finally block will not be executed if program exits(either by calling System.exit() or by causing a fatal error that causes the process to abort).[More details.](https://www.javatpoint.com/finally-block-in-exception-handling)  141) What is the difference between throw and throws?   |  |  | | --- | --- | | **throw keyword** | **throws keyword** | | 1) The **throw** keyword is used to throw an exception explicitly. | The **throws** keyword is used to declare an exception. | | 2) The checked exceptions cannot be propagated with throw only. | The checked exception can be propagated with throws | | 3) The **throw** keyword is followed by an instance. | The **throws** keyword is followed by class. | | 4) The **throw** keyword is used within the method. | The **throws** keyword is used with the method signature. | | 5) You cannot throw multiple exceptions. | You can declare multiple exceptions, e.g., public void method()throws IOException, SQLException. |   [More details.](https://www.javatpoint.com/throws-keyword-and-difference-between-throw-and-throws)  142) What is the output of the following Java program?   1. **public** **class** Main{ 2. **public** **static** **void** main(String []args){ 3. **try** 4. { 5. **throw** 90; 6. } 7. **catch**(**int** e){ 8. System.out.println("Caught the exception "+e); 9. } 11. } 12. }   **Output**  Main.java:6: error: incompatible types: int cannot be converted to Throwable  throw 90;  ^  Main.java:8: error: unexpected type  catch(int e){  ^  required: class  found: int  2 errors  **Explanation**  In Java, the throwable objects can only be thrown. If we try to throw an integer object, The compiler will show an error since we can not throw basic data type from a block of code.  143) What is the output of the following Java program?   1. **class** Calculation **extends** Exception 2. { 3. **public** Calculation() 4. { 5. System.out.println("Calculation class is instantiated"); 6. } 7. **public** **void** add(**int** a, **int** b) 8. { 9. System.out.println("The sum is "+(a+b)); 10. } 11. } 12. **public** **class** Main{ 13. **public** **static** **void** main(String []args){ 14. **try** 15. { 16. **throw** **new** Calculation(); 17. } 18. **catch**(Calculation c){ 19. c.add(10,20); 20. } 21. } 22. }   **Output**  Calculation class is instantiated  The sum is 30  **Explanation**  The object of Calculation is thrown from the try block which is caught in the catch block. The add() of Calculation class is called with the integer values 10 and 20 by using the object of this class. Therefore there sum 30 is printed. The object of the Main class can only be thrown in the case when the type of the object is throwable. To do so, we need to extend the throwable class.  144) Can an exception be rethrown?  Yes.  145) Can subclass overriding method declare an exception if parent class method doesn't throw an exception?  Yes but only unchecked exception not checked.  [More details.](https://www.javatpoint.com/exception-handling-with-method-overriding)  146) What is exception propagation?  An exception is first thrown from the top of the stack and if it is not caught, it drops down the call stack to the previous method, If not caught there, the exception again drops down to the previous method, and so on until they are caught or until they reach the very bottom of the call stack. This procedure is called exception propagation. By default, checked exceptions are not propagated.   1. **class** TestExceptionPropagation1{ 2. **void** m(){ 3. **int** data=50/0; 4. } 5. **void** n(){ 6. m(); 7. } 8. **void** p(){ 9. **try**{ 10. n(); 11. }**catch**(Exception e){System.out.println("exception handled");} 12. } 13. **public** **static** **void** main(String args[]){ 14. TestExceptionPropagation1 obj=**new** TestExceptionPropagation1(); 15. obj.p(); 16. System.out.println("normal flow..."); 17. } 18. }   [**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=TestExceptionPropagation1)  **Output:**  exception handled  normal flow...  exception propagation [More details.](https://www.javatpoint.com/exception-propagation)    147) What is the output of the following Java program?   1. **public** **class** Main 2. { 3. **void** a() 4. { 5. **try**{ 6. System.out.println("a(): Main called"); 7. b(); 8. }**catch**(Exception e) 9. { 10. System.out.println("Exception is caught"); 11. } 12. } 13. **void** b() **throws** Exception 14. { 15. **try**{ 16. System.out.println("b(): Main called"); 17. c(); 18. }**catch**(Exception e){ 19. **throw** **new** Exception(); 20. } 21. **finally** 22. { 23. System.out.println("finally block is called"); 24. } 25. } 26. **void** c() **throws** Exception 27. { 28. **throw** **new** Exception(); 29. } 31. **public** **static** **void** main (String args[]) 32. { 33. Main m = **new** Main(); 34. m.a(); 35. } 36. }   **Output**  a(): Main called  b(): Main called  finally block is called  Exception is caught  **Explanation**  In the main method, a() of Main is called which prints a message and call b(). The method b() prints some message and then call c(). The method c() throws an exception which is handled by the catch block of method b. However, It propagates this exception by using **throw Exception()** to be handled by the method a(). As we know, finally block is always executed therefore the finally block in the method b() is executed first and prints a message. At last, the exception is handled by the catch block of the method a().  148) What is the output of the following Java program?   1. **public** **class** Calculation 2. { 3. **int** a; 4. **public** Calculation(**int** a) 5. { 6. **this**.a = a; 7. } 8. **public** **int** add() 9. { 10. a = a+10; 11. **try** 12. { 13. a = a+10; 14. **try** 15. { 16. a = a\*10; 17. **throw** **new** Exception(); 18. }**catch**(Exception e){ 19. a = a - 10; 20. } 21. }**catch**(Exception e) 22. { 23. a = a - 10; 24. } 25. **return** a; 26. } 28. **public** **static** **void** main (String args[]) 29. { 30. Calculation c = **new** Calculation(10); 31. **int** result = c.add(); 32. System.out.println("result = "+result); 33. } 34. }   **Output**  result = 290  **Explanation**  The instance variable a of class Calculation is initialized to 10 using the class constructor which is called while instantiating the class. The add method is called which returns an integer value result. In add() method, a is incremented by 10 to be 20. Then, in the first try block, 10 is again incremented by 10 to be 30. In the second try block, a is multiplied by 10 to be 300. The second try block throws the exception which is caught by the catch block associated with this try block. The catch block again alters the value of a by decrementing it by 10 to make it 290. Thus the add() method returns 290 which is assigned to result. However, the catch block associated with the outermost try block will never be executed since there is no exception which can be handled by this catch block.  Java: String Handling Interview Questions  There is given a list of string handling interview questions with short and pointed answers. If you know any string handling interview question, kindly post it in the comment section.  149) What is String Pool?  String pool is the space reserved in the heap memory that can be used to store the strings. The main advantage of using the String pool is whenever we create a string literal; the JVM checks the "string constant pool" first. If the string already exists in the pool, a reference to the pooled instance is returned. If the string doesn't exist in the pool, a new string instance is created and placed in the pool. Therefore, it saves the memory by avoiding the duplicacy.  Java string literal  150) What is the meaning of immutable regarding String?  The simple meaning of immutable is unmodifiable or unchangeable. In Java, String is immutable, i.e., once string object has been created, its value can't be changed. Consider the following example for better understanding.   1. **class** Testimmutablestring{ 2. **public** **static** **void** main(String args[]){ 3. String s="Sachin"; 4. s.concat(" Tendulkar");//concat() method appends the string at the end 5. System.out.println(s);//will print Sachin because strings are immutable objects 6. } 7. }   [**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=Testimmutablestring)  **Output:**  Sachin  [More details.](https://www.javatpoint.com/immutable-string)  151) Why are the objects immutable in java?  Because Java uses the concept of the string literal. Suppose there are five reference variables, all refer to one object "sachin". If one reference variable changes the value of the object, it will be affected by all the reference variables. That is why string objects are immutable in java.  Heap diagram [More details.](https://www.javatpoint.com/immutable-string)  152) How many ways can we create the string object?  1) String Literal  Java String literal is created by using double quotes. For Example:   1. String s="welcome";   Each time you create a string literal, the JVM checks the "string constant pool" first. If the string already exists in the pool, a reference to the pooled instance is returned. If the string doesn't exist in the pool, a new string instance is created and placed in the pool. String objects are stored in a special memory area known as the **string constant pool** For example:   1. String s1="Welcome"; 2. String s2="Welcome";//It doesn't create a new instance   2) By new keyword   1. String s=**new** String("Welcome");//creates two objects and one reference variable   In such case, JVM will create a new string object in normal (non-pool) heap memory, and the literal "Welcome" will be placed in the constant string pool. The variable s will refer to the object in a heap (non-pool).  153) How many objects will be created in the following code?   1. String s1="Welcome"; 2. String s2="Welcome"; 3. String s3="Welcome";   Only one object will be created using the above code because strings in Java are immutable.  [More details.](https://www.javatpoint.com/string-handling-in-java)  154) Why java uses the concept of the string literal?  To make Java more memory efficient (because no new objects are created if it exists already in the string constant pool).  [More details.](https://www.javatpoint.com/string-handling-in-java)  155) How many objects will be created in the following code?   1. String s = **new** String("Welcome");   Two objects, one in string constant pool and other in non-pool(heap).  [More details.](https://www.javatpoint.com/string-handling-in-java)  156) What is the output of the following Java program?   1. **public** **class** Test 3. **public** **static** **void** main (String args[]) 4. { 5. String a = **new** String("Sharma is a good player"); 6. String b = "Sharma is a good player"; 7. **if**(a == b) 8. { 9. System.out.println("a == b"); 10. } 11. **if**(a.equals(b)) 12. { 13. System.out.println("a equals b"); 14. } 15. }   **Output**  a equals b  **Explanation**  The operator **==** also check whether the references of the two string objects are equal or not. Although both of the strings contain the same content, their references are not equal because both are created by different ways(Constructor and String literal) therefore, **a == b** is unequal. On the other hand, the equal() method always check for the content. Since their content is equal hence, **a equals b** is printed.  157) What is the output of the following Java program?   1. **public** **class** Test 2. { 3. **public** **static** **void** main (String args[]) 4. { 5. String s1 = "Sharma is a good player"; 6. String s2 = **new** String("Sharma is a good player"); 7. s2 = s2.intern(); 8. System.out.println(s1 ==s2); 9. } 10. }   **Output**  true  **Explanation**  The intern method returns the String object reference from the string pool. In this case, s1 is created by using string literal whereas, s2 is created by using the String pool. However, s2 is changed to the reference of s1, and the operator **==** returns true.  158) What are the differences between String and StringBuffer?  The differences between the String and StringBuffer is given in the table below.   |  |  |  | | --- | --- | --- | | **No.** | **String** | **StringBuffer** | | 1) | The String class is immutable. | The StringBuffer class is mutable. | | 2) | The String is slow and consumes more memory when you concat too many strings because every time it creates a new instance. | The StringBuffer is fast and consumes less memory when you cancat strings. | | 3) | The String class overrides the equals() method of Object class. So you can compare the contents of two strings by equals() method. | The StringBuffer class doesn't override the equals() method of Object class. |   159) What are the differences between StringBuffer and StringBuilder?  The differences between the StringBuffer and StringBuilder is given below.   |  |  |  | | --- | --- | --- | | **No.** | **StringBuffer** | **StringBuilder** | | 1) | StringBuffer is *synchronized*, i.e., thread safe. It means two threads can't call the methods of StringBuffer simultaneously. | StringBuilder is *non-synchronized*,i.e., not thread safe. It means two threads can call the methods of StringBuilder simultaneously. | | 2) | StringBuffer is *less efficient* than StringBuilder. | StringBuilder is *more efficient* than StringBuffer. |   160) How can we create an immutable class in Java?  We can create an immutable class by defining a final class having all of its members as final. Consider the following example.   1. **public** **final** **class** Employee{ 2. **final** String pancardNumber; 4. **public** Employee(String pancardNumber){ 5. **this**.pancardNumber=pancardNumber; 6. } 8. **public** String getPancardNumber(){ 9. **return** pancardNumber; 10. } 12. }   [More details.](https://www.javatpoint.com/how-to-create-immutable-class)  161) What is the purpose of toString() method in Java?  The toString() method returns the string representation of an object. If you print any object, java compiler internally invokes the toString() method on the object. So overriding the toString() method, returns the desired output, it can be the state of an object, etc. depending upon your implementation. By overriding the toString() method of the Object class, we can return the values of the object, so we don't need to write much code. Consider the following example.   1. **class** Student{ 2. **int** rollno; 3. String name; 4. String city; 6. Student(**int** rollno, String name, String city){ 7. **this**.rollno=rollno; 8. **this**.name=name; 9. **this**.city=city; 10. } 12. **public** String toString(){//overriding the toString() method 13. **return** rollno+" "+name+" "+city; 14. } 15. **public** **static** **void** main(String args[]){ 16. Student s1=**new** Student(101,"Raj","lucknow"); 17. Student s2=**new** Student(102,"Vijay","ghaziabad"); 19. System.out.println(s1);//compiler writes here s1.toString() 20. System.out.println(s2);//compiler writes here s2.toString() 21. } 22. }   **Output:**  101 Raj lucknow  102 Vijay ghaziabad  [More details.](https://www.javatpoint.com/understanding-toString()-method)  162) Why CharArray() is preferred over String to store the password?  String stays in the string pool until the garbage is collected. If we store the password into a string, it stays in the memory for a longer period, and anyone having the memory-dump can extract the password as clear text. On the other hand, Using CharArray allows us to set it to blank whenever we are done with the password. It avoids the security threat with the string by enabling us to control the memory.  163) Write a Java program to count the number of words present in a string?  **Program:**   1. **public** **class** Test 2. { 3. **public** **static** **void** main (String args[]) 4. { 5. String s = "Sharma is a good player and he is so punctual"; 6. String words[] = s.split(" "); 7. System.out.println("The Number of words present in the string are : "+words.length); 8. } 9. }   **Output**  The Number of words present in the string are : 10  164) Name some classes present in **java.util.regex** package.  There are the following classes and interfaces present in java.util.regex package.   * MatchResult Interface * Matcher class * Pattern class * PatternSyntaxException class   Java Regex API  165) How the metacharacters are different from the ordinary characters?  Metacharacters have the special meaning to the regular expression engine. The metacharacters are ^, $, ., \*, +, etc. The regular expression engine does not consider them as the regular characters. To enable the regular expression engine treating the metacharacters as ordinary characters, we need to escape the metacharacters with the backslash.  166) Write a regular expression to validate a password. A password must start with an alphabet and followed by alphanumeric characters; Its length must be in between 8 to 20.  The regular expression for the above criteria will be: **^[a-zA-Z][a-zA-Z0-9]{8,19}** where ^ represents the start of the regex, [a-zA-Z] represents that the first character must be an alphabet, [a-zA-Z0-9] represents the alphanumeric character, {8,19} represents that the length of the password must be in between 8 and 20.  167) What is the output of the following Java program?   1. **import** java.util.regex.\*; 2. **class** RegexExample2{ 3. **public** **static** **void** main(String args[]){ 4. System.out.println(Pattern.matches(".s", "as")); //line 4 5. System.out.println(Pattern.matches(".s", "mk")); //line 5 6. System.out.println(Pattern.matches(".s", "mst")); //line 6 7. System.out.println(Pattern.matches(".s", "amms")); //line 7 8. System.out.println(Pattern.matches("..s", "mas")); //line 8 9. }}   **Output**  true  false  false  false  true  **Explanation**  line 4 prints true since the second character of string is s, line 5 prints false since the second character is not s, line 6 prints false since there are more than 3 characters in the string, line 7 prints false since there are more than 2 characters in the string, and it contains more than 2 characters as well, line 8 prints true since the third character of the string is s.  Core Java: Nested classes and Interfaces Interview Questions  168) What are the advantages of Java inner classes?  There are two types of advantages of Java inner classes.   * Nested classes represent a special type of relationship that is it can access all the members (data members and methods) of the outer class including private. * Nested classes are used to develop a more readable and maintainable code because it logically groups classes and interfaces in one place only. * **Code Optimization:** It requires less code to write.   169) What is a nested class?  The nested class can be defined as the class which is defined inside another class or interface. We use the nested class to logically group classes and interfaces in one place so that it can be more readable and maintainable. A nested class can access all the data members of the outer class including private data members and methods. The syntax of the nested class is defined below.   1. **class** Java\_Outer\_class{ 2. //code 3. **class** Java\_Nested\_class{ 4. //code 5. } 6. }   There are two types of nested classes, static nested class, and non-static nested class. The non-static nested class can also be called as inner-class  [More details.](https://www.javatpoint.com/difference-between-nested-classes-and-inner-classes)  170) What are the disadvantages of using inner classes?  There are the following main disadvantages of using inner classes.   * Inner classes increase the total number of classes used by the developer and therefore increases the workload of JVM since it has to perform some routine operations for those extra classes which result in slower performance. * IDEs provide less support to the inner classes as compare to the top level classes and therefore it annoys the developers while working with inner classes.   171) What are the types of inner classes (non-static nested class) used in Java?  There are mainly three types of inner classes used in Java.   |  |  | | --- | --- | | **Type** | **Description** | | [Member Inner Class](https://www.javatpoint.com/member-inner-class) | A class created within class and outside method. | | [Anonymous Inner Class](https://www.javatpoint.com/anonymous-inner-class) | A class created for implementing an interface or extending class. Its name is decided by the java compiler. | | [Local Inner Class](https://www.javatpoint.com/local-inner-class) | A class created within the method. |   172) Is there any difference between nested classes and inner classes?  Yes, inner classes are non-static nested classes. In other words, we can say that inner classes are the part of nested classes.  [More details.](https://www.javatpoint.com/difference-between-nested-classes-and-inner-classes)  173) Can we access the non-final local variable, inside the local inner class?  No, the local variable must be constant if you want to access it in the local inner class.  [More details.](https://www.javatpoint.com/local-inner-class)  174) How many class files are created on compiling the OuterClass in the following program?   1. **public** **class** Person { 2. String name, age, address; 3. **class** Employee{ 4. **float** salary=10000; 5. } 6. **class** BusinessMen{ 7. **final** String gstin="£4433drt3$"; 8. } 9. **public** **static** **void** main (String args[]) 10. { 11. Person p = **new** Person(); 12. } 13. }   3 class-files will be created named as Person.class, Person$BusinessMen.class, and Person$Employee.class.  175) What are anonymous inner classes?  Anonymous inner classes are the classes that are automatically declared and instantiated within an expression. We cannot apply different access modifiers to them. Anonymous class cannot be static, and cannot define any static fields, method, or class. In other words, we can say that it a class without the name and can have only one object that is created by its definition. Consider the following example.   1. **abstract** **class** Person{ 2. **abstract** **void** eat(); 3. } 4. **class** TestAnonymousInner{ 5. **public** **static** **void** main(String args[]){ 6. Person p=**new** Person(){ 7. **void** eat(){System.out.println("nice fruits");} 8. }; 9. p.eat(); 10. } 11. }   [**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=TestAnnonymousInner)  Output:  nice fruits  Consider the following example for the working of the anonymous class using interface.   1. **interface** Eatable{ 2. **void** eat(); 3. } 4. **class** TestAnnonymousInner1{ 5. **public** **static** **void** main(String args[]){ 6. Eatable e=**new** Eatable(){ 7. **public** **void** eat(){System.out.println("nice fruits");} 8. }; 9. e.eat(); 10. } 11. }   [**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=TestAnnonymousInner1)  Output:  nice fruits  176) What is the nested interface?  An Interface that is declared inside the interface or class is known as the nested interface. It is static by default. The nested interfaces are used to group related interfaces so that they can be easy to maintain. The external interface or class must refer to the nested interface. It can't be accessed directly. The nested interface must be public if it is declared inside the interface but it can have any access modifier if declared within the class. The syntax of the nested interface is given as follows.   1. **interface** interface\_name{ 2. ... 3. **interface** nested\_interface\_name{ 4. ... 5. } 6. }   [More details.](https://www.javatpoint.com/nested-interface)  177) Can a class have an interface?  Yes, an interface can be defined within the class. It is called a nested interface.  [More details.](https://www.javatpoint.com/nested-interface)  178) Can an Interface have a class?  Yes, they are static implicitly.  [More details.](https://www.javatpoint.com/nested-interface)  Garbage Collection Interview Questions  179) What is Garbage Collection?  Garbage collection is a process of reclaiming the unused runtime objects. It is performed for memory management. In other words, we can say that It is the process of removing unused objects from the memory to free up space and make this space available for Java Virtual Machine. Due to garbage collection java gives 0 as output to a variable whose value is not set, i.e., the variable has been defined but not initialized. For this purpose, we were using free() function in the C language and delete() in C++. In Java, it is performed automatically. So, java provides better memory management.  [More details.](https://www.javatpoint.com/Garbage-Collection)  180) What is gc()?  The gc() method is used to invoke the garbage collector for cleanup processing. This method is found in System and Runtime classes. This function explicitly makes the Java Virtual Machine free up the space occupied by the unused objects so that it can be utilized or reused. Consider the following example for the better understanding of how the gc() method invoke the garbage collector.   1. **public** **class** TestGarbage1{ 2. **public** **void** finalize(){System.out.println("object is garbage collected");} 3. **public** **static** **void** main(String args[]){ 4. TestGarbage1 s1=**new** TestGarbage1(); 5. TestGarbage1 s2=**new** TestGarbage1(); 6. s1=**null**; 7. s2=**null**; 8. System.gc(); 9. } 10. }   [**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=TestGarbage1)  object is garbage collected  object is garbage collected  181) How is garbage collection controlled?  Garbage collection is managed by JVM. It is performed when there is not enough space in the memory and memory is running low. We can externally call the System.gc() for the garbage collection. However, it depends upon the JVM whether to perform it or not.  182) How can an object be unreferenced?  There are many ways:   * By nulling the reference * By assigning a reference to another * By anonymous object etc.   Java Garbage Collection Scenario  1) By nulling a reference:   1. Employee e=**new** Employee(); 2. e=**null**;   2) By assigning a reference to another:   1. Employee e1=**new** Employee(); 2. Employee e2=**new** Employee(); 3. e1=e2;//now the first object referred by e1 is available for garbage collection   3) By anonymous object:   1. **new** Employee();   183) What is the purpose of the finalize() method?  The finalize() method is invoked just before the object is garbage collected. It is used to perform cleanup processing. The Garbage collector of JVM collects only those objects that are created by new keyword. So if you have created an object without new, you can use the finalize method to perform cleanup processing (destroying remaining objects). The cleanup processing is the process to free up all the resources, network which was previously used and no longer needed. It is essential to remember that it is not a reserved keyword, finalize method is present in the object class hence it is available in every class as object class is the superclass of every class in java. Here, we must note that neither finalization nor garbage collection is guaranteed. Consider the following example.   1. **public** **class** FinalizeTest { 2. **int** j=12; 3. **void** add() 4. { 5. j=j+12; 6. System.out.println("J="+j); 7. } 8. **public** **void** finalize() 9. { 10. System.out.println("Object is garbage collected"); 11. } 12. **public** **static** **void** main(String[] args) { 13. **new** FinalizeTest().add(); 14. System.gc(); 15. **new** FinalizeTest().add(); 16. } 17. }   184) Can an unreferenced object be referenced again?  Yes,  185) What kind of thread is the Garbage collector thread?  Daemon thread.  186) What is the difference between final, finally and finalize?   |  |  |  |  | | --- | --- | --- | --- | | **No.** | **final** | **finally** | **finalize** | | 1) | Final is used to apply restrictions on class, method, and variable. The final class can't be inherited, final method can't be overridden, and final variable value can't be changed. | Finally is used to place important code, it will be executed whether an exception is handled or not. | Finalize is used to perform clean up processing just before an object is garbage collected. | | 2) | Final is a keyword. | Finally is a block. | Finalize is a method. |   187) What is the purpose of the Runtime class?  Java Runtime class is used to interact with a java runtime environment. Java Runtime class provides methods to execute a process, invoke GC, get total and free memory, etc. There is only one instance of java.lang.Runtime class is available for one java application. The Runtime.getRuntime() method returns the singleton instance of Runtime class.  188) How will you invoke any external process in Java?  By Runtime.getRuntime().exec(?) method. Consider the following example.   1. **public** **class** Runtime1{ 2. **public** **static** **void** main(String args[])**throws** Exception{ 3. Runtime.getRuntime().exec("notepad");//will open a new notepad 4. } 5. }   I/O Interview Questions  189) Give the hierarchy of InputStream and OutputStream classes.  **OutputStream Hierarchy**  **InputStream Hierarchy**  190) What do you understand by an IO stream?  The stream is a sequence of data that flows from source to destination. It is composed of bytes. In Java, three streams are created for us automatically.   * System.out: standard output stream * System.in: standard input stream * System.err: standard error stream   191) What is the difference between the Reader/Writer class hierarchy and the InputStream/OutputStream class hierarchy?  The Reader/Writer class hierarchy is character-oriented, and the InputStream/OutputStream class hierarchy is byte-oriented. The ByteStream classes are used to perform input-output of 8-bit bytes whereas the CharacterStream classes are used to perform the input/output for the 16-bit Unicode system. There are many classes in the ByteStream class hierarchy, but the most frequently used classes are FileInputStream and FileOutputStream. The most frequently used classes CharacterStream class hierarchy is FileReader and FileWriter.  192) What are the super most classes for all the streams?  All the stream classes can be divided into two types of classes that are ByteStream classes and CharacterStream Classes. The ByteStream classes are further divided into InputStream classes and OutputStream classes. CharacterStream classes are also divided into Reader classes and Writer classes. The SuperMost classes for all the InputStream classes is java.io.InputStream and for all the output stream classes is java.io.OutPutStream. Similarly, for all the reader classes, the super-most class is java.io.Reader, and for all the writer classes, it is java.io.Writer.  193) What are the FileInputStream and FileOutputStream?  **Java FileOutputStream** is an output stream used for writing data to a file. If you have some primitive values to write into a file, use FileOutputStream class. You can write byte-oriented as well as character-oriented data through the FileOutputStream class. However, for character-oriented data, it is preferred to use FileWriter than FileOutputStream. Consider the following example of writing a byte into a file.   1. **import** java.io.FileOutputStream; 2. **public** **class** FileOutputStreamExample { 3. **public** **static** **void** main(String args[]){ 4. **try**{ 5. FileOutputStream fout=**new** FileOutputStream("D:\\testout.txt"); 6. fout.write(65); 7. fout.close(); 8. System.out.println("success..."); 9. }**catch**(Exception e){System.out.println(e);} 10. } 11. }   **Java FileInputStream class**obtains input bytes from a file. It is used for reading byte-oriented data (streams of raw bytes) such as image data, audio, video, etc. You can also read character-stream data. However, for reading streams of characters, it is recommended to use FileReader class. Consider the following example for reading bytes from a file.   1. **import** java.io.FileInputStream; 2. **public** **class** DataStreamExample { 3. **public** **static** **void** main(String args[]){ 4. **try**{ 5. FileInputStream fin=**new** FileInputStream("D:\\testout.txt"); 6. **int** i=fin.read(); 7. System.out.print((**char**)i); 9. fin.close(); 10. }**catch**(Exception e){System.out.println(e);} 11. } 12. }   194) What is the purpose of using BufferedInputStream and BufferedOutputStream classes?  Java BufferedOutputStream class is used for buffering an output stream. It internally uses a buffer to store data. It adds more efficiency than to write data directly into a stream. So, it makes the performance fast. Whereas, Java BufferedInputStream class is used to read information from the stream. It internally uses the buffer mechanism to make the performance fast.  195) How to set the Permissions to a file in Java?  In Java, FilePermission class is used to alter the permissions set on a file. Java FilePermission class contains the permission related to a directory or file. All the permissions are related to the path. The path can be of two types:   * D:\\IO\\-: It indicates that the permission is associated with all subdirectories and files recursively. * D:\\IO\\\*: It indicates that the permission is associated with all directory and files within this directory excluding subdirectories.   Let's see the simple example in which permission of a directory path is granted with read permission and a file of this directory is granted for write permission.   1. **package** com.javatpoint; 2. **import** java.io.\*; 3. **import** java.security.PermissionCollection; 4. **public** **class** FilePermissionExample{ 5. **public** **static** **void** main(String[] args) **throws** IOException { 6. String srg = "D:\\IO Package\\java.txt"; 7. FilePermission file1 = **new** FilePermission("D:\\IO Package\\-", "read"); 8. PermissionCollection permission = file1.newPermissionCollection(); 9. permission.add(file1); 10. FilePermission file2 = **new** FilePermission(srg, "write"); 11. permission.add(file2); 12. **if**(permission.implies(**new** FilePermission(srg, "read,write"))) { 13. System.out.println("Read, Write permission is granted for the path "+srg ); 14. }**else** { 15. System.out.println("No Read, Write permission is granted for the path "+srg);            } 16. } 17. }   Output  Read, Write permission is granted for the path D:\IO Package\java.txt  196) What are FilterStreams?  **FilterStream classes** are used to add additional functionalities to the other stream classes. FilterStream classes act like an interface which read the data from a stream, filters it, and pass the filtered data to the caller. The FilterStream classes provide extra functionalities like adding line numbers to the destination file, etc.  197) What is an I/O filter?  An I/O filter is an object that reads from one stream and writes to another, usually altering the data in some way as it is passed from one stream to another. Many Filter classes that allow a user to make a chain using multiple input streams. It generates a combined effect on several filters.  198) In Java, How many ways you can take input from the console?  In Java, there are three ways by using which, we can take input from the console.   * **Using BufferedReader class:** we can take input from the console by wrapping System.in into an InputStreamReader and passing it into the BufferedReader. It provides an efficient reading as the input gets buffered. Consider the following example.   1. **import** java.io.BufferedReader;   2. **import** java.io.IOException;   3. **import** java.io.InputStreamReader;   4. **public** **class** Person   5. {   6. **public** **static** **void** main(String[] args) **throws** IOException   7. {   8. System.out.println("Enter the name of the person");   9. BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(System.in));   10. String name = reader.readLine();   11. System.out.println(name);   12. }   13. } * **Using Scanner class:** The Java Scanner class breaks the input into tokens using a delimiter that is whitespace by default. It provides many methods to read and parse various primitive values. Java Scanner class is widely used to parse text for string and primitive types using a regular expression. Java Scanner class extends Object class and implements Iterator and Closeable interfaces. Consider the following example.   1. **import** java.util.\*;   2. **public** **class** ScannerClassExample2 {   3. **public** **static** **void** main(String args[]){   4. String str = "Hello/This is JavaTpoint/My name is Abhishek.";   5. //Create scanner with the specified String Object   6. Scanner scanner = **new** Scanner(str);   7. System.out.println("Boolean Result: "+scanner.hasNextBoolean());   8. //Change the delimiter of this scanner   9. scanner.useDelimiter("/");   10. //Printing the tokenized Strings   11. System.out.println("---Tokenizes String---");   12. **while**(scanner.hasNext()){   13. System.out.println(scanner.next());   14. }   15. //Display the new delimiter   16. System.out.println("Delimiter used: " +scanner.delimiter());   17. scanner.close();   18. }   19. } * **Using Console class:** The Java Console class is used to get input from the console. It provides methods to read texts and passwords. If you read the password using the Console class, it will not be displayed to the user. The java.io.Console class is attached to the system console internally. The Console class is introduced since 1.5. Consider the following example.   1. **import** java.io.Console;   2. **class** ReadStringTest{   3. **public** **static** **void** main(String args[]){   4. Console c=System.console();   5. System.out.println("Enter your name: ");   6. String n=c.readLine();   7. System.out.println("Welcome "+n);   8. }   9. }   Serialization Interview Questions  199) What is serialization?  Serialization in Java is a mechanism of writing the state of an object into a byte stream. It is used primarily in Hibernate, RMI, JPA, EJB and JMS technologies. It is mainly used to travel object's state on the network (which is known as marshaling). Serializable interface is used to perform serialization. It is helpful when you require to save the state of a program to storage such as the file. At a later point of time, the content of this file can be restored using deserialization. It is also required to implement RMI(Remote Method Invocation). With the help of RMI, it is possible to invoke the method of a Java object on one machine to another machine.  java serialization [More details.](https://www.javatpoint.com/serialization)  200) How can you make a class serializable in Java?  A class can become serializable by implementing the Serializable interface.  201) How can you avoid serialization in child class if the base class is implementing the Serializable interface?  It is very tricky to prevent serialization of child class if the base class is intended to implement the Serializable interface. However, we cannot do it directly, but the serialization can be avoided by implementing the writeObject() or readObject() methods in the subclass and throw NotSerializableException from these methods. Consider the following example.   1. **import** java.io.FileInputStream; 2. **import** java.io.FileOutputStream; 3. **import** java.io.IOException; 4. **import** java.io.NotSerializableException; 5. **import** java.io.ObjectInputStream; 6. **import** java.io.ObjectOutputStream; 7. **import** java.io.Serializable; 8. **class** Person **implements** Serializable 9. { 10. String name = " "; 11. **public** Person(String name) 12. { 13. **this**.name = name; 14. } 15. } 16. **class** Employee **extends** Person 17. { 18. **float** salary; 19. **public** Employee(String name, **float** salary) 20. { 21. **super**(name); 22. **this**.salary = salary; 23. } 24. **private** **void** writeObject(ObjectOutputStream out) **throws** IOException 25. { 26. **throw** **new** NotSerializableException(); 27. } 28. **private** **void** readObject(ObjectInputStream in) **throws** IOException 29. { 30. **throw** **new** NotSerializableException(); 31. } 33. } 34. **public** **class** Test 35. { 36. **public** **static** **void** main(String[] args) 37. **throws** Exception 38. { 39. Employee emp = **new** Employee("Sharma", 10000); 41. System.out.println("name = " + emp.name); 42. System.out.println("salary = " + emp.salary); 44. FileOutputStream fos = **new** FileOutputStream("abc.ser"); 45. ObjectOutputStream oos = **new** ObjectOutputStream(fos); 47. oos.writeObject(emp); 49. oos.close(); 50. fos.close(); 52. System.out.println("Object has been serialized"); 54. FileInputStream f = **new** FileInputStream("ab.txt"); 55. ObjectInputStream o = **new** ObjectInputStream(f); 57. Employee emp1 = (Employee)o.readObject(); 59. o.close(); 60. f.close(); 62. System.out.println("Object has been deserialized"); 64. System.out.println("name = " + emp1.name); 65. System.out.println("salary = " + emp1.salary); 66. } 67. }   202) Can a Serialized object be transferred via network?  Yes, we can transfer a serialized object via network because the serialized object is stored in the memory in the form of bytes and can be transmitted over the network. We can also write the serialized object to the disk or the database.  203) What is Deserialization?  Deserialization is the process of reconstructing the object from the serialized state. It is the reverse operation of serialization. An ObjectInputStream deserializes objects and primitive data written using an ObjectOutputStream.   1. **import** java.io.\*; 2. **class** Depersist{ 3. **public** **static** **void** main(String args[])**throws** Exception{ 5. ObjectInputStream in=**new** ObjectInputStream(**new** FileInputStream("f.txt")); 6. Student s=(Student)in.readObject(); 7. System.out.println(s.id+" "+s.name); 9. in.close(); 10. } 11. }   211 ravi  204) What is the transient keyword?  If you define any data member as transient, it will not be serialized. By determining transient keyword, the value of variable need not persist when it is restored. [More details.](https://www.javatpoint.com/serialization)  205) What is Externalizable?  The Externalizable interface is used to write the state of an object into a byte stream in a compressed format. It is not a marker interface.  206) What is the difference between Serializable and Externalizable interface?   |  |  |  | | --- | --- | --- | | **No.** | **Serializable** | **Externalizable** | | 1) | The Serializable interface does not have any method, i.e., it is a marker interface. | The Externalizable interface contains is not a marker interface, It contains two methods, i.e., writeExternal() and readExternal(). | | 2) | It is used to "mark" Java classes so that objects of these classes may get the certain capability. | The Externalizable interface provides control of the serialization logic to the programmer. | | 3) | It is easy to implement but has the higher performance cost. | It is used to perform the serialization and often result in better performance. | | 4) | No class constructor is called in serialization. | We must call a public default constructor while using this interface. |   .  Networking Interview Questions  207) Give a brief description of Java socket programming?  Java Socket programming is used for communication between the applications running on different JRE. Java Socket programming can be connection-oriented or connectionless. Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket, and DatagramPacket classes are used for connectionless socket programming. The client in socket programming must know two information:   * IP address of the server * port number   208) What is Socket?  A socket is simply an endpoint for communications between the machines. It provides the connection mechanism to connect the two computers using TCP. The Socket class can be used to create a socket.  209) What are the steps that are followed when two computers connect through TCP?  There are the following steps that are performed when two computers connect through TCP.   * The ServerSocket object is instantiated by the server which denotes the port number to which, the connection will be made. * After instantiating the ServerSocket object, the server invokes accept() method of ServerSocket class which makes server wait until the client attempts to connect to the server on the given port. * Meanwhile, the server is waiting, a socket is created by the client by instantiating Socket class. The socket class constructor accepts the server port number and server name. * The Socket class constructor attempts to connect with the server on the specified name. If the connection is established, the client will have a socket object that can communicate with the server. * The accept() method invoked by the server returns a reference to the new socket on the server that is connected with the server.   210) Write a program in Java to establish a connection between client and server?  Consider the following program where the connection between the client and server is established.  *File: MyServer.java*   1. **import** java.io.\*; 2. **import** java.net.\*; 3. **public** **class** MyServer { 4. **public** **static** **void** main(String[] args){ 5. **try**{ 6. ServerSocket ss=**new** ServerSocket(6666); 7. Socket s=ss.accept();//establishes connection 8. DataInputStream dis=**new** DataInputStream(s.getInputStream()); 9. String  str=(String)dis.readUTF(); 10. System.out.println("message= "+str); 11. ss.close(); 12. }**catch**(Exception e){System.out.println(e);} 13. } 14. }   *File: MyClient.java*   1. **import** java.io.\*; 2. **import** java.net.\*; 3. **public** **class** MyClient { 4. **public** **static** **void** main(String[] args) { 5. **try**{ 6. Socket s=**new** Socket("localhost",6666); 7. DataOutputStream dout=**new** DataOutputStream(s.getOutputStream()); 8. dout.writeUTF("Hello Server"); 9. dout.flush(); 10. dout.close(); 11. s.close(); 12. }**catch**(Exception e){System.out.println(e);} 13. } 14. }   211) How do I convert a numeric IP address like 192.18.97.39 into a hostname like java.sun.com?  By InetAddress.getByName("192.18.97.39").getHostName() where 192.18.97.39 is the IP address. Consider the following example.   1. **import** java.io.\*; 2. **import** java.net.\*; 3. **public** **class** InetDemo{ 4. **public** **static** **void** main(String[] args){ 5. **try**{ 6. InetAddress ip=InetAddress.getByName("195.201.10.8"); 8. System.out.println("Host Name: "+ip.getHostName()); 9. }**catch**(Exception e){System.out.println(e);} 10. } 11. }   Reflection Interview Questions  212) What is the reflection?  Reflection is the process of examining or modifying the runtime behavior of a class at runtime. The java.lang.Class class provides various methods that can be used to get metadata, examine and change the runtime behavior of a class. The java.lang and java.lang.reflect packages provide classes for java reflection. It is used in:   * IDE (Integrated Development Environment), e.g., Eclipse, MyEclipse, NetBeans. * Debugger * Test Tools, etc.   213) What is the purpose of using java.lang.Class class?  The java.lang.Class class performs mainly two tasks:   * Provides methods to get the metadata of a class at runtime. * Provides methods to examine and change the runtime behavior of a class.   214) What are the ways to instantiate the Class class?  There are three ways to instantiate the Class class.   * **forName() method of Class class:**The forName() method is used to load the class dynamically. It returns the instance of Class class. It should be used if you know the fully qualified name of the class. This cannot be used for primitive types. * **getClass() method of Object class:** It returns the instance of Class class. It should be used if you know the type. Moreover, it can be used with primitives. * **the .class syntax:** If a type is available, but there is no instance then it is possible to obtain a Class by appending ".class" to the name of the type. It can be used for primitive data type also.   215) What is the output of the following Java program?   1. **class** Simple{ 2. **public** Simple() 3. { 4. System.out.println("Constructor of Simple class is invoked"); 5. } 6. **void** message(){System.out.println("Hello Java");} 7. } 9. **class** Test1{ 10. **public** **static** **void** main(String args[]){ 11. **try**{ 12. Class c=Class.forName("Simple"); 13. Simple s=(Simple)c.newInstance(); 14. s.message(); 15. }**catch**(Exception e){System.out.println(e);} 16. } 17. }   **Output**  Constructor of Simple class is invoked  Hello Java  **Explanation**  The newInstance() method of the Class class is used to invoke the constructor at runtime. In this program, the instance of the Simple class is created.  216) What is the purpose of using javap?  The javap command disassembles a class file. The javap command displays information about the fields, constructors and methods present in a class file.  **Syntax**  javap fully\_class\_name  217) Can you access the private method from outside the class?  Yes, by changing the runtime behavior of a class if the class is not secured.  [More details.](https://www.javatpoint.com/reflection6)  Miscellaneous Interview Questions  218)What are wrapper classes?  Wrapper classes are classes that allow primitive types to be accessed as objects. In other words, we can say that wrapper classes are built-in java classes which allow the conversion of objects to primitives and primitives to objects. The process of converting primitives to objects is called autoboxing, and the process of converting objects to primitives is called unboxing. There are eight wrapper classes present in **java.lang** package is given below.   |  |  | | --- | --- | | **Primitive Type** | **Wrapper class** | | boolean | Boolean | | char | Character | | byte | Byte | | short | Short | | int | Integer | | long | Long | | float | Float | | double | Double |   219)What are autoboxing and unboxing? When does it occur?  The autoboxing is the process of converting primitive data type to the corresponding wrapper class object, eg., int to Integer. The unboxing is the process of converting wrapper class object to primitive data type. For eg., integer to int. Unboxing and autoboxing occur automatically in Java. However, we can externally convert one into another by using the methods like valueOf() or xxxValue().  It can occur whenever a wrapper class object is expected, and primitive data type is provided or vice versa.   * Adding primitive types into Collection like ArrayList in Java. * Creating an instance of parameterized classes ,e.g., ThreadLocal which expect Type. * Java automatically converts primitive to object whenever one is required and another is provided in the method calling. * When a primitive type is assigned to an object type.   220) What is the output of the below Java program?   1. **public** **class** Test1 2. { 3. **public** **static** **void** main(String[] args) { 4. Integer i = **new** Integer(201); 5. Integer j = **new** Integer(201); 6. **if**(i == j) 7. { 8. System.out.println("hello"); 9. } 10. **else** 11. { 12. System.out.println("bye"); 13. } 14. } 15. }   **Output**  bye  **Explanation**  The Integer class caches integer values from -127 to 127. Therefore, the Integer objects can only be created in the range -128 to 127. The operator **==** will not work for the value greater than 127; thus **bye** is printed.  221) What is object cloning?  The object cloning is a way to create an exact copy of an object. The clone() method of the Object class is used to clone an object. The java.lang.Cloneable interface must be implemented by the class whose object clone we want to create. If we don't implement Cloneable interface, clone() method generates CloneNotSupportedException. The clone() method is defined in the Object class. The syntax of the clone() method is as follows:  **protected Object clone() throws CloneNotSupportedException**  222) What are the advantages and disadvantages of object cloning?  **Advantage of Object Cloning**   * You don't need to write lengthy and repetitive codes. Just use an abstract class with a 4- or 5-line long clone() method. * It is the easiest and most efficient way of copying objects, especially if we are applying it to an already developed or an old project. Just define a parent class, implement Cloneable in it, provide the definition of the clone() method and the task will be done. * Clone() is the fastest way to copy the array.   **Disadvantage of Object Cloning**   * To use the Object.clone() method, we have to change many syntaxes to our code, like implementing a Cloneable interface, defining the clone() method and handling CloneNotSupportedException, and finally, calling Object.clone(), etc. * We have to implement the Cloneable interface while it does not have any methods in it. We have to use it to tell the JVM that we can perform a clone() on our object. * Object.clone() is protected, so we have to provide our own clone() and indirectly call Object.clone() from it. * Object.clone() does not invoke any constructor, so we do not have any control over object construction. * If you want to write a clone method in a child class, then all of its superclasses should define the clone() method in them or inherit it from another parent class. Otherwise, the super.clone() chain will fail. * Object.clone() supports only shallow copying, but we will need to override it if we need deep cloning.   223) What is a native method?  A native method is a method that is implemented in a language other than Java. Natives methods are sometimes also referred to as foreign methods.  224) What is the purpose of the strictfp keyword?  Java strictfp keyword ensures that you will get the same result on every platform if you perform operations in the floating-point variable. The precision may differ from platform to platform that is why java programming language has provided the strictfp keyword so that you get the same result on every platform. So, now you have better control over the floating-point arithmetic.  225) What is the purpose of the System class?  The purpose of the System class is to provide access to system resources such as standard input and output. It cannot be instantiated. Facilities provided by System class are given below.   * Standard input * Error output streams * Standard output * utility method to copy the portion of an array * utilities to load files and libraries   There are the three fields of Java System class, i.e., static printstream err, static inputstream in, and standard output stream.  226) What comes to mind when someone mentions a shallow copy in Java?  Object cloning.  227) What is a singleton class?  Singleton class is the class which can not be instantiated more than once. To make a class singleton, we either make its constructor private or use the static getInstance method. Consider the following example.   1. **class** Singleton{ 2. **private** **static** Singleton single\_instance = **null**; 3. **int** i; 4. **private** Singleton () 5. { 6. i=90; 7. } 8. **public** **static** Singleton getInstance() 9. { 10. **if**(single\_instance == **null**) 11. { 12. single\_instance = **new** Singleton(); 13. } 14. **return** single\_instance; 15. } 16. } 17. **public** **class** Main 18. { 19. **public** **static** **void** main (String args[]) 20. { 21. Singleton first = Singleton.getInstance(); 22. System.out.println("First instance integer value:"+first.i); 23. first.i=first.i+90; 24. Singleton second = Singleton.getInstance(); 25. System.out.println("Second instance integer value:"+second.i); 26. } 27. }   228) Write a Java program that prints all the values given at command-line.  **Program**   1. **class** A{ 2. **public** **static** **void** main(String args[]){ 4. **for**(**int** i=0;i<args.length;i++) 5. System.out.println(args[i]); 7. } 8. } 9. compile by > javac A.java 10. run by > java A sonoo jaiswal 1 3 abc   **Output**  sonoo  jaiswal  1  3  abc  229) Which containers use a border layout as their default layout?  The Window, Frame and Dialog classes use a border layout as their default layout.  230) Which containers use a FlowLayout as their default layout?  The Panel and Applet classes use the FlowLayout as their default layout.  231) What are peerless components?  The lightweight component of Swing is called peerless components. Spring has its libraries, so it does not use resources from the Operating System, and hence it has lightweight components.  232) is there is any difference between a Scrollbar and a ScrollPane?  The Scrollbar is a Component whereas the ScrollPane is a Container. A ScrollPane handles its events and performs its scrolling.  233) What is a lightweight component?  Lightweight components are the one which does not go with the native call to obtain the graphical units. They share their parent component graphical units to render them. For example, Swing components, and JavaFX Components.  234) What is a heavyweight component?  The portable elements provided by the operating system are called heavyweight components. AWT is limited to the graphical classes provided by the operating system and therefore, It implements only the minimal subset of screen elements supported by all platforms. The Operating system dependent UI discovery tools are called heavyweight components.  235) What is an applet?  An applet is a small java program that runs inside the browser and generates dynamic content. It is embedded in the webpage and runs on the client side. It is secured and takes less response time. It can be executed by browsers running under many platforms, including Linux, Windows, Mac Os, etc. However, the plugins are required at the client browser to execute the applet. The following image shows the architecture of Applet.  hierarchy of applet  When an applet is created, the following methods are invoked in order.   * init() * start() * paint()   When an applet is destroyed, the following functions are invoked in order.   * stop() * destroy()   236) Can you write a Java class that could be used both as an applet as well as an application?  Yes. Add a main() method to the applet.  Internationalization Interview Questions  237) What is Locale?  A Locale object represents a specific geographical, political, or cultural region. This object can be used to get the locale-specific information such as country name, language, variant, etc.   1. **import** java.util.\*; 2. **public** **class** LocaleExample { 3. **public** **static** **void** main(String[] args) { 4. Locale locale=Locale.getDefault(); 5. //Locale locale=new Locale("fr","fr");//for the specific locale 7. System.out.println(locale.getDisplayCountry()); 8. System.out.println(locale.getDisplayLanguage()); 9. System.out.println(locale.getDisplayName()); 10. System.out.println(locale.getISO3Country()); 11. System.out.println(locale.getISO3Language()); 12. System.out.println(locale.getLanguage()); 13. System.out.println(locale.getCountry()); 15. } 16. }   **Output:**  United States  English  English (United States)  USA  eng  en  US  238)How will you load a specific locale?  By ResourceBundle.getBundle(?) method.  Java Bean Interview Questions  239) What is a JavaBean?  JavaBean is a reusable software component written in the Java programming language, designed to be manipulated visually by a software development environment, like JBuilder or VisualAge for Java. t. A JavaBean encapsulates many objects into one object so that we can access this object from multiple places. Moreover, it provides the easy maintenance. Consider the following example to create a JavaBean class.   1. //Employee.java 2. **package** mypack; 3. **public** **class** Employee **implements** java.io.Serializable{ 4. **private** **int** id; 5. **private** String name; 6. **public** Employee(){} 7. **public** **void** setId(**int** id){**this**.id=id;} 8. **public** **int** getId(){**return** id;} 9. **public** **void** setName(String name){**this**.name=name;} 10. **public** String getName(){**return** name;} 11. }   240) What is the purpose of using the Java bean?  According to Java white paper, it is a reusable software component. A bean encapsulates many objects into one object so that we can access this object from multiple places. Moreover, it provides the easy maintenance.  241) What do you understand by the bean persistent property?  The persistence property of Java bean comes into the act when the properties, fields, and state information are saved to or retrieve from the storage.  RMI Interview Questions  242) What is RMI?  The RMI (Remote Method Invocation) is an API that provides a mechanism to create the distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM. The RMI provides remote communication between the applications using two objects stub and skeleton.  243) What is the purpose of stub and skeleton?  **Stub**  The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes the method on the stub object, it does the following tasks:   * It initiates a connection with remote Virtual Machine (JVM). * It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM). * It waits for the result. * It reads (unmarshals) the return value or exception. * It finally, returns the value to the caller.   **Skeleton**  The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:   * It reads the parameter for the remote method. * It invokes the method on the actual remote object. * It writes and transmits (marshals) the result to the caller.   244) What are the steps involved to write RMI based programs?  There are 6 steps which are performed to write RMI based programs.   * Create the remote interface. * Provide the implementation of the remote interface. * Compile the implementation class and create the stub and skeleton objects using the rmic tool. * Start the registry service by the rmiregistry tool. * Create and start the remote application. * Create and start the client application.   245) What is the use of HTTP-tunneling in RMI?  HTTP tunneling can be defined as the method which doesn't need any setup to work within the firewall environment. It handles the HTTP connections through the proxy servers. However, it does not allow outbound TCP connections.  246) What is JRMP?  JRMP (Java Remote Method Protocol) can be defined as the Java-specific, stream-based protocol which looks up and refers to the remote objects. It requires both client and server to use Java objects. It is wire level protocol which runs under RMI and over TCP/IP.  247) Can RMI and CORBA based applications interact?  Yes, they can. RMI is available with IIOP as the transport protocol instead of JRMP.  Core Java: Data Structure interview questions  248) How to perform Bubble Sort in Java?  Consider the following program to perform Bubble sort in Java.   1. **public** **class** BubbleSort { 2. **public** **static** **void** main(String[] args) { 3. **int**[] a = {10, 9, 7, 101, 23, 44, 12, 78, 34, 23}; 4. **for**(**int** i=0;i<10;i++) 5. { 6. **for** (**int** j=0;j<10;j++) 7. { 8. **if**(a[i]<a[j]) 9. { 10. **int** temp = a[i]; 11. a[i]=a[j]; 12. a[j] = temp; 13. } 14. } 15. } 16. System.out.println("Printing Sorted List ..."); 17. **for**(**int** i=0;i<10;i++) 18. { 19. System.out.println(a[i]); 20. } 21. } 22. }   **Output:**  Printing Sorted List . . .  7  9  10  12  23  34  34  44  78  101  249) How to perform Binary Search in Java?  Consider the following program to perform the binary search in Java.   1. **import** java.util.\*; 2. **public** **class** BinarySearch { 3. **public** **static** **void** main(String[] args) { 4. **int**[] arr = {16, 19, 20, 23, 45, 56, 78, 90, 96, 100}; 5. **int** item, location = -1; 6. System.out.println("Enter the item which you want to search"); 7. Scanner sc = **new** Scanner(System.in); 8. item = sc.nextInt(); 9. location = binarySearch(arr,0,9,item); 10. **if**(location != -1) 11. System.out.println("the location of the item is "+location); 12. **else** 13. System.out.println("Item not found"); 14. } 15. **public** **static** **int** binarySearch(**int**[] a, **int** beg, **int** end, **int** item) 16. { 17. **int** mid; 18. **if**(end >= beg) 19. { 20. mid = (beg + end)/2; 21. **if**(a[mid] == item) 22. { 23. **return** mid+1; 24. } 25. **else** **if**(a[mid] < item) 26. { 27. **return** binarySearch(a,mid+1,end,item); 28. } 29. **else** 30. { 31. **return** binarySearch(a,beg,mid-1,item); 32. } 33. } 34. **return** -1; 35. } 36. }   **Output:**  Enter the item which you want to search  45  the location of the item is 5  250) How to perform Selection Sort in Java?  Consider the following program to perform selection sort in Java.   1. **public** **class** SelectionSort { 2. **public** **static** **void** main(String[] args) { 3. **int**[] a = {10, 9, 7, 101, 23, 44, 12, 78, 34, 23}; 4. **int** i,j,k,pos,temp; 5. **for**(i=0;i<10;i++) 6. { 7. pos = smallest(a,10,i); 8. temp = a[i]; 9. a[i]=a[pos]; 10. a[pos] = temp; 11. } 12. System.out.println("\nprinting sorted elements...\n"); 13. **for**(i=0;i<10;i++) 14. { 15. System.out.println(a[i]); 16. } 17. } 18. **public** **static** **int** smallest(**int** a[], **int** n, **int** i) 19. { 20. **int** small,pos,j; 21. small = a[i]; 22. pos = i; 23. **for**(j=i+1;j<10;j++) 24. { 25. **if**(a[j]<small) 26. { 27. small = a[j]; 28. pos=j; 29. } 30. } 31. **return** pos; 32. } 33. }   **Output:**  printing sorted elements...  7  9  10  12  23  23  34  44  78  101  251) How to perform Linear Search in Java?  Consider the following program to perform Linear search in Java.   1. **import** java.util.Scanner; 3. **public** **class** Leniear\_Search { 4. **public** **static** **void** main(String[] args) { 5. **int**[] arr = {10, 23, 15, 8, 4, 3, 25, 30, 34, 2, 19}; 6. **int** item,flag=0; 7. Scanner sc = **new** Scanner(System.in); 8. System.out.println("Enter Item ?"); 9. item = sc.nextInt(); 10. **for**(**int** i = 0; i<10; i++) 11. { 12. **if**(arr[i]==item) 13. { 14. flag = i+1; 15. **break**; 16. } 17. **else** 18. flag = 0; 19. } 20. **if**(flag != 0) 21. { 22. System.out.println("Item found at location" + flag); 23. } 24. **else** 25. System.out.println("Item not found"); 27. } 28. }   **Output:**  Enter Item ?  23  Item found at location 2  Enter Item ?  22  Item not found  252) How to perform merge sort in Java?  Consider the following program to perform merge sort in Java.   1. **public** **class** MyMergeSort 2. { 3. **void** merge(**int** arr[], **int** beg, **int** mid, **int** end) 4. { 6. **int** l = mid - beg + 1; 7. **int** r = end - mid; 9. intLeftArray[] = **new** **int** [l]; 10. intRightArray[] = **new** **int** [r]; 12. **for** (**int** i=0; i<l; ++i) 13. LeftArray[i] = arr[beg + i]; 15. **for** (**int** j=0; j<r; ++j) 16. RightArray[j] = arr[mid + 1+ j];  19. **int** i = 0, j = 0; 20. **int** k = beg; 21. **while** (i<l&&j<r) 22. { 23. **if** (LeftArray[i] <= RightArray[j]) 24. { 25. arr[k] = LeftArray[i]; 26. i++; 27. } 28. **else** 29. { 30. arr[k] = RightArray[j]; 31. j++; 32. } 33. k++; 34. } 35. **while** (i<l) 36. { 37. arr[k] = LeftArray[i]; 38. i++; 39. k++; 40. } 42. **while** (j<r) 43. { 44. arr[k] = RightArray[j]; 45. j++; 46. k++; 47. } 48. } 50. **void** sort(**int** arr[], **int** beg, **int** end) 51. { 52. **if** (beg<end) 53. { 54. **int** mid = (beg+end)/2; 55. sort(arr, beg, mid); 56. sort(arr , mid+1, end); 57. merge(arr, beg, mid, end); 58. } 59. } 60. **public** **static** **void** main(String args[]) 61. { 62. intarr[] = {90,23,101,45,65,23,67,89,34,23}; 63. MyMergeSort ob = **new** MyMergeSort(); 64. ob.sort(arr, 0, arr.length-1); 66. System.out.println("\nSorted array"); 67. **for**(**int** i =0; i<arr.length;i++) 68. { 69. System.out.println(arr[i]+""); 70. } 71. } 72. }   **Output:**  Sorted array  23  23  23  34  45  65  67  89  90  101  253) How to perform quicksort in Java?  Consider the following program to perform quicksort in Java.   1. **public** **class** QuickSort { 2. **public** **static** **void** main(String[] args) { 3. **int** i; 4. **int**[] arr={90,23,101,45,65,23,67,89,34,23}; 5. quickSort(arr, 0, 9); 6. System.out.println("\n The sorted array is: \n"); 7. **for**(i=0;i<10;i++) 8. System.out.println(arr[i]); 9. } 10. **public** **static** **int** partition(**int** a[], **int** beg, **int** end) 11. { 13. **int** left, right, temp, loc, flag; 14. loc = left = beg; 15. right = end; 16. flag = 0; 17. **while**(flag != 1) 18. { 19. **while**((a[loc] <= a[right]) && (loc!=right)) 20. right--; 21. **if**(loc==right) 22. flag =1; 23. elseif(a[loc]>a[right]) 24. { 25. temp = a[loc]; 26. a[loc] = a[right]; 27. a[right] = temp; 28. loc = right; 29. } 30. **if**(flag!=1) 31. { 32. **while**((a[loc] >= a[left]) && (loc!=left)) 33. left++; 34. **if**(loc==left) 35. flag =1; 36. elseif(a[loc] <a[left]) 37. { 38. temp = a[loc]; 39. a[loc] = a[left]; 40. a[left] = temp; 41. loc = left; 42. } 43. } 44. } 45. returnloc; 46. } 47. **static** **void** quickSort(**int** a[], **int** beg, **int** end) 48. { 50. **int** loc; 51. **if**(beg<end) 52. { 53. loc = partition(a, beg, end); 54. quickSort(a, beg, loc-1); 55. quickSort(a, loc+1, end); 56. } 57. } 58. }   **Output:**  The sorted array is:  23  23  23  34  45  65  67  89  90  101  254) Write a program in Java to create a doubly linked list containing n nodes.  Consider the following program to create a doubly linked list containing n nodes.   1. **public** **class** CountList { 3. //Represent a node of the doubly linked list 5. **class** Node{ 6. **int** data; 7. Node previous; 8. Node next; 10. **public** Node(**int** data) { 11. **this**.data = data; 12. } 13. } 15. //Represent the head and tail of the doubly linked list 16. Node head, tail = **null**; 18. //addNode() will add a node to the list 19. **public** **void** addNode(**int** data) { 20. //Create a new node 21. Node newNode = **new** Node(data); 23. //If list is empty 24. **if**(head == **null**) { 25. //Both head and tail will point to newNode 26. head = tail = newNode; 27. //head's previous will point to null 28. head.previous = **null**; 29. //tail's next will point to null, as it is the last node of the list 30. tail.next = **null**; 31. } 32. **else** { 33. //newNode will be added after tail such that tail's next will point to newNode 34. tail.next = newNode; 35. //newNode's previous will point to tail 36. newNode.previous = tail; 37. //newNode will become new tail 38. tail = newNode; 39. //As it is last node, tail's next will point to null 40. tail.next = **null**; 41. } 42. } 44. //countNodes() will count the nodes present in the list 45. **public** **int** countNodes() { 46. **int** counter = 0; 47. //Node current will point to head 48. Node current = head; 50. **while**(current != **null**) { 51. //Increment the counter by 1 for each node 52. counter++; 53. current = current.next; 54. } 55. **return** counter; 56. } 58. //display() will print out the elements of the list 59. **public** **void** display() { 60. //Node current will point to head 61. Node current = head; 62. **if**(head == **null**) { 63. System.out.println("List is empty"); 64. **return**; 65. } 66. System.out.println("Nodes of doubly linked list: "); 67. **while**(current != **null**) { 68. //Prints each node by incrementing the pointer. 70. System.out.print(current.data + " "); 71. current = current.next; 72. } 73. } 75. **public** **static** **void** main(String[] args) { 77. CountList dList = **new** CountList(); 78. //Add nodes to the list 79. dList.addNode(1); 80. dList.addNode(2); 81. dList.addNode(3); 82. dList.addNode(4); 83. dList.addNode(5); 85. //Displays the nodes present in the list 86. dList.display(); 88. //Counts the nodes present in the given list 89. System.out.println("\nCount of nodes present in the list: " + dList.countNodes()); 90. } 91. }   **Output:**  Nodes of doubly linked list:  1 2 3 4 5  Count of nodes present in the list: 5  255) Write a program in Java to find the maximum and minimum value node from a circular linked list.  Consider the following program.   1. **public** **class** MinMax { 2. //Represents the node of list. 3. **public** **class** Node{ 4. **int** data; 5. Node next; 6. **public** Node(**int** data) { 7. **this**.data = data; 8. } 9. } 11. //Declaring head and tail pointer as null. 12. **public** Node head = **null**; 13. **public** Node tail = **null**; 15. //This function will add the new node at the end of the list. 16. **public** **void** add(**int** data){ 17. //Create new node 18. Node newNode = **new** Node(data); 19. //Checks if the list is empty. 20. **if**(head == **null**) { 21. //If list is empty, both head and tail would point to new node. 22. head = newNode; 23. tail = newNode; 24. newNode.next = head; 25. } 26. **else** { 27. //tail will point to new node. 28. tail.next = newNode; 29. //New node will become new tail. 30. tail = newNode; 31. //Since, it is circular linked list tail will points to head. 32. tail.next = head; 33. } 34. } 36. //Finds out the minimum value node in the list 37. **public** **void** minNode() { 38. Node current = head; 39. //Initializing min to initial node data 40. **int** min = head.data; 41. **if**(head == **null**) { 42. System.out.println("List is empty"); 43. } 44. **else** { 45. **do**{ 46. //If current node's data is smaller than min 47. //Then replace value of min with current node's data 48. **if**(min > current.data) { 49. min = current.data; 50. } 51. current= current.next; 52. }**while**(current != head); 54. System.out.println("Minimum value node in the list: "+ min); 55. } 56. } 58. //Finds out the maximum value node in the list 59. **public** **void** maxNode() { 60. Node current = head; 61. //Initializing max to initial node data 62. **int** max = head.data; 63. **if**(head == **null**) { 64. System.out.println("List is empty"); 65. } 66. **else** { 67. **do**{ 68. //If current node's data is greater than max 69. //Then replace value of max with current node's data 70. **if**(max < current.data) { 71. max = current.data; 72. } 73. current= current.next; 74. }**while**(current != head); 76. System.out.println("Maximum value node in the list: "+ max); 77. } 78. } 80. **public** **static** **void** main(String[] args) { 81. MinMax cl = **new** MinMax(); 82. //Adds data to the list 83. cl.add(5); 84. cl.add(20); 85. cl.add(10); 86. cl.add(1); 87. //Prints the minimum value node in the list 88. cl.minNode(); 89. //Prints the maximum value node in the list 90. cl.maxNode(); 91. } 92. }   **Output:**  Minimum value node in the list: 1  Maximum value node in the list: 20  256) Write a program in Java to calculate the difference between the sum of the odd level and even level nodes of a Binary Tree.  Consider the following program.   1. **import** java.util.LinkedList; 2. **import** java.util.Queue; 4. **public** **class** DiffOddEven { 6. //Represent a node of binary tree 7. **public** **static** **class** Node{ 8. **int** data; 9. Node left; 10. Node right; 12. **public** Node(**int** data){ 13. //Assign data to the new node, set left and right children to null 14. **this**.data = data; 15. **this**.left = **null**; 16. **this**.right = **null**; 17. } 18. } 20. //Represent the root of binary tree 21. **public** Node root; 23. **public** DiffOddEven(){ 24. root = **null**; 25. } 27. //difference() will calculate the difference between sum of odd and even levels of binary tree 28. **public** **int** difference() { 29. **int** oddLevel = 0, evenLevel = 0, diffOddEven = 0; 31. //Variable nodesInLevel keep tracks of number of nodes in each level 32. **int** nodesInLevel = 0; 34. //Variable currentLevel keep track of level in binary tree 35. **int** currentLevel = 0; 37. //Queue will be used to keep track of nodes of tree level-wise 38. Queue<Node> queue = **new** LinkedList<Node>(); 40. //Check if root is null 41. **if**(root == **null**) { 42. System.out.println("Tree is empty"); 43. **return** 0; 44. } 45. **else** { 46. //Add root node to queue as it represents the first level 47. queue.add(root); 48. currentLevel++; 50. **while**(queue.size() != 0) { 52. //Variable nodesInLevel will hold the size of queue i.e. number of elements in queue 53. nodesInLevel = queue.size(); 55. **while**(nodesInLevel > 0) { 56. Node current = queue.remove(); 58. //Checks if currentLevel is even or not. 59. **if**(currentLevel % 2 == 0) 60. //If level is even, add nodes's to variable evenLevel 61. evenLevel += current.data; 62. **else** 63. //If level is odd, add nodes's to variable oddLevel 64. oddLevel += current.data; 66. //Adds left child to queue 67. **if**(current.left != **null**) 68. queue.add(current.left); 69. //Adds right child to queue 70. **if**(current.right != **null**) 71. queue.add(current.right); 72. nodesInLevel--; 73. } 74. currentLevel++; 75. } 76. //Calculates difference between oddLevel and evenLevel 77. diffOddEven = Math.abs(oddLevel - evenLevel); 78. } 79. **return** diffOddEven; 80. } 82. **public** **static** **void** main (String[] args) { 84. DiffOddEven bt = **new** DiffOddEven(); 85. //Add nodes to the binary tree 86. bt.root = **new** Node(1); 87. bt.root.left = **new** Node(2); 88. bt.root.right = **new** Node(3); 89. bt.root.left.left = **new** Node(4); 90. bt.root.right.left = **new** Node(5); 91. bt.root.right.right = **new** Node(6); 93. //Display the difference between sum of odd level and even level nodes 94. System.out.println("Difference between sum of odd level and even level nodes: " + bt.difference()); 95. } 96. }   **Output:**  Difference between sum of odd level and even level nodes: 11  Next Topic[Java Multithreading interview Questions](https://www.javatpoint.com/java-multithreading-interview-questions)  [← Prev](https://www.javatpoint.com/corejava-interview-questions)[Next →](https://www.javatpoint.com/java-multithreading-interview-questions)  [**1**](https://www.javatpoint.com/corejava-interview-questions) [**2**](https://www.javatpoint.com/corejava-interview-questions-2) [**3**](https://www.javatpoint.com/java-multithreading-interview-questions) [**4**](https://www.javatpoint.com/java-collections-interview-questions) [**5**](https://www.javatpoint.com/jdbc-interview-questions)   |  |  | | --- | --- | | [**Java Basics Interview Questions**](https://www.javatpoint.com/corejava-interview-questions#corebasicsinterview) | [**Java OOPs Interview Questions**](https://www.javatpoint.com/corejava-interview-questions#oopsinterview) | | [**Java Multithreading Interview Questions**](https://www.javatpoint.com/java-multithreading-interview-questions) | [**Java String & Exception Interview Questions**](https://www.javatpoint.com/corejava-interview-questions-3) | | [**Java Collection Interview Questions**](https://www.javatpoint.com/java-collections-interview-questions) | [**JDBC Interview Questions**](https://www.javatpoint.com/jdbc-interview-questions) | | [**Servlet Interview Questions**](https://www.javatpoint.com/servletinterview) | [**JSP Interview Questions**](https://www.javatpoint.com/jspinterview) | | [**Spring Interview Questions**](https://www.javatpoint.com/spring-interview-questions) | [**Hibernate Interview Questions**](https://www.javatpoint.com/hibernate-interview-questions) | | [**PL/SQL Interview Questions**](https://www.javatpoint.com/pl-sql-interview-questions) | [**SQL Interview Questions**](https://www.javatpoint.com/sql-interview-questions) | | [**Oracle Interview Questions**](https://www.javatpoint.com/oracle-interview-questions) | [**Android Interview Questions**](https://www.javatpoint.com/android-interview-questions) | | [**SQL Server Interview Questions**](https://www.javatpoint.com/sql-server-interview-questions) | [**MySQL Interview Questions**](https://www.javatpoint.com/mysql-interview-questions) |     [ADVERTISEMENT BY ADRECOVER](https://adrecover.com/?utm_campaign=ByAdRecover&utm_source=www.javatpoint.com&utm_medium=ByAdRecover-336) |

**Javatpoint Services**

JavaTpoint offers too many high quality services. Mail us on hr@javatpoint.com, to get more information about given services.

* Website Designing
* Website Development
* Java Development
* PHP Development
* WordPress
* Graphic Designing
* Logo
* Digital Marketing
* On Page and Off Page SEO
* PPC
* Content Development
* Corporate Training
* Classroom and Online Training
* Data Entry

**Training For College Campus**

JavaTpoint offers college campus training on Core Java, Advance Java, .Net, Android, Hadoop, PHP, Web Technology and Python. Please mail your requirement at hr@javatpoint.com.  
Duration: 1 week to 2 week

Like/Subscribe us for latest updates or newsletter[RSS Feed](https://feeds.feedburner.com/javatpointsonoo) [Subscribe to Get Email Alerts](https://feedburner.google.com/fb/a/mailverify?uri=javatpointsonoo) [Facebook Page](https://www.facebook.com/javatpoint) [Twitter Page](https://twitter.com/pagejavatpoint) [YouTube](https://www.youtube.com/channel/UCUnYvQVCrJoFWZhKK3O2xLg) [Blog Page](https://javatpoint.blogspot.com/)

**LEARN TUTORIALS**

[Learn Java](https://www.javatpoint.com/java-tutorial)[Learn Data Structures](https://www.javatpoint.com/data-structure-tutorial)[Learn C Programming](https://www.javatpoint.com/c-programming-language-tutorial)[Learn C++ Tutorial](https://www.javatpoint.com/cpp-tutorial)[Learn C# Tutorial](https://www.javatpoint.com/c-sharp-tutorial)[Learn PHP Tutorial](https://www.javatpoint.com/php-tutorial)[Learn HTML Tutorial](https://www.javatpoint.com/html-tutorial)[Learn JavaScript Tutorial](https://www.javatpoint.com/javascript-tutorial)[Learn jQuery Tutorial](https://www.javatpoint.com/jquery-tutorial)[Learn Spring Tutorial](https://www.javatpoint.com/spring-tutorial)

**OUR WEBSITES**

[Javatpoint.com](https://www.javatpoint.com/)[Hindi100.com](https://www.hindi100.com/)[Lyricsia.com](https://www.lyricsia.com/)[Quoteperson.com](https://www.quoteperson.com/)[Jobandplacement.com](https://www.jobandplacement.com/)

**OUR SERVICES**

Website Development

Android Development

Website Designing

Digital Marketing

Summer Training

Industrial Training

College Campus Training

**CONTACT**

Address: G-13, 2nd Floor, Sec-3

Noida, UP, 201301, India

Contact No: 0120-4256464, 9990449935

[Contact Us](https://www.javatpoint.com/contact-us)[Subscribe Us](https://www.javatpoint.com/subscribe.jsp)[Privacy Policy](https://www.javatpoint.com/privacy-policy)[Sitemap](https://www.javatpoint.com/sitemap.xml)  
[About Me](https://www.javatpoint.com/sonoo-jaiswal)

© Copyright 2011-2018 www.javatpoint.com. All rights reserved. Developed by JavaTpoint.

⇧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [Javatpoint Logo](https://www.javatpoint.com/)  Top of Form   |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | |  |   Bottom of Form |

* [Home](https://www.javatpoint.com/)
* [Interview Questions](https://www.javatpoint.com/interview-questions-and-answers)
* [Java](https://www.javatpoint.com/java-tutorial)
* [C](https://www.javatpoint.com/c-programming-language-tutorial)
* [SQL](https://www.javatpoint.com/sql-tutorial)
* [Android](https://www.javatpoint.com/android-tutorial)
* [HTML](https://www.javatpoint.com/html-tutorial)
* [XML](https://www.javatpoint.com/xml-tutorial)
* [JavaScript](https://www.javatpoint.com/javascript-tutorial)
* [Ajax](https://www.javatpoint.com/ajax-tutorial)
* [Cloud](https://www.javatpoint.com/cloud-computing-tutorial)
* [Python](https://www.javatpoint.com/python-tutorial)
* [Web Services](https://www.javatpoint.com/web-services-tutorial)
* [Quiz](https://www.javatpoint.com/examaccess)
* [Projects](https://www.javatpoint.com/free-java-projects)
* [Comment](https://www.javatpoint.com/comment.jsp)
* [Forum](https://www.javatpoint.com/forum)
* [Training](https://www.javatpoint.com/training)

**All Interview**

[Interview Tips](https://www.javatpoint.com/interview-tips)[JOB/HR Interview](https://www.javatpoint.com/job-interview-questions)[Competency Interview](https://www.javatpoint.com/competency-based-interview-questions)[Business Analyst](https://www.javatpoint.com/business-analyst-interview-questions)[Behavioral Interview](https://www.javatpoint.com/behavioral-interview-questions)

**Company Interview**

[Company Interview](https://www.javatpoint.com/company-interview-questions-and-recruitment-process)[IBM Interview](https://www.javatpoint.com/ibm-interview-questions)[Infosys Interview](https://www.javatpoint.com/infosys-interview-questions)[Capgemini Interview](https://www.javatpoint.com/capgemini-interview-questions)[Cognizant Interview](https://www.javatpoint.com/cognizant-interview-questions)[Wipro Interview](https://www.javatpoint.com/wipro-interview-questions)[Accenture Interview](https://www.javatpoint.com/accenture-interview-questions)[iGate Interview](https://www.javatpoint.com/igate-interview-questions)[TCS Interview](https://www.javatpoint.com/tcs-interview-questions)[HCL Interview](https://www.javatpoint.com/hcl-interview-questions)[Adobe Interview](https://www.javatpoint.com/adobe-interview-questions)[Microsoft Interview](https://www.javatpoint.com/microsoft-interview-questions)[DXC Technology Interview](https://www.javatpoint.com/dxc-technology-interview-questions)[Ericsson Interview](https://www.javatpoint.com/ericsson-interview-questions)[EXL Service Interview](https://www.javatpoint.com/exl-interview-questions)[IndiaMART Interview](https://www.javatpoint.com/indiamart-interview-questions)[Intuit Interview](https://www.javatpoint.com/intuit-interview-questions)[SpaceX Interview](https://www.javatpoint.com/spacex-interview-questions)[Sapient Interview](https://www.javatpoint.com/sapient-interview-questions)[Amazon Interview](https://www.javatpoint.com/amazon-interview-questions)

**Technical Interview**

[C Interview](https://www.javatpoint.com/c-interview-questions)[C++ Interview](https://www.javatpoint.com/cpp-interview-questions)[Data Structure Interview](https://www.javatpoint.com/data-structure-interview-questions)[Linux Interview](https://www.javatpoint.com/linux-interview-questions)[Unix Interview](https://www.javatpoint.com/unix-interview-questions)[Shell Scripting](https://www.javatpoint.com/shell-scripting-interview-questions)[Networking Interview](https://www.javatpoint.com/networking-interview-questions)[CCNA Interview](https://www.javatpoint.com/ccna-interview-questions)[Android Interview](https://www.javatpoint.com/android-interview-questions)[Cloud Computing](https://www.javatpoint.com/cloud-computing-interview-questions)[Hadoop Interview](https://www.javatpoint.com/hadoop-interview-questions)[Testing/QTP Interview](https://www.javatpoint.com/software-testing-interview-questions)[Selenium Interview](https://www.javatpoint.com/selenium-interview-questions)[Web Services Interview](https://www.javatpoint.com/web-services-interview-questions)[OS Interview](https://www.javatpoint.com/operating-system-interview-questions)[Excel Interview](https://www.javatpoint.com/excel-interview-questions)[SEO Interview](https://www.javatpoint.com/seo-interview-questions)[Digital Marketing](https://www.javatpoint.com/digital-marketing-interview-questions-and-answers)[Python Interview](https://www.javatpoint.com/python-interview-questions)[Django Interview](https://www.javatpoint.com/django-interview-questions-and-answers)[Pascal Interview](https://www.javatpoint.com/pascal-interview-questions-and-answers)[Ruby Interview](https://www.javatpoint.com/ruby-interview-questions)[Ruby on Rails Interview](https://www.javatpoint.com/ruby-on-rails-interview-questions)[Memcached Interview](https://www.javatpoint.com/memcached-interview-questions-and-answers)[Go Interview](https://www.javatpoint.com/go-programming-interview-questions)[OpenStack Interview](https://www.javatpoint.com/openstack-interview-questions-and-answers)[Scala Interview](https://www.javatpoint.com/scala-interview-questions)[Control Systems](https://www.javatpoint.com/control-systems-interview-questions)[Electrical Machines](https://www.javatpoint.com/electrical-machines-interview-questions)[Power System](https://www.javatpoint.com/power-system-interview-questions)[Digital Electronics](https://www.javatpoint.com/digital-electronics-interview-questions)[Robotics Interview](https://www.javatpoint.com/robotics-interview-questions)[TypeScript Interview](https://www.javatpoint.com/typescript-interview-questions)[Swift Interview](https://www.javatpoint.com/swift-interview-questions)[Blockchain Interview](https://www.javatpoint.com/blockchain-interview-questions)[Bitcoin Interview](https://www.javatpoint.com/bitcoin-interview-questions)[AWS Interview](https://www.javatpoint.com/aws-interview-questions)[Informatica Interview](https://www.javatpoint.com/informatica-interview-questions)[QA Interview](https://www.javatpoint.com/qa-interview-questions)[React Interview](https://www.javatpoint.com/react-interview-questions)[GIT Interview](https://www.javatpoint.com/git-interview-questions)[J2EE Interview](https://www.javatpoint.com/j2ee-interview-questions)[Deep Learning](https://www.javatpoint.com/deep-learning-interview-questions)[Salesforce Interview](https://www.javatpoint.com/salesforce-interview-questions)[TestNG Interview](https://www.javatpoint.com/testng-interview-questions)[Agile Interview](https://www.javatpoint.com/agile-interview-questions)[Machine Learning](https://www.javatpoint.com/machine-learning-interview-questions)[API Testing](https://www.javatpoint.com/api-testing-interview-questions)[Algorithm](https://www.javatpoint.com/algorithm-interview-questions)[Java Design Pattern](https://www.javatpoint.com/java-design-pattern-interview-questions)[Teradata](https://www.javatpoint.com/teradata-interview-questions)[TensorFlow](https://www.javatpoint.com/tensorflow-interview-questions)[LINQ](https://www.javatpoint.com/linq-interview-questions)[Mainframe](https://www.javatpoint.com/mainframe-interview-questions)[Laravel](https://www.javatpoint.com/laravel-interview-questions)[JIRA](https://www.javatpoint.com/jira-interview-questions)[JCL](https://www.javatpoint.com/jcl-interview-questions)[Pytorch](https://www.javatpoint.com/pytorch-interview-questions)[ETL Testing](https://www.javatpoint.com/etl-testing-interview-questions)[Linked List](https://www.javatpoint.com/linked-list-interview-questions)[DataStage](https://www.javatpoint.com/datastage-interview-questions)[Tableau](https://www.javatpoint.com/tableau-interview-questions)[SAP ABAP](https://www.javatpoint.com/sap-abap-interview-questions)[DAA](https://www.javatpoint.com/daa-interview-questions)[Computer Graphics](https://www.javatpoint.com/computer-graphics-interview-questions)[Data Science](https://www.javatpoint.com/data-science-interview-questions)[Software Engineering](https://www.javatpoint.com/software-engineering-interview-questions)[Power BI](https://www.javatpoint.com/power-bi-interview-questions)[Xamarin](https://www.javatpoint.com/xamarin-interview-questions)[Ansible](https://www.javatpoint.com/ansible-interview-questions)[Data Warehouse](https://www.javatpoint.com/data-warehouse-interview-questions)[Ionic](https://www.javatpoint.com/ionic-interview-questions)[Devops](https://www.javatpoint.com/devops-interview-questions)[React Native](https://www.javatpoint.com/react-native-interview-questions)[Python Pandas](https://www.javatpoint.com/python-pandas-interview-questions)[HTTP Interview](https://www.javatpoint.com/http-interview-questions)[Kali Linux Interview](https://www.javatpoint.com/kali-linux-interview-questions)[PowerShell Interview](https://www.javatpoint.com/powershell-interview-questions)[SharePoint Interview](https://www.javatpoint.com/sharepoint-interview-questions)[Talend Interview](https://www.javatpoint.com/talend-interview-questions)[Microsoft Azure Interview](https://www.javatpoint.com/microsoft-azure-interview-questions)[R Interview](https://www.javatpoint.com/r-interview-questions)[Flutter Interview](https://www.javatpoint.com/flutter-interview-questions)[MATLAB Interview](https://www.javatpoint.com/matlab-interview-questions)[ES6 Interview](https://www.javatpoint.com/es6-interview-questions)[RPA Interview](https://www.javatpoint.com/rpa-interview-questions)[Desktop Support Interview](https://www.javatpoint.com/desktop-support-interview-questions)[Angular 8 Interview](https://www.javatpoint.com/angular-8-interview-questions)[GraphQL Interview](https://www.javatpoint.com/graphql-interview-questions)[Vue.js Interview](https://www.javatpoint.com/vue-js-interview-questions)[Automation Anywhere Interview](https://www.javatpoint.com/automation-anywhere-interview-questions)[AI Interview](https://www.javatpoint.com/artificial-intelligence-interview-questions)[UiPath Interview](https://www.javatpoint.com/uipath-interview-questions)[Blue Prism Interview](https://www.javatpoint.com/blue-prism-interview-questions)[SSIS Interview](https://www.javatpoint.com/ssis-interview-questions)[Mobile Computing Interview](https://www.javatpoint.com/mobile-computing-interview-questions)[ITIL Interview](https://www.javatpoint.com/itil-interview-questions)[SAS Interview](https://www.javatpoint.com/sas-interview-questions)[Elasticsearch Interview](https://www.javatpoint.com/elasticsearch-interview-questions)[Entity Framework Interview](https://www.javatpoint.com/entity-framework-interview-questions)[RxJS Interview](https://www.javatpoint.com/rxjs-interview-questions)[Accounting Interview](https://www.javatpoint.com/accounting-interview-questions)[Electron.js Interview](https://www.javatpoint.com/electronjs-interview-questions)[Knockout JS Interview](https://www.javatpoint.com/knockout-js-interview-questions)[Top Angular Interview](https://www.javatpoint.com/top-angular-interview-questions)[IAS Interview Question](https://www.javatpoint.com/ias-interview-questions)[Array Interview Questions in Java](https://www.javatpoint.com/array-interview-questions-in-java)[Banking Interview](https://www.javatpoint.com/banking-interview-questions)[Interview Questions for Freshers](https://www.javatpoint.com/interview-questions-for-freshers)[PowerPoint Interview](https://www.javatpoint.com/powerpoint-interview-questions)[BPO Interview](https://www.javatpoint.com/bpo-interview-questions)[Civil Engineering Interview](https://www.javatpoint.com/civil-engineering-interview-questions)[Apache Spark Interview](https://www.javatpoint.com/apache-spark-interview-questions)[Data Mining Interview](https://www.javatpoint.com/data-mining-interview-questions)[Java Support Interview](https://www.javatpoint.com/java-support-interview-questions)[Kubernetes Interview](https://www.javatpoint.com/kubernetes-interview-questions)[Web API Interview](https://www.javatpoint.com/web-api-interview-questions)[Scrum Master Interview](https://www.javatpoint.com/scrum-master-interview-questions)[System Design Interview](https://www.javatpoint.com/system-design-interview-questions)[SSB Interview Questions](https://www.javatpoint.com/ssb-interview-questions)[Teachers Interview Questions](https://www.javatpoint.com/teachers-interview-questions)[Finance Interview Questions](https://www.javatpoint.com/finance-interview-questions)[Technical Support Interview](https://www.javatpoint.com/technical-support-interview-questions)[Active Directory Interview](https://www.javatpoint.com/active-directory-interview-questions)[Appium Interview](https://www.javatpoint.com/appium-interview-questions)[Cucumber Interview](https://www.javatpoint.com/cucumber-interview-questions)[Full-Stack Developer](https://www.javatpoint.com/full-stack-developer-interview-questions-java)

**Web Interview**

[HTML Interview](https://www.javatpoint.com/html-interview-questions)[CSS Interview](https://www.javatpoint.com/css-interview-questions)[JavaScript Interview](https://www.javatpoint.com/javascript-interview-questions)[jQuery Interview](https://www.javatpoint.com/jquery-interview-questions)[AngularJS Interview](https://www.javatpoint.com/angularjs-interview-questions)[Angular Interview](https://www.javatpoint.com/angular-interview-questions)[Angular 7 Interview](https://www.javatpoint.com/angular-7-interview-questions)[Node.js Interview](https://www.javatpoint.com/node-js-interview-questions)[AJAX Interview](https://www.javatpoint.com/ajax-interview-questions)[XML Interview](https://www.javatpoint.com/xml-interview-questions)[Dojo Interview](https://www.javatpoint.com/dojo-interview-questions-and-answers)[Backbone.js Interview](https://www.javatpoint.com/backbone-js-interview-questions-and-answers)[Ember.js Interview](https://www.javatpoint.com/ember-js-interview-questions)[XHTML Interview](https://www.javatpoint.com/xhtml-interview-questions)[XSLT Interview](https://www.javatpoint.com/xslt-interview-questions)[XPath Interview](https://www.javatpoint.com/xpath-interview-questions)[XQuery Interview](https://www.javatpoint.com/xquery-interview-questions)[XForms Interview](https://www.javatpoint.com/xforms-interview-questions)[SASS Interview](https://www.javatpoint.com/sass-interview-questions-and-answers)[CoffeeScript Interview](https://www.javatpoint.com/coffeescript-interview-questions-and-answers)[LESS Interview](https://www.javatpoint.com/less-interview-questions-and-answers)[Perl Interview](https://www.javatpoint.com/perl-interview-questions)[Pure.CSS Interview](https://www.javatpoint.com/pure-css-interview-questions)[Materialize Interview](https://www.javatpoint.com/materialize-css-interview-questions)[Framework7 Interview](https://www.javatpoint.com/framework7-interview-questions)[SVG Interview](https://www.javatpoint.com/svg-interview-questions)

**PHP Interview**

[PHP Interview](https://www.javatpoint.com/php-interview-questions)[Wordpress Interview](https://www.javatpoint.com/wordpress-interview-questions)[Joomla Interview](https://www.javatpoint.com/joomla-interview-questions-and-answers)[Drupal Interview](https://www.javatpoint.com/drupal-interview-questions-and-answers)[Magento Interview](https://www.javatpoint.com/magento-interview-questions-and-answers)[CodeIgniter Interview](https://www.javatpoint.com/codeigniter-interview-questions-and-answers)[Phalcon Interview](https://www.javatpoint.com/phalcon-interview-questions)

**.Net Interview**

[.Net Interview](https://www.javatpoint.com/dot-net-interview-questions)[C# Interview](https://www.javatpoint.com/c-sharp-interview-questions)[ASP.NET Interview](https://www.javatpoint.com/asp-dot-net-interview-questions)[ADO.NET Interview](https://www.javatpoint.com/ado-dot-net-interview-questions)[WCF Interview](https://www.javatpoint.com/wcf-interview-questions)[WPF Interview](https://www.javatpoint.com/wpf-interview-questions)[Silverlight Interview](https://www.javatpoint.com/silverlight-interview-questions-and-answers)[F# Interview](https://www.javatpoint.com/fsharp-interview-questions)

**Java Interview**

[Java 1 Interview](https://www.javatpoint.com/corejava-interview-questions)[Java 2 Interview](https://www.javatpoint.com/corejava-interview-questions-2)[Exception & String](https://www.javatpoint.com/corejava-interview-questions-3)[**Multithreading**[open link](https://www.javatpoint.com/java-multithreading-interview-questions)](https://www.javatpoint.com/java-multithreading-interview-questions)[Collections Interview](https://www.javatpoint.com/java-collections-interview-questions)[JDBC Interview](https://www.javatpoint.com/jdbc-interview-questions)[Servlet Interview](https://www.javatpoint.com/servletinterview)[JSP Interview](https://www.javatpoint.com/jspinterview)[EJB Interview](https://www.javatpoint.com/ejb-interview-questions)[Struts Interview](https://www.javatpoint.com/struts-interview-questions)[Hibernate Interview](https://www.javatpoint.com/hibernate-interview-questions)[Spring Interview](https://www.javatpoint.com/spring-interview-questions)[Spring Boot Interview](https://www.javatpoint.com/spring-boot-interview-questions)[Spring MVC Interview](https://www.javatpoint.com/spring-mvc-interview-questions)[JPA Interview](https://www.javatpoint.com/jpa-interview-questions)[Maven Interview](https://www.javatpoint.com/maven-interview-questions)[JUnit Interview](https://www.javatpoint.com/junit-interview-questions)[JSF Interview](https://www.javatpoint.com/jsf-interview-questions)[PrimeFaces Interview](https://www.javatpoint.com/primefaces-interview-questions)[RichFaces Interview](https://www.javatpoint.com/richfaces-interview-questions)[GWT Interview](https://www.javatpoint.com/gwt-interview-questions)[JDB Interview](https://www.javatpoint.com/jdb-interview-questions)[JOGL Interview](https://www.javatpoint.com/jogl-interview-questions)[jBPM Interview](https://www.javatpoint.com/jbpm-interview-questions)[Jenkins Interview](https://www.javatpoint.com/jenkins-interview-questions)[Java Constructor Interview Questions](https://www.javatpoint.com/java-constructor-interview-questions)[Java Technical Architect Interview Questions](https://www.javatpoint.com/java-technical-architect-interview-questions)

**Database Interview**

[DBMS Interview](https://www.javatpoint.com/dbms-interview-questions)[SQL Interview](https://www.javatpoint.com/sql-interview-questions)[PL/SQL Interview](https://www.javatpoint.com/pl-sql-interview-questions)[Oracle Interview](https://www.javatpoint.com/oracle-interview-questions)[MySql Interview](https://www.javatpoint.com/mysql-interview-questions)[SQL Server Interview](https://www.javatpoint.com/sql-server-interview-questions)[MongoDB Interview](https://www.javatpoint.com/mongodb-interview-questions)[Cassandra Interview](https://www.javatpoint.com/cassandra-interview-questions-and-answers)[DB2 Interview](https://www.javatpoint.com/db2-interview-questions-and-answers)[Access Interview](https://www.javatpoint.com/microsoft-access-interview-questions-and-answers)[SQLite Interview](https://www.javatpoint.com/sqlite-interview-questions)[CouchDB Interview](https://www.javatpoint.com/couchdb-interview-questions)[Neo4j Interview](https://www.javatpoint.com/neo4j-interview-questions-and-answers)[MariaDB Interview](https://www.javatpoint.com/mariadb-interview-questions)[PouchDB Interview](https://www.javatpoint.com/pouchdb-interview-questions)[Redis Interview](https://www.javatpoint.com/redis-interview-questions-and-answers)

[ADS BY ADRECOVER](https://adrecover.com/?utm_campaign=ByAdRecover&utm_source=www.javatpoint.com&utm_medium=ByAdRecover-160)

[ADVERTISEMENT BY ADRECOVER](https://adrecover.com/?utm_campaign=ByAdRecover&utm_source=www.javatpoint.com&utm_medium=ByAdRecover-728)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Java Multithreading and Concurrency Interview Questions  Multithreading and Synchronization are considered as the typical chapter in java programming. In game development companies, multithreading related interview questions are asked mostly. A list of frequently asked java multithreading and concurrency interview questions is given below.  Multithreading Interview Questions  1) What is multithreading?  Multithreading is a process of executing multiple threads simultaneously. Multithreading is used to obtain the multitasking. It consumes less memory and gives the fast and efficient performance. Its main advantages are:   * Threads share the same address space. * The thread is lightweight. * The cost of communication between the processes is low.   [More details.](https://www.javatpoint.com/multithreading)  2) What is the thread?  A thread is a lightweight subprocess. It is a separate path of execution because each thread runs in a different stack frame. A process may contain multiple threads. Threads share the process resources, but still, they execute independently.  [More details.](https://www.javatpoint.com/multithreading)  3) Differentiate between process and thread?  There are the following differences between the process and thread.   * A Program in the execution is called the process whereas; A thread is a subset of the process * Processes are independent whereas threads are the subset of process. * Process have different address space in memory, while threads contain a shared address space. * Context switching is faster between the threads as compared to processes. * Inter-process communication is slower and expensive than inter-thread communication. * Any change in Parent process doesn't affect the child process whereas changes in parent thread can affect the child thread.   Java Multithreading  4) What do you understand by inter-thread communication?   * The process of communication between synchronized threads is termed as inter-thread communication. * Inter-thread communication is used to avoid thread polling in Java. * The thread is paused running in its critical section, and another thread is allowed to enter (or lock) in the same critical section to be executed. * It can be obtained by wait(), notify(), and notifyAll() methods.   5) What is the purpose of wait() method in Java?  The wait() method is provided by the Object class in Java. This method is used for inter-thread communication in Java. The java.lang.Object.wait() is used to pause the current thread, and wait until another thread does not call the notify() or notifyAll() method. Its syntax is given below.  public final void wait()  6) Why must wait() method be called from the synchronized block?  We must call the wait method otherwise it will throw **java.lang.IllegalMonitorStateException** exception. Moreover, we need wait() method for inter-thread communication with notify() and notifyAll(). Therefore It must be present in the synchronized block for the proper and correct communication.  7) What are the advantages of multithreading?  Multithreading programming has the following advantages:   * Multithreading allows an application/program to be always reactive for input, even already running with some background tasks * Multithreading allows the faster execution of tasks, as threads execute independently. * Multithreading provides better utilization of cache memory as threads share the common memory resources. * Multithreading reduces the number of the required server as one server can execute multiple threads at a time.   8) What are the states in the lifecycle of a Thread?  A thread can have one of the following states during its lifetime:   1. **New:** In this state, a Thread class object is created using a new operator, but the thread is not alive. Thread doesn't start until we call the start() method. 2. **Runnable:** In this state, the thread is ready to run after calling the start() method. However, the thread is not yet selected by the thread scheduler. 3. **Running:** In this state, the thread scheduler picks the thread from the ready state, and the thread is running. 4. **Waiting/Blocked:** In this state, a thread is not running but still alive, or it is waiting for the other thread to finish. 5. **Dead/Terminated:** A thread is in terminated or dead state when the run() method exits.   Java thread life cycle  9) What is the difference between preemptive scheduling and time slicing?  Under preemptive scheduling, the highest priority task executes until it enters the waiting or dead states or a higher priority task comes into existence. Under time slicing, a task executes for a predefined slice of time and then reenters the pool of ready tasks. The scheduler then determines which task should execute next, based on priority and other factors.  10) What is context switching?  In Context switching the state of the process (or thread) is stored so that it can be restored and execution can be resumed from the same point later. Context switching enables the multiple processes to share the same CPU.  11) Differentiate between the Thread class and Runnable interface for creating a Thread?  The Thread can be created by using two ways.   * By extending the Thread class * By implementing the Thread class   However, the primary differences between both the ways are given below:   * By extending the Thread class, we cannot extend any other class, as Java does not allow multiple inheritances while implementing the Runnable interface; we can also extend other base class(if required). * By extending the Thread class, each of thread creates the unique object and associates with it while implementing the Runnable interface; multiple threads share the same object * Thread class provides various inbuilt methods such as getPriority(), isAlive and many more while the Runnable interface provides a single method, i.e., run().   12) What does join() method?  The join() method waits for a thread to die. In other words, it causes the currently running threads to stop executing until the thread it joins with completes its task. Join method is overloaded in Thread class in the following ways.   * public void join()throws InterruptedException * public void join(long milliseconds)throws InterruptedException   [More details.](https://www.javatpoint.com/join()-method)  13) Describe the purpose and working of sleep() method.  The sleep() method in java is used to block a thread for a particular time, which means it pause the execution of a thread for a specific time. There are two methods of doing so.  **Syntax:**   * public static void sleep(long milliseconds)throws InterruptedException * public static void sleep(long milliseconds, int nanos)throws InterruptedException   **Working of sleep() method**  When we call the sleep() method, it pauses the execution of the current thread for the given time and gives priority to another thread(if available). Moreover, when the waiting time completed then again previous thread changes its state from waiting to runnable and comes in running state, and the whole process works so on till the execution doesn't complete.  14) What is the difference between wait() and sleep() method?   |  |  | | --- | --- | | **wait()** | **sleep()** | | 1) The wait() method is defined in Object class. | The sleep() method is defined in Thread class. | | 2) The wait() method releases the lock. | The sleep() method doesn't release the lock. |   15) Is it possible to start a thread twice?  No, we cannot restart the thread, as once a thread started and executed, it goes to the Dead state. Therefore, if we try to start a thread twice, it will give a runtimeException "java.lang.IllegalThreadStateException". Consider the following example.   1. **public** **class** Multithread1 **extends** Thread 2. { 3. **public** **void** run() 4. { 5. **try** { 6. System.out.println("thread is executing now........"); 7. } **catch**(Exception e) { 8. } 9. } 10. **public** **static** **void** main (String[] args) { 11. Multithread1 m1= **new** Multithread1(); 12. m1.start(); 13. m1.start(); 14. } 15. }   **Output**  thread is executing now........  Exception in thread "main" java.lang.IllegalThreadStateException  at java.lang.Thread.start(Thread.java:708)  at Multithread1.main(Multithread1.java:13)  [More details.](https://www.javatpoint.com/can-we-start-a-thread-twice)  16) Can we call the run() method instead of start()?  Yes, calling run() method directly is valid, but it will not work as a thread instead it will work as a normal object. There will not be context-switching between the threads. When we call the start() method, it internally calls the run() method, which creates a new stack for a thread while directly calling the run() will not create a new stack.  [More details.](https://www.javatpoint.com/what-if-we-call-run()-method-directly)  17) What about the daemon threads?  The daemon threads are the low priority threads that provide the background support and services to the user threads. Daemon thread gets automatically terminated by the JVM if the program remains with the daemon thread only, and all other user threads are ended/died. There are two methods for daemon thread available in the Thread class:   * **public void setDaemon(boolean status):** It used to mark the thread daemon thread or a user thread. * **public boolean isDaemon():** It checks the thread is daemon or not.   [More details.](https://www.javatpoint.com/daemon-thread)  18)Can we make the user thread as daemon thread if the thread is started?  No, if you do so, it will throw IllegalThreadStateException. Therefore, we can only create a daemon thread before starting the thread.   1. **class** Testdaemon1 **extends** Thread{ 2. **public** **void** run(){ 3. System.out.println("Running thread is daemon..."); 4. } 5. **public** **static** **void** main (String[] args) { 6. Testdaemon1 td= **new** Testdaemon1(); 7. td.start(); 8. setDaemon(**true**);// It will throw the exception: td. 9. } 10. }   **Output**  Running thread is daemon...  Exception in thread "main" java.lang.IllegalThreadStateException  at java.lang.Thread.setDaemon(Thread.java:1359)  at Testdaemon1.main(Testdaemon1.java:8)  [More details.](https://www.javatpoint.com/daemon-thread)  19)What is shutdown hook?  The shutdown hook is a thread that is invoked implicitly before JVM shuts down. So we can use it to perform clean up the resource or save the state when JVM shuts down normally or abruptly. We can add shutdown hook by using the following method:   1. **public** **void** addShutdownHook(Thread hook){} 2. Runtime r=Runtime.getRuntime(); 3. r.addShutdownHook(**new** MyThread());   Some important points about shutdown hooks are :   * Shutdown hooks initialized but can only be started when JVM shutdown occurred. * Shutdown hooks are more reliable than the finalizer() because there are very fewer chances that shutdown hooks not run. * The shutdown hook can be stopped by calling the halt(int) method of Runtime class.   [More details.](https://www.javatpoint.com/ShutdownHook-thread)  20)When should we interrupt a thread?  We should interrupt a thread when we want to break out the sleep or wait state of a thread. We can interrupt a thread by calling the interrupt() throwing the InterruptedException.  [More details.](https://www.javatpoint.com/interrupting-a-thread)  21) What is the synchronization?  Synchronization is the capability to control the access of multiple threads to any shared resource. It is used:   1. To prevent thread interference. 2. To prevent consistency problem.   When the multiple threads try to do the same task, there is a possibility of an erroneous result, hence to remove this issue, Java uses the process of synchronization which allows only one thread to be executed at a time. Synchronization can be achieved in three ways:   * by the synchronized method * by synchronized block * by static synchronization   Syntax for synchronized block   1. **synchronized**(object reference expression) 2. { 3. //code block 4. }   [More details.](https://www.javatpoint.com/synchronization)  22) What is the purpose of the Synchronized block?  The Synchronized block can be used to perform synchronization on any specific resource of the method. Only one thread at a time can execute on a particular resource, and all other threads which attempt to enter the synchronized block are blocked.   * Synchronized block is used to lock an object for any shared resource. * The scope of the synchronized block is limited to the block on which, it is applied. Its scope is smaller than a method.   [More details.](https://www.javatpoint.com/synchronized-block-example)  23)Can Java object be locked down for exclusive use by a given thread?  Yes. You can lock an object by putting it in a "synchronized" block. The locked object is inaccessible to any thread other than the one that explicitly claimed it.  24) What is static synchronization?  If you make any static method as synchronized, the lock will be on the class not on the object. If we use the synchronized keyword before a method so it will lock the object (one thread can access an object at a time) but if we use static synchronized so it will lock a class (one thread can access a class at a time). [More details.](https://www.javatpoint.com/static-synchronization-example)  25)What is the difference between notify() and notifyAll()?  The notify() is used to unblock one waiting thread whereas notifyAll() method is used to unblock all the threads in waiting state.  26)What is the deadlock?  Deadlock is a situation in which every thread is waiting for a resource which is held by some other waiting thread. In this situation, Neither of the thread executes nor it gets the chance to be executed. Instead, there exists a universal waiting state among all the threads. Deadlock is a very complicated situation which can break our code at runtime.  [More details.](https://www.javatpoint.com/deadlock-in-java)  27) How to detect a deadlock condition? How can it be avoided?  We can detect the deadlock condition by running the code on cmd and collecting the Thread Dump, and if any deadlock is present in the code, then a message will appear on cmd.  **Ways to avoid the deadlock condition in Java:**   * **Avoid Nested lock:** Nested lock is the common reason for deadlock as deadlock occurs when we provide locks to various threads so we should give one lock to only one thread at some particular time. * **Avoid unnecessary locks:** we must avoid the locks which are not required. * **Using thread join:** Thread join helps to wait for a thread until another thread doesn't finish its execution so we can avoid deadlock by maximum use of join method.   28) What is Thread Scheduler in java?  In Java, when we create the threads, they are supervised with the help of a Thread Scheduler, which is the part of JVM. Thread scheduler is only responsible for deciding which thread should be executed. Thread scheduler uses two mechanisms for scheduling the threads: Preemptive and Time Slicing.  Java thread scheduler also works for deciding the following for a thread:   * It selects the priority of the thread. * It determines the waiting time for a thread * It checks the Nature of thread   29) Does each thread have its stack in multithreaded programming?  Yes, in multithreaded programming every thread maintains its own or separate stack area in memory due to which every thread is independent of each other.  30) How is the safety of a thread achieved?  If a method or class object can be used by multiple threads at a time without any race condition, then the class is thread-safe. Thread safety is used to make a program safe to use in multithreaded programming. It can be achieved by the following ways:   * Synchronization * Using Volatile keyword * Using a lock based mechanism * Use of atomic wrapper classes   31) What is race-condition?  A Race condition is a problem which occurs in the multithreaded programming when various threads execute simultaneously accessing a shared resource at the same time. The proper use of synchronization can avoid the Race condition.  32) What is the volatile keyword in java?  Volatile keyword is used in multithreaded programming to achieve the thread safety, as a change in one volatile variable is visible to all other threads so one variable can be used by one thread at a time.  33) What do you understand by thread pool?   * Java Thread pool represents a group of worker threads, which are waiting for the task to be allocated. * Threads in the thread pool are supervised by the service provider which pulls one thread from the pool and assign a job to it. * After completion of the given task, thread again came to the thread pool. * The size of the thread pool depends on the total number of threads kept at reserve for execution.   The advantages of the thread pool are :   * Using a thread pool, performance can be enhanced. * Using a thread pool, better system stability can occur.   Concurrency Interview Questions  34) What are the main components of concurrency API?  Concurrency API can be developed using the class and interfaces of java.util.Concurrent package. There are the following classes and interfaces in java.util.Concurrent package.   * Executor * FarkJoinPool * ExecutorService * ScheduledExecutorService * Future * TimeUnit(Enum) * CountDownLatch * CyclicBarrier * Semaphore * ThreadFactory * BlockingQueue * DelayQueue * Locks * Phaser   35) What is the Executor interface in Concurrency API in Java?  The Executor Interface provided by the package java.util.concurrent is the simple interface used to execute the new task. The execute() method of Executor interface is used to execute some given command. The syntax of the execute() method is given below.  **void execute(Runnable command)**  Consider the following example:   1. **import** java.util.concurrent.Executor; 2. **import** java.util.concurrent.Executors; 3. **import** java.util.concurrent.ThreadPoolExecutor; 4. **import** java.util.concurrent.TimeUnit; 6. **public** **class** TestThread { 7. **public** **static** **void** main(**final** String[] arguments) **throws** InterruptedException { 8. Executor e = Executors.newCachedThreadPool(); 9. e.execute(**new** Thread()); 10. ThreadPoolExecutor pool = (ThreadPoolExecutor)e; 11. pool.shutdown(); 12. } 14. **static** **class** Thread **implements** Runnable { 15. **public** **void** run() { 16. **try** { 17. Long duration = (**long**) (Math.random() \* 5); 18. System.out.println("Running Thread!"); 19. TimeUnit.SECONDS.sleep(duration); 20. System.out.println("Thread Completed"); 21. } **catch** (InterruptedException ex) { 22. ex.printStackTrace(); 23. } 24. } 25. } 26. }   **Output**  Running Thread!  Thread Completed  36) What is BlockingQueue?  The java.util.concurrent.BlockingQueue is the subinterface of Queue that supports the operations such as waiting for the space availability before inserting a new value or waiting for the queue to become non-empty before retrieving an element from it. Consider the following example.    2. **import** java.util.Random; 3. **import** java.util.concurrent.ArrayBlockingQueue; 4. **import** java.util.concurrent.BlockingQueue; 6. **public** **class** TestThread { 8. **public** **static** **void** main(**final** String[] arguments) **throws** InterruptedException { 9. BlockingQueue<Integer> queue = **new** ArrayBlockingQueue<Integer>(10); 11. Insert i = **new** Insert(queue); 12. Retrieve r = **new** Retrieve(queue); 14. **new** Thread(i).start(); 15. **new** Thread(r).start(); 17. Thread.sleep(2000); 18. }  21. **static** **class** Insert **implements** Runnable { 22. **private** BlockingQueue<Integer> queue; 24. **public** Insert(BlockingQueue queue) { 25. **this**.queue = queue; 26. } 28. @Override 29. **public** **void** run() { 30. Random random = **new** Random(); 32. **try** { 33. **int** result = random.nextInt(200); 34. Thread.sleep(1000); 35. queue.put(result); 36. System.out.println("Added: " + result); 38. result = random.nextInt(10); 39. Thread.sleep(1000); 40. queue.put(result); 41. System.out.println("Added: " + result); 43. result = random.nextInt(50); 44. Thread.sleep(1000); 45. queue.put(result); 46. System.out.println("Added: " + result); 47. } **catch** (InterruptedException e) { 48. e.printStackTrace(); 49. } 50. } 51. } 53. **static** **class** Retrieve **implements** Runnable { 54. **private** BlockingQueue<Integer> queue; 56. **public** Retrieve(BlockingQueue queue) { 57. **this**.queue = queue; 58. } 60. @Override 61. **public** **void** run() { 63. **try** { 64. System.out.println("Removed: " + queue.take()); 65. System.out.println("Removed: " + queue.take()); 66. System.out.println("Removed: " + queue.take()); 67. } **catch** (InterruptedException e) { 68. e.printStackTrace(); 69. } 70. } 71. } 72. }   **Output**  Added: 96  Removed: 96  Added: 8  Removed: 8  Added: 5  Removed: 5  37) How to implement producer-consumer problem by using BlockingQueue?  The producer-consumer problem can be solved by using BlockingQueue in the following way.    2. **import** java.util.concurrent.BlockingQueue; 3. **import** java.util.concurrent.LinkedBlockingQueue; 4. **import** java.util.logging.Level; 5. **import** java.util.logging.Logger; 6. **public** **class** ProducerConsumerProblem { 7. **public** **static** **void** main(String args[]){ 8. //Creating shared object 9. BlockingQueue sharedQueue = **new** LinkedBlockingQueue(); 11. //Creating Producer and Consumer Thread 12. Thread prod = **new** Thread(**new** Producer(sharedQueue)); 13. Thread cons = **new** Thread(**new** Consumer(sharedQueue)); 15. //Starting producer and Consumer thread 16. prod.start(); 17. cons.start(); 18. } 20. } 22. //Producer Class in java 23. **class** Producer **implements** Runnable { 25. **private** **final** BlockingQueue sharedQueue; 27. **public** Producer(BlockingQueue sharedQueue) { 28. **this**.sharedQueue = sharedQueue; 29. } 31. @Override 32. **public** **void** run() { 33. **for**(**int** i=0; i<10; i++){ 34. **try** { 35. System.out.println("Produced: " + i); 36. sharedQueue.put(i); 37. } **catch** (InterruptedException ex) { 38. Logger.getLogger(Producer.**class**.getName()).log(Level.SEVERE, **null**, ex); 39. } 40. } 41. } 43. } 45. //Consumer Class in Java 46. **class** Consumer **implements** Runnable{ 48. **private** **final** BlockingQueue sharedQueue; 50. **public** Consumer (BlockingQueue sharedQueue) { 51. **this**.sharedQueue = sharedQueue; 52. } 54. @Override 55. **public** **void** run() { 56. **while**(**true**){ 57. **try** { 58. System.out.println("Consumed: "+ sharedQueue.take()); 59. } **catch** (InterruptedException ex) { 60. Logger.getLogger(Consumer.**class**.getName()).log(Level.SEVERE, **null**, ex); 61. } 62. } 63. } 64. }   **Output**  Produced: 0  Produced: 1  Produced: 2  Produced: 3  Produced: 4  Produced: 5  Produced: 6  Produced: 7  Produced: 8  Produced: 9  Consumed: 0  Consumed: 1  Consumed: 2  Consumed: 3  Consumed: 4  Consumed: 5  Consumed: 6  Consumed: 7  Consumed: 8  Consumed: 9  38) What is the difference between Java Callable interface and Runnable interface?  The Callable interface and Runnable interface both are used by the classes which wanted to execute with multiple threads. However, there are two main differences between the both :   * A Callable <V> interface can return a result, whereas the Runnable interface cannot return any result. * A Callable <V> interface can throw a checked exception, whereas the Runnable interface cannot throw checked exception. * A Callable <V> interface cannot be used before the Java 5 whereas the Runnable interface can be used.   39) What is the Atomic action in Concurrency in Java?   * The Atomic action is the operation which can be performed in a single unit of a task without any interference of the other operations. * The Atomic action cannot be stopped in between the task. Once started it fill stop after the completion of the task only. * An increment operation such as a++ does not allow an atomic action. * All reads and writes operation for the primitive variable (except long and double) are the atomic operation. * All reads and writes operation for the volatile variable (including long and double) are the atomic operation. * The Atomic methods are available in java.util.Concurrent package.   40) What is lock interface in Concurrency API in Java?  The java.util.concurrent.locks.Lock interface is used as the synchronization mechanism. It works similar to the synchronized block. There are a few differences between the lock and synchronized block that are given below.   * Lock interface provides the guarantee of sequence in which the waiting thread will be given the access, whereas the synchronized block doesn't guarantee it. * Lock interface provides the option of timeout if the lock is not granted whereas the synchronized block doesn't provide that. * The methods of Lock interface, i.e., Lock() and Unlock() can be called in different methods whereas single synchronized block must be fully contained in a single method.   41) Explain the ExecutorService Interface.  The ExecutorService Interface is the subinterface of Executor interface and adds the features to manage the lifecycle. Consider the following example.    2. **import** java.util.concurrent.ExecutorService; 3. **import** java.util.concurrent.Executors; 4. **import** java.util.concurrent.TimeUnit; 6. **public** **class** TestThread { 7. **public** **static** **void** main(**final** String[] arguments) **throws** InterruptedException { 8. ExecutorService e = Executors.newSingleThreadExecutor(); 10. **try** { 11. e.submit(**new** Thread()); 12. System.out.println("Shutdown executor"); 13. e.shutdown(); 14. e.awaitTermination(5, TimeUnit.SECONDS); 15. } **catch** (InterruptedException ex) { 16. System.err.println("tasks interrupted"); 17. } **finally** { 19. **if** (!e.isTerminated()) { 20. System.err.println("cancel non-finished tasks"); 21. } 22. e.shutdownNow(); 23. System.out.println("shutdown finished"); 24. } 25. } 27. **static** **class** Task **implements** Runnable { 29. **public** **void** run() { 31. **try** { 32. Long duration = (**long**) (Math.random() \* 20); 33. System.out.println("Running Task!"); 34. TimeUnit.SECONDS.sleep(duration); 35. } **catch** (InterruptedException ex) { 36. ex.printStackTrace(); 37. } 38. } 39. } 40. }   **Output**  Shutdown executor  shutdown finished  42) What is the difference between Synchronous programming and Asynchronous programming regarding a thread?  **Synchronous programming:**In Synchronous programming model, a thread is assigned to complete a task and hence thread started working on it, and it is only available for other tasks once it will end the assigned task.  **Asynchronous Programming:**In Asynchronous programming, one job can be completed by multiple threads and hence it provides maximum usability of the various threads.  43) What do you understand by Callable and Future in Java?  **Java Callable interface:**In Java5 callable interface was provided by the package java.util.concurrent. It is similar to the Runnable interface but it can return a result, and it can throw an Exception. It also provides a run() method for execution of a thread. Java Callable can return any object as it uses Generic.  **Syntax:**  public interface Callable<V>  **Java Future interface:** Java Future interface gives the result of a concurrent process. The Callable interface returns the object of java.util.concurrent.Future.  Java Future provides following methods for implementation.   * **cancel(boolean mayInterruptIfRunning):** It is used to cancel the execution of the assigned task. * **get():** It waits for the time if execution not completed and then retrieved the result. * **isCancelled():** It returns the Boolean value as it returns true if the task was canceled before the completion. * **isDone():** It returns true if the job is completed successfully else returns false.   44. What is the difference between ScheduledExecutorService and ExecutorService interface?  ExecutorServcie and ScheduledExecutorService both are the interfaces of java.util.Concurrent package but scheduledExecutorService provides some additional methods to execute the Runnable and Callable tasks with the delay or every fixed time period.  45) Define FutureTask class in Java?  Java FutureTask class provides a base implementation of the Future interface. The result can only be obtained if the execution of one task is completed, and if the computation is not achieved then get method will be blocked. If the execution is completed, then it cannot be re-started and can't be canceled.  **Syntax**  public class FutureTask<V> extends Object implements RunnableFuture<V>  Next Topic[Java Collections interview Questions](https://www.javatpoint.com/java-collections-interview-questions)  [← Prev](https://www.javatpoint.com/corejava-interview-questions-5)[Next →](https://www.javatpoint.com/java-collections-interview-questions)  [**1**](https://www.javatpoint.com/corejava-interview-questions) [**2**](https://www.javatpoint.com/corejava-interview-questions-2) [**3**](https://www.javatpoint.com/corejava-interview-questions-3) [**4**](https://www.javatpoint.com/corejava-interview-questions-4) [**5**](https://www.javatpoint.com/corejava-interview-questions-5) [**6**](https://www.javatpoint.com/java-multithreading-interview-questions) [**7**](https://www.javatpoint.com/java-collections-interview-questions) [**8**](https://www.javatpoint.com/jdbc-interview-questions)   |  |  | | --- | --- | | [**Java Basics Interview Questions**](https://www.javatpoint.com/corejava-interview-questions#corebasicsinterview) | [**Java OOPs Interview Questions**](https://www.javatpoint.com/corejava-interview-questions#oopsinterview) | | [**Java Multithreading Interview Questions**](https://www.javatpoint.com/java-multithreading-interview-questions) | [**Java String & Exception Interview Questions**](https://www.javatpoint.com/corejava-interview-questions-3) | | [**Java Collection Interview Questions**](https://www.javatpoint.com/java-collections-interview-questions) | [**JDBC Interview Questions**](https://www.javatpoint.com/jdbc-interview-questions) | | [**Servlet Interview Questions**](https://www.javatpoint.com/servletinterview) | [**JSP Interview Questions**](https://www.javatpoint.com/jspinterview) | | [**Spring Interview Questions**](https://www.javatpoint.com/spring-interview-questions) | [**Hibernate Interview Questions**](https://www.javatpoint.com/hibernate-interview-questions) | | [**PL/SQL Interview Questions**](https://www.javatpoint.com/pl-sql-interview-questions) | [**SQL Interview Questions**](https://www.javatpoint.com/sql-interview-questions) | | [**Oracle Interview Questions**](https://www.javatpoint.com/oracle-interview-questions) | [**Android Interview Questions**](https://www.javatpoint.com/android-interview-questions) | | [**SQL Server Interview Questions**](https://www.javatpoint.com/sql-server-interview-questions) | [**MySQL Interview Questions**](https://www.javatpoint.com/mysql-interview-questions) |   [ADVERTISEMENT BY ADRECOVER](https://adrecover.com/?utm_campaign=ByAdRecover&utm_source=www.javatpoint.com&utm_medium=ByAdRecover-336) |

**Javatpoint Services**

JavaTpoint offers too many high quality services. Mail us on hr@javatpoint.com, to get more information about given services.

* Website Designing
* Website Development
* Java Development
* PHP Development
* WordPress
* Graphic Designing
* Logo
* Digital Marketing
* On Page and Off Page SEO
* PPC
* Content Development
* Corporate Training
* Classroom and Online Training
* Data Entry

**Training For College Campus**

JavaTpoint offers college campus training on Core Java, Advance Java, .Net, Android, Hadoop, PHP, Web Technology and Python. Please mail your requirement at hr@javatpoint.com.  
Duration: 1 week to 2 week

Like/Subscribe us for latest updates or newsletter[RSS Feed](https://feeds.feedburner.com/javatpointsonoo) [Subscribe to Get Email Alerts](https://feedburner.google.com/fb/a/mailverify?uri=javatpointsonoo) [Facebook Page](https://www.facebook.com/javatpoint) [Twitter Page](https://twitter.com/pagejavatpoint) [YouTube](https://www.youtube.com/channel/UCUnYvQVCrJoFWZhKK3O2xLg) [Blog Page](https://javatpoint.blogspot.com/)

**LEARN TUTORIALS**

[Learn Java](https://www.javatpoint.com/java-tutorial)[Learn Data Structures](https://www.javatpoint.com/data-structure-tutorial)[Learn C Programming](https://www.javatpoint.com/c-programming-language-tutorial)[Learn C++ Tutorial](https://www.javatpoint.com/cpp-tutorial)[Learn C# Tutorial](https://www.javatpoint.com/c-sharp-tutorial)[Learn PHP Tutorial](https://www.javatpoint.com/php-tutorial)[Learn HTML Tutorial](https://www.javatpoint.com/html-tutorial)[Learn JavaScript Tutorial](https://www.javatpoint.com/javascript-tutorial)[Learn jQuery Tutorial](https://www.javatpoint.com/jquery-tutorial)[Learn Spring Tutorial](https://www.javatpoint.com/spring-tutorial)

**OUR WEBSITES**

[Javatpoint.com](https://www.javatpoint.com/)[Hindi100.com](https://www.hindi100.com/)[Lyricsia.com](https://www.lyricsia.com/)[Quoteperson.com](https://www.quoteperson.com/)[Jobandplacement.com](https://www.jobandplacement.com/)

**OUR SERVICES**

Website Development

Android Development

Website Designing

Digital Marketing

Summer Training

Industrial Training

College Campus Training

**CONTACT**

Address: G-13, 2nd Floor, Sec-3

Noida, UP, 201301, India

Contact No: 0120-4256464, 9990449935

[Contact Us](https://www.javatpoint.com/contact-us)[Subscribe Us](https://www.javatpoint.com/subscribe.jsp)[Privacy Policy](https://www.javatpoint.com/privacy-policy)[Sitemap](https://www.javatpoint.com/sitemap.xml)  
[About Me](https://www.javatpoint.com/sonoo-jaiswal)

© Copyright 2011-2018 www.javatpoint.com. All rights reserved. Developed by JavaTpoint.

⇧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [Javatpoint Logo](https://www.javatpoint.com/)  Top of Form   |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | |  |   Bottom of Form |

* [Home](https://www.javatpoint.com/)
* [Interview Questions](https://www.javatpoint.com/interview-questions-and-answers)
* [Java](https://www.javatpoint.com/java-tutorial)
* [C](https://www.javatpoint.com/c-programming-language-tutorial)
* [SQL](https://www.javatpoint.com/sql-tutorial)
* [Android](https://www.javatpoint.com/android-tutorial)
* [HTML](https://www.javatpoint.com/html-tutorial)
* [XML](https://www.javatpoint.com/xml-tutorial)
* [JavaScript](https://www.javatpoint.com/javascript-tutorial)
* [Ajax](https://www.javatpoint.com/ajax-tutorial)
* [Cloud](https://www.javatpoint.com/cloud-computing-tutorial)
* [Python](https://www.javatpoint.com/python-tutorial)
* [Web Services](https://www.javatpoint.com/web-services-tutorial)
* [Quiz](https://www.javatpoint.com/examaccess)
* [Projects](https://www.javatpoint.com/free-java-projects)
* [Comment](https://www.javatpoint.com/comment.jsp)
* [Forum](https://www.javatpoint.com/forum)
* [Training](https://www.javatpoint.com/training)

**All Interview**

[Interview Tips](https://www.javatpoint.com/interview-tips)[JOB/HR Interview](https://www.javatpoint.com/job-interview-questions)[Competency Interview](https://www.javatpoint.com/competency-based-interview-questions)[Business Analyst](https://www.javatpoint.com/business-analyst-interview-questions)[Behavioral Interview](https://www.javatpoint.com/behavioral-interview-questions)

**Company Interview**

[Company Interview](https://www.javatpoint.com/company-interview-questions-and-recruitment-process)[IBM Interview](https://www.javatpoint.com/ibm-interview-questions)[Infosys Interview](https://www.javatpoint.com/infosys-interview-questions)[Capgemini Interview](https://www.javatpoint.com/capgemini-interview-questions)[Cognizant Interview](https://www.javatpoint.com/cognizant-interview-questions)[Wipro Interview](https://www.javatpoint.com/wipro-interview-questions)[Accenture Interview](https://www.javatpoint.com/accenture-interview-questions)[iGate Interview](https://www.javatpoint.com/igate-interview-questions)[TCS Interview](https://www.javatpoint.com/tcs-interview-questions)[HCL Interview](https://www.javatpoint.com/hcl-interview-questions)[Adobe Interview](https://www.javatpoint.com/adobe-interview-questions)[Microsoft Interview](https://www.javatpoint.com/microsoft-interview-questions)[DXC Technology Interview](https://www.javatpoint.com/dxc-technology-interview-questions)[Ericsson Interview](https://www.javatpoint.com/ericsson-interview-questions)[EXL Service Interview](https://www.javatpoint.com/exl-interview-questions)[IndiaMART Interview](https://www.javatpoint.com/indiamart-interview-questions)[Intuit Interview](https://www.javatpoint.com/intuit-interview-questions)[SpaceX Interview](https://www.javatpoint.com/spacex-interview-questions)[Sapient Interview](https://www.javatpoint.com/sapient-interview-questions)[Amazon Interview](https://www.javatpoint.com/amazon-interview-questions)

**Technical Interview**

[C Interview](https://www.javatpoint.com/c-interview-questions)[C++ Interview](https://www.javatpoint.com/cpp-interview-questions)[Data Structure Interview](https://www.javatpoint.com/data-structure-interview-questions)[Linux Interview](https://www.javatpoint.com/linux-interview-questions)[Unix Interview](https://www.javatpoint.com/unix-interview-questions)[Shell Scripting](https://www.javatpoint.com/shell-scripting-interview-questions)[Networking Interview](https://www.javatpoint.com/networking-interview-questions)[CCNA Interview](https://www.javatpoint.com/ccna-interview-questions)[Android Interview](https://www.javatpoint.com/android-interview-questions)[Cloud Computing](https://www.javatpoint.com/cloud-computing-interview-questions)[Hadoop Interview](https://www.javatpoint.com/hadoop-interview-questions)[Testing/QTP Interview](https://www.javatpoint.com/software-testing-interview-questions)[Selenium Interview](https://www.javatpoint.com/selenium-interview-questions)[Web Services Interview](https://www.javatpoint.com/web-services-interview-questions)[OS Interview](https://www.javatpoint.com/operating-system-interview-questions)[Excel Interview](https://www.javatpoint.com/excel-interview-questions)[SEO Interview](https://www.javatpoint.com/seo-interview-questions)[Digital Marketing](https://www.javatpoint.com/digital-marketing-interview-questions-and-answers)[Python Interview](https://www.javatpoint.com/python-interview-questions)[Django Interview](https://www.javatpoint.com/django-interview-questions-and-answers)[Pascal Interview](https://www.javatpoint.com/pascal-interview-questions-and-answers)[Ruby Interview](https://www.javatpoint.com/ruby-interview-questions)[Ruby on Rails Interview](https://www.javatpoint.com/ruby-on-rails-interview-questions)[Memcached Interview](https://www.javatpoint.com/memcached-interview-questions-and-answers)[Go Interview](https://www.javatpoint.com/go-programming-interview-questions)[OpenStack Interview](https://www.javatpoint.com/openstack-interview-questions-and-answers)[Scala Interview](https://www.javatpoint.com/scala-interview-questions)[Control Systems](https://www.javatpoint.com/control-systems-interview-questions)[Electrical Machines](https://www.javatpoint.com/electrical-machines-interview-questions)[Power System](https://www.javatpoint.com/power-system-interview-questions)[Digital Electronics](https://www.javatpoint.com/digital-electronics-interview-questions)[Robotics Interview](https://www.javatpoint.com/robotics-interview-questions)[TypeScript Interview](https://www.javatpoint.com/typescript-interview-questions)[Swift Interview](https://www.javatpoint.com/swift-interview-questions)[Blockchain Interview](https://www.javatpoint.com/blockchain-interview-questions)[Bitcoin Interview](https://www.javatpoint.com/bitcoin-interview-questions)[AWS Interview](https://www.javatpoint.com/aws-interview-questions)[Informatica Interview](https://www.javatpoint.com/informatica-interview-questions)[QA Interview](https://www.javatpoint.com/qa-interview-questions)[React Interview](https://www.javatpoint.com/react-interview-questions)[GIT Interview](https://www.javatpoint.com/git-interview-questions)[J2EE Interview](https://www.javatpoint.com/j2ee-interview-questions)[Deep Learning](https://www.javatpoint.com/deep-learning-interview-questions)[Salesforce Interview](https://www.javatpoint.com/salesforce-interview-questions)[TestNG Interview](https://www.javatpoint.com/testng-interview-questions)[Agile Interview](https://www.javatpoint.com/agile-interview-questions)[Machine Learning](https://www.javatpoint.com/machine-learning-interview-questions)[API Testing](https://www.javatpoint.com/api-testing-interview-questions)[Algorithm](https://www.javatpoint.com/algorithm-interview-questions)[Java Design Pattern](https://www.javatpoint.com/java-design-pattern-interview-questions)[Teradata](https://www.javatpoint.com/teradata-interview-questions)[TensorFlow](https://www.javatpoint.com/tensorflow-interview-questions)[LINQ](https://www.javatpoint.com/linq-interview-questions)[Mainframe](https://www.javatpoint.com/mainframe-interview-questions)[Laravel](https://www.javatpoint.com/laravel-interview-questions)[JIRA](https://www.javatpoint.com/jira-interview-questions)[JCL](https://www.javatpoint.com/jcl-interview-questions)[Pytorch](https://www.javatpoint.com/pytorch-interview-questions)[ETL Testing](https://www.javatpoint.com/etl-testing-interview-questions)[Linked List](https://www.javatpoint.com/linked-list-interview-questions)[DataStage](https://www.javatpoint.com/datastage-interview-questions)[Tableau](https://www.javatpoint.com/tableau-interview-questions)[SAP ABAP](https://www.javatpoint.com/sap-abap-interview-questions)[DAA](https://www.javatpoint.com/daa-interview-questions)[Computer Graphics](https://www.javatpoint.com/computer-graphics-interview-questions)[Data Science](https://www.javatpoint.com/data-science-interview-questions)[Software Engineering](https://www.javatpoint.com/software-engineering-interview-questions)[Power BI](https://www.javatpoint.com/power-bi-interview-questions)[Xamarin](https://www.javatpoint.com/xamarin-interview-questions)[Ansible](https://www.javatpoint.com/ansible-interview-questions)[Data Warehouse](https://www.javatpoint.com/data-warehouse-interview-questions)[Ionic](https://www.javatpoint.com/ionic-interview-questions)[Devops](https://www.javatpoint.com/devops-interview-questions)[React Native](https://www.javatpoint.com/react-native-interview-questions)[Python Pandas](https://www.javatpoint.com/python-pandas-interview-questions)[HTTP Interview](https://www.javatpoint.com/http-interview-questions)[Kali Linux Interview](https://www.javatpoint.com/kali-linux-interview-questions)[PowerShell Interview](https://www.javatpoint.com/powershell-interview-questions)[SharePoint Interview](https://www.javatpoint.com/sharepoint-interview-questions)[Talend Interview](https://www.javatpoint.com/talend-interview-questions)[Microsoft Azure Interview](https://www.javatpoint.com/microsoft-azure-interview-questions)[R Interview](https://www.javatpoint.com/r-interview-questions)[Flutter Interview](https://www.javatpoint.com/flutter-interview-questions)[MATLAB Interview](https://www.javatpoint.com/matlab-interview-questions)[ES6 Interview](https://www.javatpoint.com/es6-interview-questions)[RPA Interview](https://www.javatpoint.com/rpa-interview-questions)[Desktop Support Interview](https://www.javatpoint.com/desktop-support-interview-questions)[Angular 8 Interview](https://www.javatpoint.com/angular-8-interview-questions)[GraphQL Interview](https://www.javatpoint.com/graphql-interview-questions)[Vue.js Interview](https://www.javatpoint.com/vue-js-interview-questions)[Automation Anywhere Interview](https://www.javatpoint.com/automation-anywhere-interview-questions)[AI Interview](https://www.javatpoint.com/artificial-intelligence-interview-questions)[UiPath Interview](https://www.javatpoint.com/uipath-interview-questions)[Blue Prism Interview](https://www.javatpoint.com/blue-prism-interview-questions)[SSIS Interview](https://www.javatpoint.com/ssis-interview-questions)[Mobile Computing Interview](https://www.javatpoint.com/mobile-computing-interview-questions)[ITIL Interview](https://www.javatpoint.com/itil-interview-questions)[SAS Interview](https://www.javatpoint.com/sas-interview-questions)[Elasticsearch Interview](https://www.javatpoint.com/elasticsearch-interview-questions)[Entity Framework Interview](https://www.javatpoint.com/entity-framework-interview-questions)[RxJS Interview](https://www.javatpoint.com/rxjs-interview-questions)[Accounting Interview](https://www.javatpoint.com/accounting-interview-questions)[Electron.js Interview](https://www.javatpoint.com/electronjs-interview-questions)[Knockout JS Interview](https://www.javatpoint.com/knockout-js-interview-questions)[Top Angular Interview](https://www.javatpoint.com/top-angular-interview-questions)[IAS Interview Question](https://www.javatpoint.com/ias-interview-questions)[Array Interview Questions in Java](https://www.javatpoint.com/array-interview-questions-in-java)[Banking Interview](https://www.javatpoint.com/banking-interview-questions)[Interview Questions for Freshers](https://www.javatpoint.com/interview-questions-for-freshers)[PowerPoint Interview](https://www.javatpoint.com/powerpoint-interview-questions)[BPO Interview](https://www.javatpoint.com/bpo-interview-questions)[Civil Engineering Interview](https://www.javatpoint.com/civil-engineering-interview-questions)[Apache Spark Interview](https://www.javatpoint.com/apache-spark-interview-questions)[Data Mining Interview](https://www.javatpoint.com/data-mining-interview-questions)[Java Support Interview](https://www.javatpoint.com/java-support-interview-questions)[Kubernetes Interview](https://www.javatpoint.com/kubernetes-interview-questions)[Web API Interview](https://www.javatpoint.com/web-api-interview-questions)[Scrum Master Interview](https://www.javatpoint.com/scrum-master-interview-questions)[System Design Interview](https://www.javatpoint.com/system-design-interview-questions)[SSB Interview Questions](https://www.javatpoint.com/ssb-interview-questions)[Teachers Interview Questions](https://www.javatpoint.com/teachers-interview-questions)[Finance Interview Questions](https://www.javatpoint.com/finance-interview-questions)[Technical Support Interview](https://www.javatpoint.com/technical-support-interview-questions)[Active Directory Interview](https://www.javatpoint.com/active-directory-interview-questions)[Appium Interview](https://www.javatpoint.com/appium-interview-questions)[Cucumber Interview](https://www.javatpoint.com/cucumber-interview-questions)[Full-Stack Developer](https://www.javatpoint.com/full-stack-developer-interview-questions-java)

**Web Interview**

[HTML Interview](https://www.javatpoint.com/html-interview-questions)[CSS Interview](https://www.javatpoint.com/css-interview-questions)[JavaScript Interview](https://www.javatpoint.com/javascript-interview-questions)[jQuery Interview](https://www.javatpoint.com/jquery-interview-questions)[AngularJS Interview](https://www.javatpoint.com/angularjs-interview-questions)[Angular Interview](https://www.javatpoint.com/angular-interview-questions)[Angular 7 Interview](https://www.javatpoint.com/angular-7-interview-questions)[Node.js Interview](https://www.javatpoint.com/node-js-interview-questions)[AJAX Interview](https://www.javatpoint.com/ajax-interview-questions)[XML Interview](https://www.javatpoint.com/xml-interview-questions)[Dojo Interview](https://www.javatpoint.com/dojo-interview-questions-and-answers)[Backbone.js Interview](https://www.javatpoint.com/backbone-js-interview-questions-and-answers)[Ember.js Interview](https://www.javatpoint.com/ember-js-interview-questions)[XHTML Interview](https://www.javatpoint.com/xhtml-interview-questions)[XSLT Interview](https://www.javatpoint.com/xslt-interview-questions)[XPath Interview](https://www.javatpoint.com/xpath-interview-questions)[XQuery Interview](https://www.javatpoint.com/xquery-interview-questions)[XForms Interview](https://www.javatpoint.com/xforms-interview-questions)[SASS Interview](https://www.javatpoint.com/sass-interview-questions-and-answers)[CoffeeScript Interview](https://www.javatpoint.com/coffeescript-interview-questions-and-answers)[LESS Interview](https://www.javatpoint.com/less-interview-questions-and-answers)[Perl Interview](https://www.javatpoint.com/perl-interview-questions)[Pure.CSS Interview](https://www.javatpoint.com/pure-css-interview-questions)[Materialize Interview](https://www.javatpoint.com/materialize-css-interview-questions)[Framework7 Interview](https://www.javatpoint.com/framework7-interview-questions)[SVG Interview](https://www.javatpoint.com/svg-interview-questions)

**PHP Interview**

[PHP Interview](https://www.javatpoint.com/php-interview-questions)[Wordpress Interview](https://www.javatpoint.com/wordpress-interview-questions)[Joomla Interview](https://www.javatpoint.com/joomla-interview-questions-and-answers)[Drupal Interview](https://www.javatpoint.com/drupal-interview-questions-and-answers)[Magento Interview](https://www.javatpoint.com/magento-interview-questions-and-answers)[CodeIgniter Interview](https://www.javatpoint.com/codeigniter-interview-questions-and-answers)[Phalcon Interview](https://www.javatpoint.com/phalcon-interview-questions)

**.Net Interview**

[.Net Interview](https://www.javatpoint.com/dot-net-interview-questions)[C# Interview](https://www.javatpoint.com/c-sharp-interview-questions)[ASP.NET Interview](https://www.javatpoint.com/asp-dot-net-interview-questions)[ADO.NET Interview](https://www.javatpoint.com/ado-dot-net-interview-questions)[WCF Interview](https://www.javatpoint.com/wcf-interview-questions)[WPF Interview](https://www.javatpoint.com/wpf-interview-questions)[Silverlight Interview](https://www.javatpoint.com/silverlight-interview-questions-and-answers)[F# Interview](https://www.javatpoint.com/fsharp-interview-questions)

**Java Interview**

[Java 1 Interview](https://www.javatpoint.com/corejava-interview-questions)[Java 2 Interview](https://www.javatpoint.com/corejava-interview-questions-2)[Exception & String](https://www.javatpoint.com/corejava-interview-questions-3)[Multithreading](https://www.javatpoint.com/java-multithreading-interview-questions)[**Collections Interview** [open link](https://www.javatpoint.com/java-collections-interview-questions)](https://www.javatpoint.com/java-collections-interview-questions)[JDBC Interview](https://www.javatpoint.com/jdbc-interview-questions)[Servlet Interview](https://www.javatpoint.com/servletinterview)[JSP Interview](https://www.javatpoint.com/jspinterview)[EJB Interview](https://www.javatpoint.com/ejb-interview-questions)[Struts Interview](https://www.javatpoint.com/struts-interview-questions)[Hibernate Interview](https://www.javatpoint.com/hibernate-interview-questions)[Spring Interview](https://www.javatpoint.com/spring-interview-questions)[Spring Boot Interview](https://www.javatpoint.com/spring-boot-interview-questions)[Spring MVC Interview](https://www.javatpoint.com/spring-mvc-interview-questions)[JPA Interview](https://www.javatpoint.com/jpa-interview-questions)[Maven Interview](https://www.javatpoint.com/maven-interview-questions)[JUnit Interview](https://www.javatpoint.com/junit-interview-questions)[JSF Interview](https://www.javatpoint.com/jsf-interview-questions)[PrimeFaces Interview](https://www.javatpoint.com/primefaces-interview-questions)[RichFaces Interview](https://www.javatpoint.com/richfaces-interview-questions)[GWT Interview](https://www.javatpoint.com/gwt-interview-questions)[JDB Interview](https://www.javatpoint.com/jdb-interview-questions)[JOGL Interview](https://www.javatpoint.com/jogl-interview-questions)[jBPM Interview](https://www.javatpoint.com/jbpm-interview-questions)[Jenkins Interview](https://www.javatpoint.com/jenkins-interview-questions)[Java Constructor Interview Questions](https://www.javatpoint.com/java-constructor-interview-questions)[Java Technical Architect Interview Questions](https://www.javatpoint.com/java-technical-architect-interview-questions)

**Database Interview**

[DBMS Interview](https://www.javatpoint.com/dbms-interview-questions)[SQL Interview](https://www.javatpoint.com/sql-interview-questions)[PL/SQL Interview](https://www.javatpoint.com/pl-sql-interview-questions)[Oracle Interview](https://www.javatpoint.com/oracle-interview-questions)[MySql Interview](https://www.javatpoint.com/mysql-interview-questions)[SQL Server Interview](https://www.javatpoint.com/sql-server-interview-questions)[MongoDB Interview](https://www.javatpoint.com/mongodb-interview-questions)[Cassandra Interview](https://www.javatpoint.com/cassandra-interview-questions-and-answers)[DB2 Interview](https://www.javatpoint.com/db2-interview-questions-and-answers)[Access Interview](https://www.javatpoint.com/microsoft-access-interview-questions-and-answers)[SQLite Interview](https://www.javatpoint.com/sqlite-interview-questions)[CouchDB Interview](https://www.javatpoint.com/couchdb-interview-questions)[Neo4j Interview](https://www.javatpoint.com/neo4j-interview-questions-and-answers)[MariaDB Interview](https://www.javatpoint.com/mariadb-interview-questions)[PouchDB Interview](https://www.javatpoint.com/pouchdb-interview-questions)[Redis Interview](https://www.javatpoint.com/redis-interview-questions-and-answers)

[ADS BY ADRECOVER](https://adrecover.com/?utm_campaign=ByAdRecover&utm_source=www.javatpoint.com&utm_medium=ByAdRecover-160)

[ADVERTISEMENT BY ADRECOVER](https://adrecover.com/?utm_campaign=ByAdRecover&utm_source=www.javatpoint.com&utm_medium=ByAdRecover-728)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 34 Java Collections Interview Questions  In Java, collection interview questions are most asked by the interviewers. Here is the list of the most asked collections interview questions with answers.  1) What is the Collection framework in Java?  Collection Framework is a combination of classes and interface, which is used to store and manipulate the data in the form of objects. It provides various classes such as ArrayList, Vector, Stack, and HashSet, etc. and interfaces such as List, Queue, Set, etc. for this purpose.  2) What are the main differences between array and collection?  Array and Collection are somewhat similar regarding storing the references of objects and manipulating the data, but they differ in many ways. The main differences between the array and Collection are defined below:   * Arrays are always of fixed size, i.e., a user can not increase or decrease the length of the array according to their requirement or at runtime, but In Collection, size can be changed dynamically as per need. * Arrays can only store homogeneous or similar type objects, but in Collection, heterogeneous objects can be stored. * Arrays cannot provide the ?ready-made? methods for user requirements as sorting, searching, etc. but Collection includes readymade methods to use.   3) Explain various interfaces used in Collection framework?  Collection framework implements various interfaces, Collection interface and Map interface (java.util.Map) are the mainly used interfaces of Java Collection Framework. List of interfaces of Collection Framework is given below:  **1. Collection interface:** Collection (java.util.Collection) is the primary interface, and every collection must implement this interface.  **Syntax:**   1. **public** **interface** Collection<E>**extends** Iterable   Where <E> represents that this interface is of Generic type  **2. List interface:**List interface extends the Collection interface, and it is an ordered collection of objects. It contains duplicate elements. It also allows random access of elements.  **Syntax:**   1. **public** **interface** List<E> **extends** Collection<E>   **3. Set interface:** Set (java.util.Set) interface is a collection which cannot contain duplicate elements. It can only include inherited methods of Collection interface  **Syntax:**   1. **public** **interface** Set<E> **extends** Collection<E>   **Queue interface:**Queue (java.util.Queue) interface defines queue data structure, which stores the elements in the form FIFO (first in first out).  **Syntax:**   1. **public** **interface** Queue<E> **extends** Collection<E>   **4. Dequeue interface:** it is a double-ended-queue. It allows the insertion and removal of elements from both ends. It implants the properties of both Stack and queue so it can perform LIFO (Last in first out) stack and FIFO (first in first out) queue, operations.  **Syntax:**   1. **public** **interface** Dequeue<E> **extends** Queue<E>   **5. Map interface:**A Map (java.util.Map) represents a key, value pair storage of elements. Map interface does not implement the Collection interface. It can only contain a unique key but can have duplicate elements. There are two interfaces which implement Map in java that are Map interface and Sorted Map.  4) What is the difference between ArrayList and Vector?   |  |  |  | | --- | --- | --- | | **No.** | **ArrayList** | **Vector** | | 1) | ArrayList is not synchronized. | Vector is synchronized. | | 2) | ArrayList is not a legacy class. | Vector is a legacy class. | | 3) | ArrayList increases its size by 50% of the array size. | Vector increases its size by doubling the array size. | | 4) | ArrayList is not ?thread-safe? as it is not synchronized. | Vector list is ?thread-safe? as it?s every method is synchronized. |   5) What is the difference between ArrayList and LinkedList?   |  |  |  | | --- | --- | --- | | **No.** | **ArrayList** | **LinkedList** | | 1) | ArrayList uses a dynamic array. | LinkedList uses a doubly linked list. | | 2) | ArrayList is not efficient for manipulation because too much is required. | LinkedList is efficient for manipulation. | | 3) | ArrayList is better to store and fetch data. | LinkedList is better to manipulate data. | | 4) | ArrayList provides random access. | LinkedList does not provide random access. | | 5) | ArrayList takes less memory overhead as it stores only object | LinkedList takes more memory overhead, as it stores the object as well as the address of that object. |   6) What is the difference between Iterator and ListIterator?  Iterator traverses the elements in the forward direction only whereas ListIterator traverses the elements into forward and backward direction.   |  |  |  | | --- | --- | --- | | **No.** | **Iterator** | **ListIterator** | | 1) | The Iterator traverses the elements in the forward direction only. | ListIterator traverses the elements in backward and forward directions both. | | 2) | The Iterator can be used in List, Set, and Queue. | ListIterator can be used in List only. | | 3) | The Iterator can only perform remove operation while traversing the collection. | ListIterator can perform ?add,? ?remove,? and ?set? operation while traversing the collection. |   7) What is the difference between Iterator and Enumeration?   |  |  |  | | --- | --- | --- | | **No.** | **Iterator** | **Enumeration** | | 1) | The Iterator can traverse legacy and non-legacy elements. | Enumeration can traverse only legacy elements. | | 2) | The Iterator is fail-fast. | Enumeration is not fail-fast. | | 3) | The Iterator is slower than Enumeration. | Enumeration is faster than Iterator. | | 4) | The Iterator can perform remove operation while traversing the collection. | The Enumeration can perform only traverse operation on the collection. |   8) What is the difference between List and Set?  The List and Set both extend the collection interface. However, there are some differences between the both which are listed below.   * The List can contain duplicate elements whereas Set includes unique items. * The List is an ordered collection which maintains the insertion order whereas Set is an unordered collection which does not preserve the insertion order. * The List interface contains a single legacy class which is Vector class whereas Set interface does not have any legacy class. * The List interface can allow n number of null values whereas Set interface only allows a single null value.   9) What is the difference between HashSet and TreeSet?  The HashSet and TreeSet, both classes, implement Set interface. The differences between the both are listed below.   * HashSet maintains no order whereas TreeSet maintains ascending order. * HashSet impended by hash table whereas TreeSet implemented by a Tree structure. * HashSet performs faster than TreeSet. * HashSet is backed by HashMap whereas TreeSet is backed by TreeMap.   10) What is the difference between Set and Map?  The differences between the Set and Map are given below.   * Set contains values only whereas Map contains key and values both. * Set contains unique values whereas Map can contain unique Keys with duplicate values. * Set holds a single number of null value whereas Map can include a single null key with n number of null values.   11) What is the difference between HashSet and HashMap?  The differences between the HashSet and HashMap are listed below.   * HashSet contains only values whereas HashMap includes the entry (key, value). HashSet can be iterated, but HashMap needs to convert into Set to be iterated. * HashSet implements Set interface whereas HashMap implements the Map interface * HashSet cannot have any duplicate value whereas HashMap can contain duplicate values with unique keys. * HashSet contains the only single number of null value whereas HashMap can hold a single null key with n number of null values.   12) What is the difference between HashMap and TreeMap?  The differences between the HashMap and TreeMap are given below.   * HashMap maintains no order, but TreeMap maintains ascending order. * HashMap is implemented by hash table whereas TreeMap is implemented by a Tree structure. * HashMap can be sorted by Key or value whereas TreeMap can be sorted by Key. * HashMap may contain a null key with multiple null values whereas TreeMap cannot hold a null key but can have multiple null values.   13) What is the difference between HashMap and Hashtable?   |  |  |  | | --- | --- | --- | | **No.** | **HashMap** | **Hashtable** | | 1) | HashMap is not synchronized. | Hashtable is synchronized. | | 2) | HashMap can contain one null key and multiple null values. | Hashtable cannot contain any null key or null value. | | 3) | HashMap is not ?thread-safe,? so it is useful for non-threaded applications. | Hashtable is thread-safe, and it can be shared between various threads. | | 4) | 4) HashMap inherits the AbstractMap class | Hashtable inherits the Dictionary class. |   14) What is the difference between Collection and Collections?  The differences between the Collection and Collections are given below.   * The Collection is an interface whereas Collections is a class. * The Collection interface provides the standard functionality of data structure to List, Set, and Queue. However, Collections class is to sort and synchronize the collection elements. * The Collection interface provides the methods that can be used for data structure whereas Collections class provides the static methods which can be used for various operation on a collection.   15) What is the difference between Comparable and Comparator?   |  |  |  | | --- | --- | --- | | **No.** | **Comparable** | **Comparator** | | 1) | Comparable provides only one sort of sequence. | The Comparator provides multiple sorts of sequences. | | 2) | It provides one method named compareTo(). | It provides one method named compare(). | | 3) | It is found in java.lang package. | It is located in java.util package. | | 4) | If we implement the Comparable interface, The actual class is modified. | The actual class is not changed. |   16) What do you understand by BlockingQueue?  BlockingQueue is an interface which extends the Queue interface. It provides concurrency in the operations like retrieval, insertion, deletion. While retrieval of any element, it waits for the queue to be non-empty. While storing the elements, it waits for the available space. BlockingQueue cannot contain null elements, and implementation of BlockingQueue is thread-safe.  **Syntax:**   1. **public** **interface** BlockingQueue<E> **extends** Queue <E>   17) What is the advantage of Properties file?  If you change the value in the properties file, you don't need to recompile the java class. So, it makes the application easy to manage. It is used to store information which is to be changed frequently. Consider the following example.   1. **import** java.util.\*; 2. **import** java.io.\*; 3. **public** **class** Test { 4. **public** **static** **void** main(String[] args)**throws** Exception{ 5. FileReader reader=**new** FileReader("db.properties"); 7. Properties p=**new** Properties(); 8. p.load(reader); 10. System.out.println(p.getProperty("user")); 11. System.out.println(p.getProperty("password")); 12. } 13. }   **Output**  system  oracle  18) What does the hashCode() method?  The hashCode() method returns a hash code value (an integer number).  The hashCode() method returns the same integer number if two keys (by calling equals() method) are identical.  However, it is possible that two hash code numbers can have different or the same keys.  If two objects do not produce an equal result by using the equals() method, then the hashcode() method will provide the different integer result for both the objects.  19) Why we override equals() method?  The equals method is used to check whether two objects are the same or not. It needs to be overridden if we want to check the objects based on the property.  For example, Employee is a class that has 3 data members: id, name, and salary. However, we want to check the equality of employee object by the salary. Then, we need to override the equals() method.  20) How to synchronize List, Set and Map elements?  Yes, Collections class provides methods to make List, Set or Map elements as synchronized:   |  | | --- | | public static List synchronizedList(List l){} | | public static Set synchronizedSet(Set s){} | | public static SortedSet synchronizedSortedSet(SortedSet s){} | | public static Map synchronizedMap(Map m){} | | public static SortedMap synchronizedSortedMap(SortedMap m){} |   21) What is the advantage of the generic collection?  There are three main advantages of using the generic collection.   * If we use the generic class, we don't need typecasting. * It is type-safe and checked at compile time. * Generic confirms the stability of the code by making it bug detectable at compile time.   22) What is hash-collision in Hashtable and how it is handled in Java?  Two different keys with the same hash value are known as hash-collision. Two separate entries will be kept in a single hash bucket to avoid the collision. There are two ways to avoid hash-collision.   * Separate Chaining * Open Addressing   23) What is the Dictionary class?  The Dictionary class provides the capability to store key-value pairs.  24) What is the default size of load factor in hashing based collection?  The default size of load factor is **0.75**. The default capacity is computed as initial capacity \* load factor. For example, 16 \* 0.75 = 12. So, 12 is the default capacity of Map.  25) What do you understand by fail-fast?  The Iterator in java which immediately throws ConcurrentmodificationException, if any structural modification occurs in, is called as a Fail-fast iterator. Fail-fats iterator does not require any extra space in memory.  26) [What is the difference between Array and ArrayList?](https://www.javatpoint.com/array-vs-arraylist-in-java)  The main differences between the Array and ArrayList are given below.   |  |  |  | | --- | --- | --- | | **SN** | **Array** | **ArrayList** | | 1 | The Array is of fixed size, means we cannot resize the array as per need. | ArrayList is not of the fixed size we can change the size dynamically. | | 2 | Arrays are of the static type. | ArrayList is of dynamic size. | | 3 | Arrays can store primitive data types as well as objects. | ArrayList cannot store the primitive data types it can only store the objects. |   27) [What is the difference between the length of an Array and size of ArrayList?](https://www.javatpoint.com/difference-between-length-of-array-and-size-of-arraylist-in-java)  The length of an array can be obtained using the property of length whereas ArrayList does not support length property, but we can use size() method to get the number of objects in the list.  **Finding the length of the array**   1. Int [] array = **new** **int**[4]; 2. System.out.println("The size of the array is " + array.length);   **Finding the size of the ArrayList**   1. ArrayList<String> list=**new** ArrayList<String>(); 2. list.add("ankit"); 3. list.add("nippun"); 4. System.out.println(list.size());   28) [How to convert ArrayList to Array and Array to ArrayList?](https://www.javatpoint.com/how-to-convert-arraylist-to-array-and-array-to-arraylist-in-java)  We can convert an Array to ArrayList by using the asList() method of Arrays class. asList() method is the static method of Arrays class and accepts the List object. Consider the following syntax:   1. Arrays.asList(item)   We can convert an ArrayList to Array using toArray() method of the ArrayList class. Consider the following syntax to convert the ArrayList to the List object.   1. List\_object.toArray(**new** String[List\_object.size()])   29) [How to make Java ArrayList Read-Only?](https://www.javatpoint.com/how-to-make-java-arraylist-read-only)  We can obtain java ArrayList Read-only by calling the Collections.unmodifiableCollection() method. When we define an ArrayList as Read-only then we cannot perform any modification in the collection through  add(), remove() or set() method.  30) [How to remove duplicates from ArrayList?](https://www.javatpoint.com/how-to-remove-duplicates-from-arraylist-in-java)  There are two ways to remove duplicates from the ArrayList.   * **Using HashSet:** By using HashSet we can remove the duplicate element from the ArrayList, but it will not then preserve the insertion order. * **Using LinkedHashSet:** We can also maintain the insertion order by using LinkedHashSet instead of HashSet.   The Process to remove duplicate elements from ArrayList using the LinkedHashSet:   * Copy all the elements of ArrayList to LinkedHashSet. * Empty the ArrayList using clear() method, which will remove all the elements from the list. * Now copy all the elements of LinkedHashset to ArrayList.   31) [How to reverse ArrayList?](https://www.javatpoint.com/how-to-reverse-arraylist-in-java)  To reverse an ArrayList, we can use reverse() method of Collections class. Consider the following example.   1. **import** java.util.ArrayList; 2. **import** java.util.Collection; 3. **import** java.util.Collections; 4. **import** java.util.Iterator; 5. **import** java.util.List; 6. **public** **class** ReverseArrayList { 7. **public** **static** **void** main(String[] args) { 8. List list = **new** ArrayList<>(); 9. list.add(10); 10. list.add(50); 11. list.add(30); 12. Iterator i = list.iterator(); 13. System.out.println("printing the list...."); 14. **while**(i.hasNext()) 15. { 16. System.out.println(i.next()); 17. } 18. Iterator i2 = list.iterator(); 19. Collections.reverse(list); 20. System.out.println("printing list in reverse order...."); 21. **while**(i2.hasNext()) 22. { 23. System.out.println(i2.next()); 24. } 25. } 26. }   **Output**  printing the list....  10  50  30  printing list in reverse order....  30  50  10  32) [How to sort ArrayList in descending order?](https://www.javatpoint.com/how-to-sort-java-arraylist-in-descending-order)  To sort the ArrayList in descending order, we can use the reverseOrder method of Collections class. Consider the following example.   1. **import** java.util.ArrayList; 2. **import** java.util.Collection; 3. **import** java.util.Collections; 4. **import** java.util.Comparator; 5. **import** java.util.Iterator; 6. **import** java.util.List; 8. **public** **class** ReverseArrayList { 9. **public** **static** **void** main(String[] args) { 10. List list = **new** ArrayList<>(); 11. list.add(10); 12. list.add(50); 13. list.add(30); 14. list.add(60); 15. list.add(20); 16. list.add(90); 18. Iterator i = list.iterator(); 19. System.out.println("printing the list...."); 20. **while**(i.hasNext()) 21. { 22. System.out.println(i.next()); 23. } 25. Comparator cmp = Collections.reverseOrder(); 26. Collections.sort(list,cmp); 27. System.out.println("printing list in descending order...."); 28. Iterator i2 = list.iterator(); 29. **while**(i2.hasNext()) 30. { 31. System.out.println(i2.next()); 32. } 34. } 35. }   **Output**  printing the list....  10  50  30  60  20  90  printing list in descending order....  90  60  50  30  20  10  33) [How to synchronize ArrayList?](https://www.javatpoint.com/how-to-synchronize-arraylist-in-java)  We can synchronize ArrayList in two ways.   * Using Collections.synchronizedList() method * Using CopyOnWriteArrayList<T>   34) [When to use ArrayList and LinkedList?](https://www.javatpoint.com/when-to-use-arraylist-and-linkedlist-in-java)  LinkedLists are better to use for the update operations whereas ArrayLists are better to use for the search operations.  Next Topic[JDBC interview Questions](https://www.javatpoint.com/jdbc-interview-questions)  [← Prev](https://www.javatpoint.com/java-multithreading-interview-questions)[Next →](https://www.javatpoint.com/jdbc-interview-questions)  [**1**](https://www.javatpoint.com/corejava-interview-questions) [**2**](https://www.javatpoint.com/corejava-interview-questions-2) [**3**](https://www.javatpoint.com/corejava-interview-questions-3) [**4**](https://www.javatpoint.com/corejava-interview-questions-4) [**5**](https://www.javatpoint.com/corejava-interview-questions-5) [**6**](https://www.javatpoint.com/java-multithreading-interview-questions) [**7**](https://www.javatpoint.com/java-collections-interview-questions) [**8**](https://www.javatpoint.com/jdbc-interview-questions)   |  |  | | --- | --- | | [**Java Basics Interview Questions**](https://www.javatpoint.com/corejava-interview-questions#corebasicsinterview) | [**Java OOPs Interview Questions**](https://www.javatpoint.com/corejava-interview-questions#oopsinterview) | | [**Java Multithreading Interview Questions**](https://www.javatpoint.com/java-multithreading-interview-questions) | [**Java String & Exception Interview Questions**](https://www.javatpoint.com/corejava-interview-questions-3) | | [**Java Collection Interview Questions**](https://www.javatpoint.com/java-collections-interview-questions) | [**JDBC Interview Questions**](https://www.javatpoint.com/jdbc-interview-questions) | | [**Servlet Interview Questions**](https://www.javatpoint.com/servletinterview) | [**JSP Interview Questions**](https://www.javatpoint.com/jspinterview) | | [**Spring Interview Questions**](https://www.javatpoint.com/spring-interview-questions) | [**Hibernate Interview Questions**](https://www.javatpoint.com/hibernate-interview-questions) | | [**PL/SQL Interview Questions**](https://www.javatpoint.com/pl-sql-interview-questions) | [**SQL Interview Questions**](https://www.javatpoint.com/sql-interview-questions) | | [**Oracle Interview Questions**](https://www.javatpoint.com/oracle-interview-questions) | [**Android Interview Questions**](https://www.javatpoint.com/android-interview-questions) | | [**SQL Server Interview Questions**](https://www.javatpoint.com/sql-server-interview-questions) | [**MySQL Interview Questions**](https://www.javatpoint.com/mysql-interview-questions) |   [ADVERTISEMENT BY ADRECOVER](https://adrecover.com/?utm_campaign=ByAdRecover&utm_source=www.javatpoint.com&utm_medium=ByAdRecover-336) |

**Javatpoint Services**

JavaTpoint offers too many high quality services. Mail us on hr@javatpoint.com, to get more information about given services.

* Website Designing
* Website Development
* Java Development
* PHP Development
* WordPress
* Graphic Designing
* Logo
* Digital Marketing
* On Page and Off Page SEO
* PPC
* Content Development
* Corporate Training
* Classroom and Online Training
* Data Entry

**Training For College Campus**

JavaTpoint offers college campus training on Core Java, Advance Java, .Net, Android, Hadoop, PHP, Web Technology and Python. Please mail your requirement at hr@javatpoint.com.  
Duration: 1 week to 2 week

Like/Subscribe us for latest updates or newsletter[RSS Feed](https://feeds.feedburner.com/javatpointsonoo) [Subscribe to Get Email Alerts](https://feedburner.google.com/fb/a/mailverify?uri=javatpointsonoo) [Facebook Page](https://www.facebook.com/javatpoint) [Twitter Page](https://twitter.com/pagejavatpoint) [YouTube](https://www.youtube.com/channel/UCUnYvQVCrJoFWZhKK3O2xLg) [Blog Page](https://javatpoint.blogspot.com/)

**LEARN TUTORIALS**

[Learn Java](https://www.javatpoint.com/java-tutorial)[Learn Data Structures](https://www.javatpoint.com/data-structure-tutorial)[Learn C Programming](https://www.javatpoint.com/c-programming-language-tutorial)[Learn C++ Tutorial](https://www.javatpoint.com/cpp-tutorial)[Learn C# Tutorial](https://www.javatpoint.com/c-sharp-tutorial)[Learn PHP Tutorial](https://www.javatpoint.com/php-tutorial)[Learn HTML Tutorial](https://www.javatpoint.com/html-tutorial)[Learn JavaScript Tutorial](https://www.javatpoint.com/javascript-tutorial)[Learn jQuery Tutorial](https://www.javatpoint.com/jquery-tutorial)[Learn Spring Tutorial](https://www.javatpoint.com/spring-tutorial)

**OUR WEBSITES**

[Javatpoint.com](https://www.javatpoint.com/)[Hindi100.com](https://www.hindi100.com/)[Lyricsia.com](https://www.lyricsia.com/)[Quoteperson.com](https://www.quoteperson.com/)[Jobandplacement.com](https://www.jobandplacement.com/)

**OUR SERVICES**

Website Development

Android Development

Website Designing

Digital Marketing

Summer Training

Industrial Training

College Campus Training

**CONTACT**

Address: G-13, 2nd Floor, Sec-3

Noida, UP, 201301, India

Contact No: 0120-4256464, 9990449935

[Contact Us](https://www.javatpoint.com/contact-us)[Subscribe Us](https://www.javatpoint.com/subscribe.jsp)[Privacy Policy](https://www.javatpoint.com/privacy-policy)[Sitemap](https://www.javatpoint.com/sitemap.xml)  
[About Me](https://www.javatpoint.com/sonoo-jaiswal)

© Copyright 2011-2018 www.javatpoint.com. All rights reserved. Developed by JavaTpoint.

**1. Question: What does the following Java program print?**

public class Test {

public static void main(String[] args) {

System.out.println(Math.min(Double.MIN\_VALUE, 0.0d));

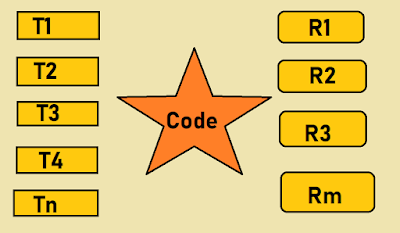
}

}

Answer: This question is tricky because unlike the [Integer](http://java67.blogspot.sg/2013/03/how-to-convert-java-string-to-int-or.html), where MIN\_VALUE is negative, both the MAX\_VALUE and MIN\_VALUE of the Double class are positive numbers. The Double.MIN\_VALUE is 2^(-1074), a double constant whose magnitude is the least among all double values.

So unlike the obvious answer, ***this program will print 0.0***because of Double.MIN\_VALUE is greater than 0. I have asked this question to a Java developer having experience of up to 3 to 5 years and surprisingly almost 70% of candidates got it wrong.

If you are not familiar with essential Java data types and wrapper classes like Double and Float then I highly recommend you to join a comprehensive Java course like [**The Complete Java Masterclass**](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fjava-the-complete-java-developer-course%2F) on Udemy to learn them, they are very very important not just for interviews but also for day to day Java work.

[](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https://www.udemy.com/java-the-complete-java-developer-course/)

**2. What will happen if you put the return statement or System.exit () on the try or catch block? Will finally block execute?**  
This is a very popular tricky Java question, and it's tricky because many programmers think that no matter what, but the [finally block](http://java67.blogspot.com/2016/06/difference-between-final-vs-finally-vs-finalize-in-java.html) will always execute. This question challenge that concept by putting a return statement in the try or catch block or calling System.exit() from try or catch block.

The answer to this tricky question in Java is that finally, the block will execute even if you put a return statement in the try block or catch block but finally block won't run if you call System.exit() from the try or catch block.  
  
  
**3. Question: Can you override a private or static method in Java?**  
Another popular Java tricky question, As I said method overriding is a good topic to ask trick questions in Java. Anyway, [you can not override a private or static method in Java](http://java67.blogspot.sg/2012/08/can-we-override-static-method-in-java.html), if you create a similar method with the same return type and same method arguments in child class then it will hide the superclass method, this is known as method hiding.

Similarly, you cannot override a private method in subclass because it's not accessible there, what you do is create another private method with the same name in the child class.

**4. Question: What do the expression 1.0 / 0.0 will return? will it throw Exception? any compile-time error?**  
Answer: This is another tricky question from Double class. Though Java developer knows about the double primitive type and Double class, while doing floating-point arithmetic they don't pay enough attention to Double.INFINITY, NaN, and -0.0 and other rules that govern the arithmetic calculations involving them.

The simple answer to this question is that it will not throw ArithmeticExcpetion and return Double.INFINITY.  
  
Also, note that the comparison x == Double.NaN always evaluates to false, even if x itself is a NaN. To test if x is a NaN, one should use the method call Double.isNaN(x) to check if the given number is NaN or not. If you know SQL, this is very close to NULL there.  
  
Btw, If you are running out of time for your interview preparation, you can also check out my book [**Grokking the Java Interview**](https://gumroad.com/l/QqjGH)for more of such popular questions,

[](https://gumroad.com/l/QqjGH)

**5. Does Java support multiple inheritances?**  
This is the trickiest question in Java if C++ can support direct multiple inheritances then why not Java is the argument Interviewer often give. The answer to this question is much more subtle than it looks like, because Java does support multiple inheritances of Type by allowing an interface to extend other interfaces, what Java doesn't support is multiple inheritances of implementation.

This distinction also gets blur because of the default method of Java 8, which now provides Java, multiple inheritances of behavior as well. See [why multiple inheritances are not supported in Java](http://javarevisited.blogspot.sg/2011/07/why-multiple-inheritances-are-not.html) to answer this tricky Java question.

**6. What will happen if we put a key object in a HashMap which is already there?**  
This tricky Java question is part of another frequently asked question, How HashMap works in Java. HashMap is also a popular topic to create a confusing and tricky question in Java.

The answer to this question is if you put the same key again then it will replace the old mapping because HashMap doesn't allow duplicate keys. The Same key will result in the same hashcode and will end up at the same position in the bucket.

Each bucket contains a linked list of Map.Entry object, which contains both Key and Value. Now Java will take the Key object from each entry and compare it with this new key using the equals() method, if that returns true then the value object in that entry will be replaced by the new value. See [How HashMap works in Java](http://java67.blogspot.sg/2013/06/how-get-method-of-hashmap-or-hashtable-works-internally.html) for more tricky Java questions from HashMap.

**7. Question: What does the following Java program print?**

public class Test {

public static void main(String[] args) throws Exception {

char[] chars = new char[] {'\u0097'};

String str = new String(chars);

byte[] bytes = str.getBytes();

System.out.println(Arrays.toString(bytes));

}

}

Answer: The trickiness of this question lies in character encoding and how String to byte array conversion works. In this program, we are first creating a String from a character array, which just has one character '\u0097', after that, we are getting the byte array from that String and print that byte.

Since \u0097 is within the 8-bit range of byte primitive type, it is reasonable to guess that the str.getBytes() call will return a byte array that contains one element with a value of -105 ((byte) 0x97).  
  
However, that's not what the program prints and that's why this question is tricky. As a matter of fact, the output of the program is operating system and locale dependent. On a Windows XP with the US locale, the above program prints [63], if you run this program on Linux or Solaris, you will get different values.  
  
To answer this question correctly, you need to know about how Unicode characters are represented in Java char values and in Java strings, and what role character encoding plays in String.getBytes().  
  
In simple word, t[o convert a string to a byte array](http://javarevisited.blogspot.sg/2014/08/2-examples-to-convert-byte-array-to-String-in-Java.html), Java iterate through all the characters that the string represents and turn each one into a number of bytes and finally put the bytes together. The rule that maps each Unicode character into a byte array is called a character encoding.

So It's possible that if the same character encoding is not used during both encoding and decoding then the retrieved value may not be correct. When we call str.getBytes() without specifying a character encoding scheme, the JVM uses the default character encoding of the platform to do the job.  
  
The default encoding scheme is an operating system and locale-dependent. On Linux, it is UTF-8 and on Windows with a US locale, the default encoding is Cp1252. This explains the output we get from running this program on Windows machines with a US locale. No matter which character encoding scheme is used, Java will always translate Unicode characters not recognized by the encoding to 63, which represents the character U+003F (the question mark, ?) in all encodings.

**8. If a method throws NullPointerException in the superclass, can we override it with a method that throws RuntimeException?**  
One more tricky Java question from the overloading and overriding concept. The answer is you can very well throw superclass of RuntimeException in overridden method, but you can not do the same if it's checked Exception. See [Rules of method overriding in Java](http://javarevisited.blogspot.sg/2011/12/method-overloading-vs-method-overriding.html) for more details.

**9. What is the issue with the following implementation of the compareTo() method in Java where an id is an integer number?**

public int compareTo(Object o){

Employee emp = (Employee) o;

return this.id - e.id;

}

Well, three is nothing wrong with this Java question until you guarantee that id is always positive. This Java question becomes tricky when you can't guarantee that the id is positive or negative. the tricky part is If id becomes negative then **subtraction may overflow** and produce an incorrect result. See [How to override the compareTo method in Java](http://javarevisited.blogspot.sg/2011/11/how-to-override-compareto-method-in.html) for the complete answer to this Java tricky question for an experienced programmer.

**10. How do you ensure that the N thread can access N resources without deadlock? (**[**answer**](http://javarevisited.blogspot.sg/2010/10/what-is-deadlock-in-java-how-to-fix-it.html)**)**  
If you are not well versed in writing multi-threading code then this is a really tricky question for you. This Java question can be tricky even for the experienced and senior programmers, who are not really exposed to deadlock and race conditions.

The key point here is ordering, if you acquire resources in a particular order and release resources in the reverse order you can prevent deadlock. If you want to prepare multithreading and concurrency in-depth, I highly recommend [**Java Multithreading for the Senior Engineering Interviews**](https://www.educative.io/courses/java-multithreading-for-senior-engineering-interviews?affiliate_id=5073518643380224) course on Educative.

[](https://www.educative.io/courses/java-multithreading-for-senior-engineering-interviews?affiliate_id=5073518643380224)

**11. Question: Consider the following Java code snippet, which is initializing two variables and both are not volatile, and two threads T1 and T2 are modifying these values as following, both are not synchronized**

int x = 0;

boolean bExit = false;

Thread 1 (not synchronized)

x = 1;

bExit = true;

Thread 2 (not synchronized)

if (bExit == true)

System.out.println("x=" + x);

**Now tell us, is it possible for Thread 2 to print “x=0”?**  
  
Answer: It's impossible for a list of tricky Java questions to not contain anything from multi-threading. This is the simplest one I can get. The answer to this question is Yes, It's possible that thread T2 may print x=0. Why? because without any instruction to compiler e.g. synchronized or volatile, bExit=true might come before x=1 in compiler reordering. Also, x=1 might not become visible in Thread 2, so Thread 2 will load x=0. Now, how do you fix it?  
  
When I asked this question to a couple of programmers they answer differently, one suggests making both threads synchronized on a common mutex, another one said to make both variables volatile. Both are correct, as they will prevent reordering and guarantee visibility.  
  
But the best answer is you just need to make bExit as volatile, then Thread 2 can only print “x=1”. x does not need to be volatile because x cannot be reordered to come after bExit=true when bExit is volatile.

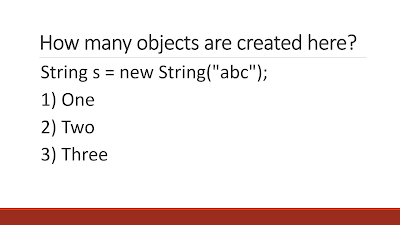
**12. What is the difference between CyclicBarrier and CountDownLatch in Java?**  
Relatively newer Java tricky question, only been introduced from Java 5. The main difference between both of them is that you can reuse CyclicBarrier even if the Barrier is broken, but you can not reuse CountDownLatch in Java. See [CyclicBarrier vs CountDownLatch in Java](http://java67.blogspot.sg/2012/08/difference-between-countdownlatch-and-cyclicbarrier-java.html) for more differences.

**13. What is the difference between StringBuffer and StringBuilder in Java?**  
Classic Java questions which some people thing tricky and some consider very easy. StringBuilder in Java was introduced in JDK 1.5, and the only difference between both of them is that StringBuffer methods like length(), capacity(), or append() are [synchronized](http://javarevisited.blogspot.sg/2011/04/synchronization-in-java-synchronized.html) while corresponding methods in StringBuilder are not synchronized.  
  
Because of this fundamental difference, concatenation of String using StringBuilder is faster than StringBuffer. Actually, it's considered a bad practice to use StringBuffer anymore, because, in almost 99% of scenarios, you perform string concatenation on the same thread. See [StringBuilder vs StringBuffer](http://javarevisited.blogspot.sg/2011/07/string-vs-stringbuffer-vs-stringbuilder.html) for more differences.

**14. Can you access a non-static variable in the static context?**  
Another tricky Java question from Java fundamentals. No, you can not access a non-static variable from the static context in Java. If you try, it will give a compile-time error. This is actually a common problem beginner in Java face when they try to access instance variables inside the main method.

Because main is static in Java, and instance variables are non-static, you can not access instance variable inside main. See, [why you can not access a non-static variable from the static method](http://javarevisited.blogspot.sg/2012/02/why-non-static-variable-cannot-be.html) to learn more about these tricky Java questions.  
  
  
**15. How many String objects are created by the following code?**

you might be thinking "one" object but that's wrong.  Btw, if you don't find these questions tricky enough, then you should check Joshua Bloch's other classic book, [Java Puzzlers](http://www.java67.com/2016/06/12-must-read-advance-java-books-for-intermediate-programmers.html) for super tricky questions. I am sure you will find them challenging enough.

[](https://4.bp.blogspot.com/-kitlSuknjow/V3xrNGYnOLI/AAAAAAAAGi0/8mkbr22IlngwqfdKzZgB_SRWkqNe0Ar_wCLcB/s1600/How+many+object+is+created+here+String+tricky+question+java.png)

Read more: <https://www.java67.com/2012/09/top-10-tricky-java-interview-questions-answers.html#ixzz6urISBJBI>

[**JAVA HUNGRY**](https://javahungry.blogspot.com/)

*Java developers tutorials and coding.*

Top of Form

Bottom of Form

* [**HOME**](https://javahungry.blogspot.com/)
* [**STRING**](https://javahungry.blogspot.com/p/string.html)
* [**COLLECTIONS**](https://javahungry.blogspot.com/2017/12/java-collections-framework-tutorial.html)
* [**INTERVIEW**](https://javahungry.blogspot.com/p/interview.html)
* [**INTERVIEW TIPS**](https://javahungry.blogspot.com/p/interview-tips.html)
* [**DATA-STRUCTURES**](https://javahungry.blogspot.com/p/data-structures-in-java.html)
* [SCJP](https://javahungry.blogspot.com/p/scjp.html)

* [THREADS](https://javahungry.blogspot.com/p/threads.html)

* [BEST JAVA BOOKS](https://javahungry.blogspot.com/2014/02/best-books-for-learning-java-must-read.html)

* [JAVA CODING PROGRAM](https://javahungry.blogspot.com/2018/09/70-top-java-interview-programs-with-coding-solutions.html)

* [MORE JAVA TOPICS](https://javahungry.blogspot.com/p/more-java-topics.html)

Top 28 Most Frequently Asked Interview Core Java Interview Questions And Answers 2018

We are sharing 28 java interview questions , these questions are frequently asked by the recruiters.Java interview questions can be asked from any core java topic . So I try my best to provide you the java interview questions and answers for experienced which should be in your to do list before facing java questions in  technical interview  .  
 **Top 28  Java Interview Questions :**  
 **1. Which two method you need to implement for key Object in HashMap ?**

In order to use any object as Key in HashMap, it must implements equals and hashcode method in Java. Read [**How HashMap works in Java**](http://javahungry.blogspot.com/2013/08/hashing-how-hash-map-works-in-java-or.html)  for detailed explanation on how equals and hashcode method is used to put and get object from HashMap.   
  
**2. What is immutable object? Can you write immutable object?**Immutable classes are Java classes whose objects can not be modified once created. Any modification in Immutable object result in new object. For example is String is immutable in Java. Mostly Immutable are also final in Java, in order to prevent sub class from overriding methods in Java which can compromise Immutability. You can achieve same functionality by making member as non final but private and not modifying them except in constructor.

**3. What is the difference between creating String as new() and literal?**

When we create string with new() Operator, it’s created in heap and not added into string pool while String created using literal are created in String pool itself which exists in PermGen area of heap.

String s = new String("Test");  
   
does not  put the object in String pool , we need to call String.intern() method which is used to put  them into String pool explicitly. its only when you create String object as String literal e.g. String s = "Test" Java automatically put that into String pool.

**4. What is**[**difference between StringBuffer and StringBuilder**](http://javahungry.blogspot.com/2013/06/difference-between-string-stringbuilder.html)**in Java ?**

Classic Java questions which some people thing tricky and some consider very easy. StringBuilder in Java is introduced in Java 5 and only difference between both of them is that Stringbuffer methods are synchronized while StringBuilder is non synchronized. See [StringBuilder vs StringBuffer](http://javahungry.blogspot.com/2013/06/difference-between-string-stringbuilder.html) for more differences.

**5.  Write code to find the First non repeated character in the String  ?**

Another good Java interview question, This question is mainly asked by Amazon and equivalent companies. See [first non repeated character in the string : Amazon interview question](http://javahungry.blogspot.com/2013/12/first-non-repeated-character-in-string-java-program-code-example.html)

**6. What is the difference between ArrayList and Vector ?**  
This question is mostly used as a start up question in Technical interviews  on the topic of Collection framework . Answer is explained in detail here [Difference between ArrayList and Vector](http://javahungry.blogspot.com/2013/12/difference-between-arraylist-and-vector-in-java-collection-interview-question.html) .

**7. How do you handle error condition  while writing stored procedure or accessing stored procedure from java?**

This is one of the tough Java interview question and its open for all, my friend didn't know the answer so he didn't mind telling me. my take is that stored procedure should return error code if some operation fails but if stored procedure itself fail than catching SQLException is only choice.

**8. What is difference between Executor.submit() and Executer.execute() method ?**

*There is a difference when looking at exception handling. If your tasks throws an exception and if it was submitted with execute this exception will go to the uncaught exception handler (when you don't have provided one explicitly, the default one will just print the stack trace to System.err). If you submitted the task with submit any thrown exception, checked exception or not, is then part of the task's return status. For a task that was submitted with submit and that terminates with an exception, the Future.get will re-throw this exception, wrapped in an ExecutionException.*

**9. What is the difference between factory and abstract factory pattern?**

*Abstract Factory provides one more level of abstraction. Consider different factories each extended from an Abstract Factory and responsible for creation of different hierarchies of objects based on the type of factory. E.g. AbstractFactory extended by AutomobileFactory, UserFactory, RoleFactory etc. Each individual factory would be responsible for creation of objects in that genre.*

You can also refer What is Factory method design pattern in Java to know more details.

**10. What is Singleton? is it better to make whole method synchronized or only critical section synchronized ?**

[Singleton in Java is a class with just one instance in whole Java application](http://javahungry.blogspot.com/2013/08/singleton-design-pattern-use-in-java.html), for example java.lang.Runtime is a Singleton class. Creating Singleton was tricky prior Java 4 but once Java 5 introduced Enum its very easy. see my article [How to create thread-safe Singleton in Java](http://javahungry.blogspot.com/2013/08/singleton-design-pattern-use-in-java.html) for more details on writing Singleton using enum and double checked locking which is purpose of this Java interview question.

**11. Can you write critical section code for singleton?**

This core Java question is followup of previous question and expecting candidate to write Java singleton using double checked locking. Remember to use volatile variable to make Singleton thread-safe.

**12. Can you write code for**[**iterating**](http://javahungry.blogspot.com/2013/06/difference-between-iterator-and-enumeration-collections-java-interview-question-with-example.html)**over HashMap in Java 7 and Java 8 ?**

Tricky one but he managed to write using while and for loop. You can find the answer here [How to iterate or loop over HashMap in Java with Example](http://javahungry.blogspot.com/2017/11/how-to-iterate-or-loop-over-hashmap-in-java-with-example.html).  
  
**13. When do you override hashcode and equals() ?**  
Whenever necessary especially if you want to do equality check or want to use your object as key in HashMap.  
  
**14. What will be the problem if you don't override hashcode() method ?**  
You will not be able to recover your object from hash Map if that is used as key in HashMap.  
See here  [How HashMap works in Java](http://javahungry.blogspot.com/2013/08/hashing-how-hash-map-works-in-java-or.html) for detailed explanation.  
  
**15. Is it better to synchronize critical section of getInstance() method or whole getInstance() method ?**  
Answer is critical section because if we lock whole method than every time some one call this method will have to wait even though we are not creating any object)  
  
**16. What is the difference when String is gets created using literal or new() operator ?**  
When we create string with new() its created in heap and not added into string pool while String created using literal are created in String pool itself which exists in Perm area of heap.  
  
**17. Does not overriding hashcode() method has any performance implication ?**  
This is a good question and open to all , as per my knowledge a poor hashcode function will result in frequent collision in HashMap which eventually increase time for adding an object into Hash Map.  
  
**18. What’s wrong using HashMap in multithreaded environment? When get() method go to infinite loop ?**  
Another good question. His answer was during concurrent access and re-sizing.  
  
**19.  What do you understand by thread-safety ? Why is it required ? And finally, how to achieve thread-safety in Java Applications ?**  
  
Java Memory Model defines the legal interaction of threads with the memory in a real computer system. In a way, it describes what behaviors are legal in multi-threaded code. It determines when a Thread can reliably see writes to variables made by other threads. It defines semantics for volatile, final & synchronized, that makes guarantee of visibility of memory operations across the Threads.  
  
Let's first discuss about Memory Barrier which are the base for our further discussions. There are two type of memory barrier instructions in JMM - read barriers and write barrier.  
  
A read barrier invalidates the local memory (cache, registers, etc) and then reads the contents from the main memory, so that changes made by other threads becomes visible to the current Thread.  
A write barrier flushes out the contents of the processor's local memory to the main memory, so that changes made by the current Thread becomes visible to the other threads.  
**JMM semantics for synchronized**  
When a thread acquires monitor of an object, by entering into a synchronized block of code, it performs a read barrier (invalidates the local memory and reads from the heap instead). Similarly exiting from a synchronized block as part of releasing the associated monitor, it performs a write barrier (flushes changes to the main memory)  
Thus modifications to a shared state using synchronized block by one Thread, is guaranteed to be visible to subsequent synchronized reads by other threads. This guarantee is provided by JMM in presence of synchronized code block.  
  
**JMM semantics for Volatile  fields**  
Read & write to volatile variables have same memory semantics as that of acquiring and releasing a monitor using synchronized code block. So the visibility of volatile field is guaranteed by the JMM. Moreover afterwards Java 1.5, volatile reads and writes are not reorderable with any other memory operations (volatile and non-volatile both). Thus when Thread A writes to a volatile variable V, and afterwards Thread B reads from variable V, any variable values that were visible to A at the time V was written are guaranteed now to be visible to B.

Let's try to understand the same using the following code

Data data = null;

volatile boolean flag = false;

Thread A

-------------

data = new Data();

flag = true; <-- writing to volatile will flush data as well as flag to main memory

Thread B

-------------

if(flag==true){ <-- as="" barrier="" data.="" flag="" font="" for="" from="" perform="" read="" reading="" volatile="" well="" will="">

use data; <!--- data is guaranteed to visible even though it is not declared volatile because of the JMM semantics of volatile flag.

}

**20.  What will happen if you call return statement or System.exit on try or catch block ? will finally block execute?**

This is a very *popular tricky Java question* and its tricky because many programmer think that finally block always executed. This question challenge that concept by putting return statement in try or catch block or calling System.exit from try or catch block. Answer of this tricky question in Java is that finally block will execute even if you put return statement in try block or catch block but finally block won't run if you call System.exit form try or catch.

**21. Can you override private or static method in Java ?**

Another popular Java tricky question, As I said method overriding is a good topic to ask trick questions in Java.  Anyway, you can not override private or static method in Java, if you create similar method with same return type and same method arguments that's called method hiding.

**22. What will happen if we put a key object in a HashMap which is already there ?**

This tricky Java questions is part of [How HashMap works in Java](http://javahungry.blogspot.com/2013/08/hashing-how-hash-map-works-in-java-or.html), which is also a popular topic to create confusing and tricky question in Java. well if you put the same key again than it will replace the old mapping because HashMap doesn't allow duplicate keys.

**23. If a method throws NullPointerException in super class, can we override it with a method which throws RuntimeException?**

One more tricky Java questions from overloading and overriding concept. Answer is you can very well throw super class of RuntimeException in overridden method but you can not do same if its checked Exception.

**24. What is the issue with following implementation of compareTo() method in Java**

public int compareTo(Object o){

   Employee emp = (Employee) emp;

   return this.id - o.id;

}

**25. How do you ensure that N thread can access N resources without deadlock**

If you are not well versed in writing multi-threading code then this is real tricky question for you. This Java question can be tricky even for experienced and senior programmer, who are not really exposed to deadlock and race conditions. Key point here is order, if you acquire resources in a particular order and release resources in reverse order you can prevent deadlock.

**26. What is difference between CyclicBarrier and CountDownLatch in Java**

Relatively newer Java tricky question, only been introduced form Java 5. Main difference between both of them is that you can reuse CyclicBarrier even if Barrier is broken but you can not reuse CountDownLatch in Java. See CyclicBarrier vs CountDownLatch in Java for more differences.

**27. Can you access non static variable in static context?**

Another tricky Java question from Java fundamentals. No you can not access static variable in non static context in Java. Read why you can not access non-static variable from static method to learn more about this tricky Java questions.  
  
**28. What is the difference between sleep() and wait() method?**  
sleep() does not release the lock while wait method release the lock. sleep() method is present in java.lang.Thread class while wait() method  is present in java.lang.Object class. For more differences  
please check [difference between sleep and wait method](http://javahungry.blogspot.com/2015/11/5-difference-between-sleep-and-wait-with-example.html).

**About The Author**

Subham Mittal has worked in Oracle for 3 years .  
For more java articles ,Click here to [Subscribe JavaHungry](https://javahungry.blogspot.com/p/subscribe-for-our-newsletter.html)

You might also like:

[[](https://javahungry.blogspot.com/2017/10/java-multithreading-interview-questions-and-answers.html)](https://javahungry.blogspot.com/2017/10/java-multithreading-interview-questions-and-answers.html)

[Top 25 Most Frequently Asked Java Multithreading Interview Questions and Answers](https://javahungry.blogspot.com/2017/10/java-multithreading-interview-questions-and-answers.html)

[[https://1.bp.blogspot.com/-l_4Sb1tEk_c/VbSfpcirsbI/AAAAAAAAAo4/zqLjasVLwWw/w120-h1-p/java%2Bweb%2Bservices%2Binterview%2Bquestions%2Band%2Banswers.jpg](https://javahungry.blogspot.com/2015/07/web-services-interview-questions-and-answers.html)](https://javahungry.blogspot.com/2015/07/web-services-interview-questions-and-answers.html)

[Top 40 Java Web Services Interview Questions and Answers](https://javahungry.blogspot.com/2015/07/web-services-interview-questions-and-answers.html)

[[https://4.bp.blogspot.com/-EA0vQgrKTf0/UkQNB27tvgI/AAAAAAAAAgo/NyDPA2f2XfQ/w120-h1-p/questions1.jpg](https://javahungry.blogspot.com/2013/09/core-java-coding-programming-questions-answers-technical-interview-java.html)](https://javahungry.blogspot.com/2013/09/core-java-coding-programming-questions-answers-technical-interview-java.html)

[Core Java Coding / Programming Questions and Answers : Technical Interview in Java](https://javahungry.blogspot.com/2013/09/core-java-coding-programming-questions-answers-technical-interview-java.html)

[[https://3.bp.blogspot.com/-VYWTK4CgLAo/VFh7oS-KhuI/AAAAAAAAAh8/st6lzc629kc/w120-h1-p/good%2Bquestions%2Bto%2Bask%2Bat%2Ban%2Binterview.jpg](https://javahungry.blogspot.com/2014/11/15-good-questions-to-ask-at-an-interview.html)](https://javahungry.blogspot.com/2014/11/15-good-questions-to-ask-at-an-interview.html)

[Top 20 Good Questions to Ask at an Interview](https://javahungry.blogspot.com/2014/11/15-good-questions-to-ask-at-an-interview.html)

[[https://4.bp.blogspot.com/-SxBnMrnzrXs/U0e5Z9JBDfI/AAAAAAAAAUc/vxusnBKhx80/w120-h1-p/behavioural+interview+questions+and+answers.jpg](https://javahungry.blogspot.com/2014/04/top-15-must-prepare-behavioral-interview-questions-and-answers-hr-round-non-technical.html)](https://javahungry.blogspot.com/2014/04/top-15-must-prepare-behavioral-interview-questions-and-answers-hr-round-non-technical.html)

[Top 15 Must Prepare Behavioral Questions and Answers (HR round) : Non Technical Java Interview Questions](https://javahungry.blogspot.com/2014/04/top-15-must-prepare-behavioral-interview-questions-and-answers-hr-round-non-technical.html)

[[https://2.bp.blogspot.com/-xwtWtaPKvRU/U0m44L5nwxI/AAAAAAAAAU4/Ve0sgn5eTWM/w120-h1-p/Hashset+interview+questions+and+answers.jpg](https://javahungry.blogspot.com/2014/04/top-10-hashset-java-interview-questions-collection.html)](https://javahungry.blogspot.com/2014/04/top-10-hashset-java-interview-questions-collection.html)

[Top 10 HashSet Java Interview Questions : Collections](https://javahungry.blogspot.com/2014/04/top-10-hashset-java-interview-questions-collection.html)

[[](https://javahungry.blogspot.com/2013/06/jdbc-interview-questions-in-java-top-15-core-java-technical-round.html)](https://javahungry.blogspot.com/2013/06/jdbc-interview-questions-in-java-top-15-core-java-technical-round.html)

[Jdbc Interview Questions And Answers for Java Developers](https://javahungry.blogspot.com/2013/06/jdbc-interview-questions-in-java-top-15-core-java-technical-round.html)

[[https://2.bp.blogspot.com/-hvL6cukvIFs/UbTHcLS_TNI/AAAAAAAAAKY/J8QMmqqlOTU/w120-h1-p/Permutation.jpg](https://javahungry.blogspot.com/2013/06/find-all-possible-permutations-of-given.html)](https://javahungry.blogspot.com/2013/06/find-all-possible-permutations-of-given.html)

[Find the all possible permutations of the given string : Java program code](https://javahungry.blogspot.com/2013/06/find-all-possible-permutations-of-given.html)

|  |  |  |
| --- | --- | --- |
|  |  |  |

[Login](https://javahungry.blogspot.com/2013/06/top-25-most-frequently-asked-core-java.html)

https://gravatar.com/avatar/undefined?d=https://embed.widgetpack.com/widget/img/avatar.png&r=PG&s=36



Comments

[Sort by Best](https://javahungry.blogspot.com/2013/06/top-25-most-frequently-asked-core-java.html)

Be the first to comment!

[ADD WIDGETPACK TO YOUR SITE](https://widgetpack.com/comment-system)

[POWERED BY WIDGET PACK™](https://widgetpack.com/)

WHATS HOT

* [Difference between Arraylist and Vector : Core Java Interview Collection Question](https://javahungry.blogspot.com/2013/12/difference-between-arraylist-and-vector-in-java-collection-interview-question.html)

* [Best Books for Learning Java](https://javahungry.blogspot.com/2014/02/best-books-for-learning-java-must-read.html)

* [Amazon Interview Question : First Non repeated character in String](https://javahungry.blogspot.com/2013/12/first-non-repeated-character-in-string-java-program-code-example.html)

* [Count total number of times each alphabet appears in the string java program code with example](https://javahungry.blogspot.com/2013/10/count-total-number-of-times-each-alphabet-appears-in-the-string.html)

* [Java 8 new features : Lambda expressions , optional class , Defender methods with examples](https://javahungry.blogspot.com/2013/10/oracle-java-8-new-features-lambda-expressions-optional-defender-methods-example.html)

[Subscribe for Our Newsletter](https://javahungry.blogspot.com/p/subscribe-for-our-newsletter.html)

POPULAR POSTS

* [Java Interview Questions](https://javahungry.blogspot.com/2013/06/top-25-most-frequently-asked-core-java.html)

* [Top 50 Java Collections Interview Questions and Answers](https://javahungry.blogspot.com/2015/05/50-java-collections-interview-questions-and-answers.html)

* [Web Services Interview Questions](https://javahungry.blogspot.com/2015/07/web-services-interview-questions-and-answers.html)

* [Java Multithreading Interview Questions and Answers](https://javahungry.blogspot.com/2017/10/java-multithreading-interview-questions-and-answers.html)

* [Java Interview Programs](https://javahungry.blogspot.com/2018/09/70-top-java-interview-programs-with-coding-solutions.html)

[Return to Top ↑](https://javahungry.blogspot.com/2013/06/top-25-most-frequently-asked-core-java.html)

[Affiliate-Disclaimer](https://javahungry.blogspot.com/p/affilliate-disclaimer.html)[Privacy](https://javahungry.blogspot.com/p/this-privacy-policy-governs-manner-in.html)[Java Hungry](https://javahungry.blogspot.com/) Copyright © 2019 · All Rights Reserved. Designed by [studiopress](https://www.studiopress.com/)



[**Interview Cake**](https://www.interviewcake.com/)

* [Interview Tips](https://www.interviewcake.com/java-interview-questions)
* [Glossary](https://www.interviewcake.com/java-interview-questions)
* [Full Course](https://www.interviewcake.com/table-of-contents)
* [**Get the full course**](https://www.interviewcake.com/upgrade)
* [Log in to save progress](https://www.interviewcake.com/java-interview-questions)

**Java Interview Questions**

How to pass the Java programming interview

**Rich** got the job at **Apple**:

***Thanks to your site, I have offers from Apple, Facebook and Google. That's a hell of an ROI!****I didn't go to school for computer science, and now I have multiple senior engineering offers from the world's best tech companies.*

**Patrick** got the job at **Dropbox**:

***Honestly, this feels like cheating.****Very efficient way to spend your study time and definitely pays for itself!*

**Cyrus** got the job at **Facebook**:

*Your site was an invaluable resource and****really helped me develop the iterative algorithmic problem solving skills big tech companies & startups are looking for.****Great case of quality over quantity.*

**Chris** got the job at **Palantir**:

*I used a number of resources to help prep for the coding interviews but****Interview Cake stood out as by far and away the most useful.****I owe you a massive debt of thanks*

**Eric** got the job at **Amazon**:

***Your site was really indispensable for the Google/Amazon interview process.****I've recommended your site to everyone I know that wants to become both a better programmer and a better interviewer. Great work.*

**Zak** got the job at **Mixpanel**:

***I got offers from 7/8 of the companies at which I interviewed.****After not going through a formal interview process in nearly a decade, your site really helped build my confidence. You're a hero, Parker ;)*

**Michael** got the job at **Google**:

*Your website was so helpful that****I didn't just receive an offer from Google, I also received a competing offer from Amazon.***

**Luna** got the job at **Apple**:

*I wish I had known about this website back when I was in algorithms class in college.****This made me finally understand a number of concepts that had eluded me for years.***

**Richie** got the job at **Microsoft**:

*Right now I have an offer from Microsoft and I'm still in the interview process with Google and LinkedIn.****Life is pretty good, and I owe so much of that to Interview Cake.***

**Akshay** got the job at **Amazon**:

*Having tried several resources for interview prep, I can honestly say that****Interview Cake provides the most structured and confidence-inducing approach.***

**Learn the algorithmic Java stuff**

In this free email course, I'll teach you the right *way of thinking* for breaking down tricky algorithmic coding interview questions.

No CS degree necessary.

Top of Form



No spam, ever. Easy unsubscribe.

Bottom of Form

×

**Java Interview Questions in This Article**

**[What are Singletons for?](https://www.interviewcake.com/java-interview-questions" \l "singletons)**

[Explain what kinds of problems singletons solve. Then implement a singleton class and test that is in fact a singleton. keep reading »](https://www.interviewcake.com/java-interview-questions" \l "singletons)

**[What's the Difference Between a String Literal and a String Object?](https://www.interviewcake.com/java-interview-questions" \l "strings)**

[How do I know which one to use in a given context? And should I ever use a naked char array instead? keep reading »](https://www.interviewcake.com/java-interview-questions" \l "strings)

**[What's the Difference Between an int and a long in Java?](https://www.interviewcake.com/java-interview-questions" \l "int-long)**

[How should I decide which one to use? keep reading »](https://www.interviewcake.com/java-interview-questions" \l "int-long)

**[What's the Difference Between an ArrayList, a LinkedList, and a Vector?](https://www.interviewcake.com/java-interview-questions" \l "arraylist-linkedlist)**

[How do I pick which one to use? keep reading »](https://www.interviewcake.com/java-interview-questions" \l "arraylist-linkedlist)

**What are singletons used for in Java applications?**

Singletons are classes which can have no more than one object. They're most useful for storing global state across a system.

Some situations where one might use a singleton include:

1. **A system-wide "global value," that many parts of the system may need to access—e.g. the software's license number**. Some software requires a valid "license" in order to run. Such software might want to make the current license available to different parts of the software system while it's running. A singleton is a good place to store that information, since there's only ever one correct answer to the question "what license are we using?"
2. **Logging**. You might want different loggers with different configurations. For example, you might want a "loud" logger that emails exceptions back to the software maintainer, to alert her of crucial issues, as well as a "quiet" logger that simply logs errors to a file on the user's system. Your software might have several components (read: several *classes*) that want to use the loud logger (e.g. payment-related stuff) and several components that want to use the *quiet* logger (e.g. caching systems—if the cache fails the system might still run correctly, just more slowly). java.util.logging.LogManager manages a set of individual loggers which are singletons. You can access them by name with getLogger(), and you can add new ones with addLogger().

Singletons are contentious these days. [Many people believe they should be avoided](http://programmers.stackexchange.com/questions/148108/why-is-global-state-so-evil), or at least be used less often than they generally are.

Even so, implementing a singleton is an interesting coding challenge.

Suppose we wanted a singleton called InstallationDetails that stored some information, including the licenseNumber. How would we implement this?

We have several options. The first is **lazy**: have the class get or create its instance just in time, as it's requested:

public final class InstallationDetails {

private static InstallationDetails INSTANCE = null;

private long licenseNumber;

public long getLicenseNumber() {

return licenseNumber;

}

// by making the constructor private, we prevent instantiation

private InstallationDetails() {

this.licenseNumber = ... ;

}

public static InstallationDetails getInstance() {

if(INSTANCE == null) {

INSTANCE = new InstallationDetails();

}

return INSTANCE;

}

}

Java

To make this thread-safe:

public final class InstallationDetails {

private static volatile InstallationDetails INSTANCE = null;

private long licenseNumber;

public long getLicenseNumber() {

return licenseNumber;

}

// by making the constructor private, we prevent instantiation

private InstallationDetails() {

this.licenseNumber = ... ;

}

public static InstallationDetails getInstance() {

if(INSTANCE == null) {

synchronized (InstallationDetails.class) {

if(INSTANCE == null) {

INSTANCE = new InstallationDetails();

}

}

}

return INSTANCE;

}

}

Java

Another is to **eagerly** have the class instantiate its singleton object even before one is requested:

public final class InstallationDetails {

private static final InstallationDetails INSTANCE = new InstallationDetails();

private long licenseNumber;

public long getLicenseNumber() {

return licenseNumber;

}

// by making the constructor private, we prevent instantiation

private InstallationDetails() {

this.licenseNumber = ... ;

}

public static InstallationDetails getInstance() {

return INSTANCE;

}

}

Java

There's also the initialization-on-demand way:

public final class InstallationDetails {

private long licenseNumber;

public long getLicenseNumber() {

return licenseNumber;

}

// by making the constructor private, we prevent instantiation

private InstallationDetails() {

this.licenseNumber = ... ;

}

private static class InstallationDetailsHolder {

private static final InstallationDetails INSTANCE = new InstallationDetails();

}

public static InstallationDetails getInstance() {

return InstallationDetailsHolder.INSTANCE;

}

}

Java

This method is lazy like the first approach, and also thread-safe.

Then there's the enum way:

public enum InstallationDetails {

INSTANCE;

private InstallationDetails() {

this.licenseNumber = ... ;

}

private long licenseNumber;

public long getLicenseNumber() {

return licenseNumber;

}

}

Java

Regardless of which method we use, we can test that our class is indeed a singleton like so:

public void testInstallationDetailsIsSingleton {

InstallationDetails obj1 = InstallationDetails.getInstance();

InstallationDetails obj2 = InstallationDetails.getInstance();

assert(obj1 == obj2);

assert(obj1.getLicenseNumber() == obj2.getLicenseNumber());

System.out.println("InstallationDetails is really a singleton!");

System.out.println("Your software license number: " + obj1.getLicenseNumber());

}

Java

**Pro Tip:** Although learning the answers to these common Java trivia questions is important, it's *so much more important to be able to quickly solve Java problems you've never seen before*. Interviewers want to see that you can do more than just memorize facts!

If you really want to take your prep to the next level, and learn the *right way of thinking* to quickly solve new problems, you should check out my 7-day email course:

Top of Form



No spam, ever. Easy unsubscribe.

Bottom of Form

**In Java, how do I decide whether to use a string literal or a string object?**

To hard-code a string in Java, we have two options. A **string literal**:

String username = "CakeLover89";

Java

And a **string object**:

String username = new String("CakeLover89");

Java

What's different about these two options?

When you use a string literal, the string is **interned**. That means it's stored in the **"string pool"** or "string intern pool". In the string pool, each string is stored no more than once. So if you have two separate variables holding the same string literals in a Java program:

String awayMessage = "I am the cake king.";

String emailSignature = "I am the cake king.";

Java

Those two Strings don't just contain the same objects in the same order, they are in fact both pointers to *the same single canonical string in the string pool*. This means they will pass an '==' check.

String awayMessage = "I am the cake king.";

String emailSignature = "I am the cake king.";

awayMessage == emailSignature; // True -- same object!

awayMessage.equals(emailSignature) // True -- same contents

Java

If our Strings were instantiated as objects, however, they would not be "interned," so they would remain separate objects (stored outside of the string pool).

String awayMessage = new String("I am the cake king.");

String emailSignature = new String("I am the cake king.");

awayMessage == emailSignature; // False -- different objects!

awayMessage.equals(emailSignature) // True -- same contents

Java

In some languages, like Lisp and Ruby, interned strings are called "symbols."

Can you intern a string "by hand?" Absolutely:

String awayMessage = new String("I am the cake king.");

String emailSignature = new String("I am the cake king.");

// intern those strings!

awayMessage = awayMessage.intern();

emailSignature = emailSignature.intern();

awayMessage == emailSignature; // True -- same object!

awayMessage.equals(emailSignature) // True -- same contents

Java

Given this, String literals can be thought of as *syntactic sugar* for instantiating a String and immediately interning it.

So which should you use, string literals or String objects?

**You should almost always use String literals.** Here's why:

**It saves time**. Comparing equality of interned strings is a constant-time operation, whereas comparing with .equals() is O(n) time.

**It saves space**. You can have several variables referencing one string while only storing that set of characters in one canonical place (the string pool).

Use String objects only if you want to be able to have two separate string objects with the same contents.

**Bonus: consider storing *sensitive* strings (like passwords) as char arrays.** This has a few nice features:

1. A char array can be "zeroed out" when you're done with it, giving *some* assurance that the string has been removed from memory (though it may still exist in some caching layers).
2. If a char array is accidentally printed (e.g. in a debug statement), by default its *address in memory* will be printed, rather than its contents.

The Swing library, for example, has a getPassword() method which always returns a char[].

**What's the difference between an int and a long in Java?**

32 bits. The difference is 32 bits :)

ints are 32-bit numbers, while longs are 64-bit numbers. This means ints take up half as much space in memory as longs, but it also means ints can't store as big of numbers as longs.

Here's a full listing of non-decimal number primitive types in Java, along with the maximum and minimum values they can hold:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **bits** | **min value** | **max value** |
| byte | 88 | -128−128 | 127127 |
| short | 1616 | -32,768−32,768 | 32,76732,767 |
| int | 3232 | -2^{31}−231 (~-2 billion) | 2^{31}-1231−1 (~2 billion) |
| long | 6464 | -2^{63}−263 (~-9 "billion billion billion") | 2^{63}-1263−1 (~9 "billion billion billion") |

Which should you use? It depends how big you expect your numbers to be. **In general, you should use the data type that's big enough to hold the numbers you expect to store, but no bigger.**

If you choose a type that's *too small*, you risk integer overflow. ↴ In Java, integers "silently" overflow—that is, no error is thrown, the integer simply goes from a very large value to a very small value. This can cause some *very* difficult-to-diagnose bugs. In Java 8, you can use Math.addExact() and Math.subtractExact() to force an exception to be thrown if the operation causes an overflow.

**What's an example of a time when 32 bits is not enough? When you're counting views on a viral video**. YouTube famously ran into trouble when the Gangnam Style video hit over 2^{31}-1231−1 views, [forcing them to upgrade their view counts from 32-bit to 64-bit numbers](http://arstechnica.com/business/2014/12/gangnam-style-overflows-int_max-forces-youtube-to-go-64-bit/).

Given the threat of integer overflow, one might be tempted to just *always* use longs, "to be safe." But you'd risk using up more space than you needed to. Specifically, if you use longs when you could be using ints, you'll use *twice as much space* as you need to. If you're dealing with a big array of numbers that takes up several gigabytes of space, a space savings of one half is huge.

Another nice side effect of using the correctly-sized data type to store your numbers is that it serves a bit of *documentation* in your code—a reminder to yourself and to other engineers about the specific range of numbers you're expecting for a given variable.

**Psssst..** Learn how to avoid the dreaded "whiteboard freeze". In our 7 day email crash course, I'll give you a strategy for quickly breaking down and solving any coding interview question.

Top of Form



No spam, ever. Easy unsubscribe.

Bottom of Form

**What's the difference between a Java ArrayList, Vector, and LinkedList? How do I pick which one to use?**

**Linked List vs Dynamic Array**

The first difference is that LinkedList is, predictably enough, an implementation of a linked list. ↴ ArrayList and Vector, on the other hand, are implementations of dynamic arrays. ↴

So LinkedList has the strengths and weaknesses of a linked list, while ArrayList and Vector have the strengths and weaknesses of a dynamic array. In particular:

**Advantages of Dynamic Arrays:**

1. **Getting the item at a specific position/index (get()) is faster**. It's O(1)*O*(1) time, vs O(n)*O*(*n*) time for a linked list
2. **They take up less space than linked lists**. In a linked list, each new "node" is a separate data structure which incurs some space overhead, whereas for a dynamic array each new item is simply another element in the underlying array. This difference is asymptotically insignificant, however—both data structures take O(n)*O*(*n*) space.
3. **They're more cache friendly**, since the elements are actually next to each-other in memory. This means that reads, especially sequential reads, often end up being much faster. Again, this difference is asymptotically insignificant.

**Advantages of Linked Lists:**

1. **Iterator.remove() is faster with a linked list**. It's O(1)*O*(1) time, vs O(n)*O*(*n*) time for a dynamic array (dynamic arrays have to "scoot over" each subsequent item to fill in the gap created by the removal).
2. **ListIterator.add() is faster with a linked list**. It's O(1)*O*(1) time, vs O(n)*O*(*n*) time for a dynamic array (dynamic arrays have to "scoot over" each subsequent item to make space for the new item).
3. **add() is always O(1)*O*(1) time**. Dynamic arrays have an *amortized* O(1)*O*(1) time cost for add(), but a *worst case* O(n)*O*(*n*) time cost, because an add() could trigger creating a new larger underlying array.

So which should you use? **Conventional wisdom is that dynamic arrays are *usually* the right choice.** The main exception is if you plan to use Iterator.remove() and/or ListIterator.add() very heavily and you *don't* plan to use get() very often. *Then* a LinkedList might be the right choice, although it may take up more memory and—because it's less cache-friendly—it may have slower reads.

**ArrayList vs Vector**

So within our options for dynamic array data structures, which one should we choose?

The main difference (though there are others) is that Vector is entirely thread-safe, because it synchronizes on each individual operation.

But **you should almost always use ArrayList, even if you're writing code that needs to be thread-safe**. The reason is that synchronizing on *each operation* is generally not the best way to make your dynamic array thread-safe. Often what you really want is to synchronize a *whole set of operations*, such as looping through the dynamic array, making some modifications as you go. If you're going to be doing multiple operations while looping through a Vector, you'll need to take out a lock for that whole series of operations *anyway*, or else another thread could modify the Vector underneath you, causing a ConcurrentModificationException.

For this reason, [Vectors are generally considered to be obsolete](http://stackoverflow.com/questions/1386275/why-is-java-vector-class-considered-obsolete-or-deprecated). Use an ArrayList instead, and manage any necessary synchronization by hand.

**Ready for more?**

If you're ready to start applying these concepts to some problems, check out [our mock coding interview questions](https://www.interviewcake.com/next).

They mimic a real interview by offering hints when you're stuck or you're missing an optimization.

**[MillionGazillion »](https://www.interviewcake.com/question/java/compress-url-list)**

[I'm making a new search engine called MillionGazillion(tm), and I need help figuring out what data structures to use. keep reading »](https://www.interviewcake.com/question/java/compress-url-list)

**[Largest Stack »](https://www.interviewcake.com/question/java/largest-stack)**

[You've implemented a Stack class, but you want to access the largest element in your stack from time to time. Write an augmented LargestStack class. keep reading »](https://www.interviewcake.com/question/java/largest-stack)

**[Balanced Binary Tree »](https://www.interviewcake.com/question/java/balanced-binary-tree)**

[Write a function to see if a binary tree is 'superbalanced'--a new tree property we just made up. keep reading »](https://www.interviewcake.com/question/java/balanced-binary-tree)

**[Binary Search Tree Checker »](https://www.interviewcake.com/question/java/bst-checker)**

[Write a function to check that a binary tree is a valid binary search tree. keep reading »](https://www.interviewcake.com/question/java/bst-checker)

**[2nd Largest Item in a Binary Search Tree »](https://www.interviewcake.com/question/java/second-largest-item-in-bst)**

[Find the second largest element in a binary search tree. keep reading »](https://www.interviewcake.com/question/java/second-largest-item-in-bst)

**[Implement A Queue With Two Stacks »](https://www.interviewcake.com/question/java/queue-two-stacks)**

[Implement a queue with two stacks. Assume you already have a stack implementation. keep reading »](https://www.interviewcake.com/question/java/queue-two-stacks)

**[Making Change »](https://www.interviewcake.com/question/java/coin)**

[Write a function that will replace your role as a cashier and make everyone rich or something. keep reading »](https://www.interviewcake.com/question/java/coin)

**[The Cake Thief »](https://www.interviewcake.com/question/java/cake-thief)**

[You've hit the mother lode: the cake vault of the Queen of England. Figure out how much of each cake to carry out to maximize profit. keep reading »](https://www.interviewcake.com/question/java/cake-thief)

**[Find Repeat, Space Edition »](https://www.interviewcake.com/question/java/find-duplicate-optimize-for-space)**

[Figure out which number is repeated. But here's the catch: optimize for space. keep reading »](https://www.interviewcake.com/question/java/find-duplicate-optimize-for-space)

**[Find Repeat, Space Edition BEAST MODE »](https://www.interviewcake.com/question/java/find-duplicate-optimize-for-space-beast-mode)**

[Figure out which number is repeated. But here's the catch: do it in linear time and constant space! keep reading »](https://www.interviewcake.com/question/java/find-duplicate-optimize-for-space-beast-mode)

**[Product of All Other Numbers »](https://www.interviewcake.com/question/java/product-of-other-numbers)**

[For each number in an array, find the product of all the other numbers. You can do it faster than you'd think! keep reading »](https://www.interviewcake.com/question/java/product-of-other-numbers)

**[Highest Product of 3 »](https://www.interviewcake.com/question/java/highest-product-of-3)**

[Find the highest possible product that you can get by multiplying any 3 numbers from an input array. keep reading »](https://www.interviewcake.com/question/java/highest-product-of-3)

**[Merging Meeting Times »](https://www.interviewcake.com/question/java/merging-ranges)**

[Write a function for merging meeting times given everyone's schedules. It's an enterprise end-to-end scheduling solution, dog. keep reading »](https://www.interviewcake.com/question/java/merging-ranges)

**[Word Cloud Data »](https://www.interviewcake.com/question/java/word-cloud)**

[You're building a word cloud. Write a function to figure out how many times each word appears so we know how big to make each word in the cloud. keep reading »](https://www.interviewcake.com/question/java/word-cloud)

**[Find in Ordered Set »](https://www.interviewcake.com/question/java/find-in-ordered-set)**

[Given an array of numbers in sorted order, how quickly could we check if a given number is present in the array? keep reading »](https://www.interviewcake.com/question/java/find-in-ordered-set)

**[Find Rotation Point »](https://www.interviewcake.com/question/java/find-rotation-point)**

[I wanted to learn some big words to make people think I'm smart, but I messed up. Write a function to help untangle the mess I made. keep reading »](https://www.interviewcake.com/question/java/find-rotation-point)

**[Inflight Entertainment »](https://www.interviewcake.com/question/java/inflight-entertainment)**

[Writing a simple recommendation algorithm that helps people choose which movies to watch during flights keep reading »](https://www.interviewcake.com/question/java/inflight-entertainment)

**[The Stolen Breakfast Drone »](https://www.interviewcake.com/question/java/find-unique-int-among-duplicates)**

[In a beautiful Amazon utopia where breakfast is delivered by drones, one drone has gone missing. Write a function to figure out which one is missing. keep reading »](https://www.interviewcake.com/question/java/find-unique-int-among-duplicates)

**[Delete Node »](https://www.interviewcake.com/question/java/delete-node)**

[Write a function to delete a node from a linked list. Turns out you can do it in constant time! keep reading »](https://www.interviewcake.com/question/java/delete-node)

**[Does This Linked List Have A Cycle? »](https://www.interviewcake.com/question/java/linked-list-cycles)**

[Check to see if a linked list has a cycle. We'll start with a simple solution and move on to some pretty tricky ones. keep reading »](https://www.interviewcake.com/question/java/linked-list-cycles)

**[Reverse A Linked List »](https://www.interviewcake.com/question/java/reverse-linked-list)**

[Write a function to reverse a linked list in place. keep reading »](https://www.interviewcake.com/question/java/reverse-linked-list)

**[Kth to Last Node in a Singly-Linked List »](https://www.interviewcake.com/question/java/kth-to-last-node-in-singly-linked-list)**

[Find the kth to last node in a singly-linked list. We'll start with a simple solution and move on to some clever tricks. keep reading »](https://www.interviewcake.com/question/java/kth-to-last-node-in-singly-linked-list)

**[Reverse String in Place »](https://www.interviewcake.com/question/java/reverse-string-in-place)**

[Write a function to reverse a string in place. keep reading »](https://www.interviewcake.com/question/java/reverse-string-in-place)

**[Reverse Words »](https://www.interviewcake.com/question/java/reverse-words)**

[Write a function to reverse the word order of a string, in place. It's to decipher a supersecret message and head off a heist. keep reading »](https://www.interviewcake.com/question/java/reverse-words)

**[Parenthesis Matching »](https://www.interviewcake.com/question/java/matching-parens)**

[Write a function that finds the corresponding closing parenthesis given the position of an opening parenthesis in a string. keep reading »](https://www.interviewcake.com/question/java/matching-parens)

**[Bracket Validator »](https://www.interviewcake.com/question/java/bracket-validator)**

[Write a super-simple JavaScript parser that can find bugs in your intern's code. keep reading »](https://www.interviewcake.com/question/java/bracket-validator)

**[Permutation Palindrome »](https://www.interviewcake.com/question/java/permutation-palindrome)**

[Check if any permutation of an input string is a palindrome. keep reading »](https://www.interviewcake.com/question/java/permutation-palindrome)

**[Recursive String Permutations »](https://www.interviewcake.com/question/java/recursive-string-permutations)**

[Write a recursive function of generating all permutations of an input string. keep reading »](https://www.interviewcake.com/question/java/recursive-string-permutations)

**[Top Scores »](https://www.interviewcake.com/question/java/top-scores)**

[Efficiently sort numbers in an array, where each number is below a certain maximum. keep reading »](https://www.interviewcake.com/question/java/top-scores)

**[Compute the nth Fibonacci Number »](https://www.interviewcake.com/question/java/nth-fibonacci)**

[Computer the nth Fibonacci number. Careful--the recursion can quickly spin out of control! keep reading »](https://www.interviewcake.com/question/java/nth-fibonacci)

**[Which Appears Twice »](https://www.interviewcake.com/question/java/which-appears-twice)**

[Find the repeat number in an array of numbers. Optimize for runtime. keep reading »](https://www.interviewcake.com/question/java/which-appears-twice)

**[In-Place Shuffle »](https://www.interviewcake.com/question/java/shuffle)**

[Do an in-place shuffle on an array of numbers. It's trickier than you might think! keep reading »](https://www.interviewcake.com/question/java/shuffle)

**[Cafe Order Checker »](https://www.interviewcake.com/question/java/cafe-order-checker)**

[Write a function to tell us if cafe customer orders are served in the same order they're paid for. keep reading »](https://www.interviewcake.com/question/java/cafe-order-checker)

**[Simulate 5-sided die »](https://www.interviewcake.com/question/java/simulate-5-sided-die)**

[Given a 7-sided die, make a 5-sided die. keep reading »](https://www.interviewcake.com/question/java/simulate-5-sided-die)

**[Simulate 7-sided die »](https://www.interviewcake.com/question/java/simulate-7-sided-die)**

[Given a 5-sided die, make a 7-sided die. keep reading »](https://www.interviewcake.com/question/java/simulate-7-sided-die)

**[Find Duplicate Files »](https://www.interviewcake.com/question/java/find-duplicate-files)**

[Your friend copied a bunch of your files and put them in random places around your hard drive. Write a function to undo the damage. keep reading »](https://www.interviewcake.com/question/java/find-duplicate-files)

**[Rectangular Love »](https://www.interviewcake.com/question/java/rectangular-love)**

[Find the area of overlap between two rectangles. In the name of love. keep reading »](https://www.interviewcake.com/question/java/rectangular-love)

**[Temperature Tracker »](https://www.interviewcake.com/question/java/temperature-tracker)**

[Write code to continually track the max, min, mean, and mode as new numbers are inserted into a tracker class. keep reading »](https://www.interviewcake.com/question/java/temperature-tracker)

**[Two Egg Problem »](https://www.interviewcake.com/question/java/two-egg-problem)**

[A building has 100 floors. Figure out the highest floor an egg can be dropped from without breaking. keep reading »](https://www.interviewcake.com/question/java/two-egg-problem)

**[Apple Stocks »](https://www.interviewcake.com/question/java/stock-price)**

[Figure out the optimal buy and sell time for a given stock, given its prices yesterday. keep reading »](https://www.interviewcake.com/question/java/stock-price)

**[Ticket Sales Site »](https://www.interviewcake.com/question/java/ticket-sales-site)**

[Design a ticket sales site, like Ticketmaster keep reading »](https://www.interviewcake.com/question/java/ticket-sales-site)

[Subscribe to our weekly question email list »](https://www.interviewcake.com/free-weekly-coding-interview-problem-newsletter)

**Programming interview questions by company:**

* [Google interview questions](https://www.interviewcake.com/google-interview-questions)
* [Facebook interview questions](https://www.interviewcake.com/facebook-interview-questions)
* [Amazon interview questions](https://www.interviewcake.com/amazon-interview-questions)
* [Uber interview questions](https://www.interviewcake.com/uber-interview-questions)
* [Microsoft interview questions](https://www.interviewcake.com/microsoft-interview-questions)
* [Apple interview questions](https://www.interviewcake.com/apple-interview-questions)
* [Netflix interview questions](https://www.interviewcake.com/netflix-interview-questions)
* [Dropbox interview questions](https://www.interviewcake.com/dropbox-interview-questions)
* [eBay interview questions](https://www.interviewcake.com/ebay-interview-questions)
* [LinkedIn interview questions](https://www.interviewcake.com/linkedin-interview-questions)
* [Oracle interview questions](https://www.interviewcake.com/oracle-interview-questions)
* [PayPal interview questions](https://www.interviewcake.com/paypal-interview-questions)
* [Yahoo interview questions](https://www.interviewcake.com/yahoo-interview-questions)

**Programming interview questions by topic:**

* [SQL interview questions](https://www.interviewcake.com/sql-interview-questions)
* [Testing and QA interview questions](https://www.interviewcake.com/testing-and-qa-interview-questions)
* [Bit manipulation interview questions](https://www.interviewcake.com/bit-manipulation)
* [Java interview questions](https://www.interviewcake.com/java-interview-questions)
* [Python interview questions](https://www.interviewcake.com/python-interview-questions)
* [Ruby interview questions](https://www.interviewcake.com/ruby-interview-questions)
* [JavaScript interview questions](https://www.interviewcake.com/javascript-interview-questions)
* [C++ interview questions](https://www.interviewcake.com/cpp-interview-questions)
* [C interview questions](https://www.interviewcake.com/c-interview-questions)
* [Swift interview questions](https://www.interviewcake.com/swift-interview-questions)
* [Objective-C interview questions](https://www.interviewcake.com/objective-c-interview-questions)
* [PHP interview questions](https://www.interviewcake.com/php-interview-questions)
* [C# interview questions](https://www.interviewcake.com/c-sharp-interview-questions)

Copyright © 2021 Cake Labs, Inc. All rights reserved.

228 Park Ave S #82632, New York, NY US 10003 (862) 294-2956

[About](https://www.interviewcake.com/about) | [Privacy](https://www.interviewcake.com/privacy-policy) | [Terms](https://www.interviewcake.com/terms-and-conditions)

[Skip to content](https://www.techbeamers.com/java-interview-questions/#main)

[**TechBeamers**](https://www.techbeamers.com/)

* Tutorials
* Interview Questions
* Programming Quizzes

Top of Form

Bottom of Form

**Java Interview Questions You Must Know**

* [Java Interview](https://www.techbeamers.com/java-interview/)
* [Meenakshi Agarwal](https://www.techbeamers.com/author/meenakshi/)

Java is one of the best high-level languages which most programmers use for developing products and testers use for automation. Also, almost 65% of the modern automation solutions use Java as their backbone. Hence, the Interviewers usually ask questions on important Java topics like Java basics, [**Java Collection Quiz**](https://www.techbeamers.com/java-collections-quiz-for-experienced/), [**Java String Quiz**](https://www.techbeamers.com/core-java-quiz-online-test/), and [**Java Threading Quiz**](https://www.techbeamers.com/java-multithreading-quiz-with-20-interview-questions/). And every software tester needs to get him/herself well-versed in these areas to succeed in testing job interviews. To help them, we bring a comprehensive list of Java interview questions so that they can confidently answer anything related to Java asked by the Interviewers.

It took us a while to come up with such a diverse set of Java interview questions. And it helped many of readers in the Software testing fraternity to gain confidence in Java skills. We know that there are a lot of testers who write quality Java code. But, they don’t focus on the theoretical part that is what required to clinch a top Software testing job interview. We hope this post would quickly help them filling this gap, and they would be able to learn just more than coding. Let’s now find out which of the Java interview questions is going to help you in the interviews.

Also, check out – [**How to Crack Java Interview?**](https://www.techbeamers.com/crack-java-interview/)

**Top Java Interview Questions**

* [**Java Basics**](https://www.techbeamers.com/java-interview-questions/)
* [**Java Classes**](https://www.techbeamers.com/java-interview-questions/)
* [**Java Strings**](https://www.techbeamers.com/java-interview-questions/)
* [**Java Threads**](https://www.techbeamers.com/java-interview-questions/)

**Top 20 Java Interview Questions - Java Basics You Must Know**

In this section, we've covered some of the key fundamental Java interview questions. It's almost sure that Interviewer can ask you 3-4 questions from this section to check your understanding of Java as a programming language.

**Q: What is JRE, and why is it required?**

JRE stands for "Java Runtime Environment" which you usually download as a Java software. The JRE comprises of the Java Virtual Machine, Java platform classes, and supporting libraries. The JRE is the runtime component of Java software and is all you need to run any Java application.

**Q: What is JDK, and why is it required?**

The JDK is a superset of the JRE and includes everything that the JRE contains. Additionally, it comes with the compilers and debuggers tools required for developing Java applications.

**Q: What is JVM, and why is it required?**

JVM stands for The Java Virtual machine. It translates and executes the Java bytecode. It's the entity which transforms Java to become a "portable language" (i.e., write once, run anywhere). Though, each platform has its implementation of JVM like the Windows, Linux, macOS, etc. have a distinct version of JVM to run bytecode.

**Q: Distinguish between the Path and Classpath?**

The <Path> and <Classpath> are OS-level environment variables. Path defines the location where the system can look up for the executables (.exe) files, and classpath specifies the location of the Java class files.

**Q: Distinguish between a constructor and method?**

A constructor gets automatically invoked to create an object, whereas the method gets called explicitly.

**Q: Is it permissible for a constructor to have a different name than its class name in Java?**

No, constructors in Java should have the same name as their classes. If the name is different, then it would behave like a standard method.

**Q: Is there any difference between an argument and a parameter?**

While defining methods, you pass variables which you refer as parameters. And when you call these methods and supply values for the variables, then they are phrased as arguments.

**Q: How your program would behave if you declare the main method as private?**

It would get compiled correctly but will throw the error "Main method not public." at runtime.

**Q: What if an application get multiple classes having main() methods?**

It's certainly possible to have multiple main methods in different classes. When you start the application, you've to provide the startup class name for execution. The JVM then looks up for the main method only in the class whose name you've supplied. Hence, you won't observe any conflict with the multiple classes having the <main()> definition.

**Q: What difference you see between pass by reference and pass by value in Java?**

Pass by reference indicates, passing the address itself rather than passing the value. Pass by value means is giving a copy of the value.

**Q: What do you understand by Byte Code?**

Java compiler generates bytecode for all the Java code and converts into class files. The bytecode is platform-independent and needs the platform-specific JVM for the execution.

**Q: What do you make of each keyword in public static void main(String args[])?**

* Public- <main()> is the entry point method which the JVM calls when a program starts. So it is mandatory to be accessible from the Java environment. Hence, the access specifier has to be public.
* Static- JVM must be capable of calling this method w/o creating an instance of the class. So the method has to be declared as static.
* Void- <main()> doesn't return anything, so its return type must be void.
* The argument string represents the argument type passed from the console, and the <args> is an array of strings specified at the command line.

**Q: How to compare the final, finally, and finalize keywords?**

* Final– It's used to declare a constant.
  + Variables defined in an interface are implicitly final.
  + You can't extend a final class.
* Finally– It makes you handle exceptions.
  + It's a keyword used for exception handling. The code under the <finally> block gets executed apparently.
* Finalize– It helps in garbage collection.
  + The <finalize()> method is used just before an object is destroyed and garbage collected.

**Q: Can you compile a Java class successfully without having the "main" method?**

Yes, we can compile, but it won't run. The "main" method works as the startup function for a Java class, and the JVM calls it for the program execution.

**Q: What do you make of System, out and <println> in the function System.out.println()?**

* System -> A predefined final class,
* out -> PrintStream object and,
* The <println> -> built-in overloaded method of the out object.

**Q: What do you understand by the explicit casting?**

It's a process which instructs the complier about transforming the object into a different type.

**e.g.** *long no = 99999;*  
*int new\_no = (int) no; // Explicit casting*

**Q: Would a Java program compile/run if we use <static public void> instead of <public static void>?**

Yes, the program will compile and run as usual.

**Q: How would you prove that an array is not null but is empty?**

Call the <Print array.length>. It will print 0. That suggests that the array is empty. If it would've been null then, it would've thrown a NullPointerException on calling the <Print array.length>.

**Q: What do you understand of Garbage Collection and how to call it explicitly?**

If the object is no longer belong to any variable, Java automatically reclaims the memory. This process is known as garbage collection. You can use the <System.gc()> method to call it explicitly.

**Q: How comes an unreachable object become reachable again, is it at all possible?**

Yes, an unreachable object may get to reachable state. It can happen if the object <finalize> method gets called during the garbage collection, and there you have set an object referring to it. This situation would cause the garbage collection to skip and make the object reachable again.

**Java Interview Questions – Final Review**

We’ve now come to the end of this post, and we would certainly like to know your feedback on the above Java interview questions. It’ll be great to hear from all of you because we write and share what we know and what could be useful for our readers. When someone leaves his/her response that instantly raise the motivations level and encourage us to continue delivering quality content.

You are also welcome to add your experience to this post. Do provide us with more quality Java interview questions, and we’ll review and include them in the above list. It would immensely help the rest of us as they may get asked similar questions during their job interviews.

Lastly, don’t forget to participate in this skill enhancement drive by sharing this fantastic list of the best Java interview questions on social media to help a large no. of buddying test engineers around the globe.

**All the very best,**

**TechBeamers**

**Read more articles**

[Previous PostTop 50 JSP Interview Questions and answers for Java Developers](https://www.techbeamers.com/best-jsp-interview-questions-answers/)

[Next PostHow to Crack a Java Interview Easily?](https://www.techbeamers.com/crack-java-interview/)

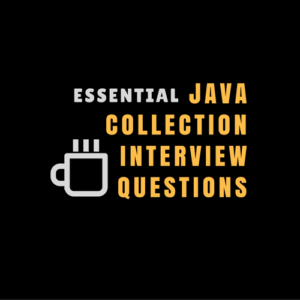
**YOU MIGHT ALSO LIKE**

[](https://www.techbeamers.com/best-jsp-interview-questions-answers/)

[**Top 50 JSP Interview Questions and answers for Java Developers**](https://www.techbeamers.com/best-jsp-interview-questions-answers/)

March 20, 2016[](https://www.techbeamers.com/crack-java-interview/)

[**How to Crack a Java Interview Easily?**](https://www.techbeamers.com/crack-java-interview/)

July 7, 2019[](https://www.techbeamers.com/java-collection-interview-questions-developers/)

[**Java Collection Interview Questions – 2019 Edition**](https://www.techbeamers.com/java-collection-interview-questions-developers/)

February 24, 2016

**Java Tutorials**

* [Java Multithreading](https://www.techbeamers.com/java-multithreading-with-examples/)
* [Java Serialization](https://www.techbeamers.com/java-serialization-tutorial/)
* [Java ProcessBuilder](https://www.techbeamers.com/java-processbuilder-example-to-run-a-bat-file-with-arguments/)
* [JNA Tutorial](https://www.techbeamers.com/write-a-simple-jna-program-in-java/)
* [Date-Time Ant Script](https://www.techbeamers.com/universal-ant-script-to-change-system-date-time/)

[View more tutorials...](https://www.techbeamers.com/java-tutorials/)

**Java Interview Questions**

* [Java Interview Questions](https://www.techbeamers.com/java-interview-questions/)
* [Java Coding Questions](https://www.techbeamers.com/java-coding-questions-assess-programming-skills/)
* [JSP Interview Questions](https://www.techbeamers.com/best-jsp-interview-questions-answers/)
* [Java Collections Questions](https://www.techbeamers.com/java-collection-interview-questions-developers/)

[View more questions...](https://www.techbeamers.com/java-interview/)

**Java Programming Quizzes**

* [Core Java Quiz](https://www.techbeamers.com/core-java-quiz-online-test/)
* [Java Developer Quiz](https://www.techbeamers.com/entry-level-java-developer-quiz/)
* [Java Collections Quiz](https://www.techbeamers.com/java-collections-quiz-for-experienced/)
* [Java Threading Quiz](https://www.techbeamers.com/java-multithreading-quiz-with-20-interview-questions/)
* [Java Serialization Quiz](https://www.techbeamers.com/java-serialization-interview-questions/)
* [Java Programming Test](https://www.techbeamers.com/java-programming-practice-test/)
* [Java Exception Quiz](https://www.techbeamers.com/java-exception-interview-questions/)
* [J2EE Online Test](https://www.techbeamers.com/j2ee-online-test-beginners/)

[View more quizzes...](https://www.techbeamers.com/java-quiz/)

TUTORIALS

* [Python Tutorial for Beginners](https://www.techbeamers.com/python-tutorial-step-by-step/)
* [Selenium Python Tutorial](https://www.techbeamers.com/selenium-webdriver-python-tutorial/)
* [Selenium Webdriver Tutorial](https://www.techbeamers.com/selenium-webdriver-tutorial/)
* [Selenium Demo Websites](https://www.techbeamers.com/websites-to-practice-selenium-webdriver-online/)
* [Python Multithreading Tutorial](https://www.techbeamers.com/python-multithreading-concepts/)
* [Java Multithreading Tutorial](https://www.techbeamers.com/java-multithreading-with-examples/)
* [Python Tips & Tricks](https://www.techbeamers.com/essential-python-tips-tricks-programmers/)

INTERVIEWS QUESTIONS

* [Python Interview Questions](https://www.techbeamers.com/python-interview-questions-programmers/)
* [SQL Query Interview Questions](https://www.techbeamers.com/sql-query-questions-answers-for-practice/)
* [Selenium Interview Questions](https://www.techbeamers.com/latest-selenium-interview-questions-and-answers/)
* [QA Interview Questions](https://www.techbeamers.com/qa-interview-questions-and-answers-top-20/)
* [Manual Testing Questions](https://www.techbeamers.com/manual-testing-interview-questions-experienced-qa/)
* [Rest API Interview Questions](https://www.techbeamers.com/rest-api-interview-questions-answers/)
* [Linux Interview Questions](https://www.techbeamers.com/essential-linux-questions-answers/)

WEB DEVELOPMENT

* [AngularJS Interview Questions](https://www.techbeamers.com/latest-angularjs-interview-questions-answers/)
* [JavaScript Interview Questions](https://www.techbeamers.com/javascript-interview-questions-answers/)
* [Web Developer Interview Questions](https://www.techbeamers.com/top-web-developer-interview-questions/)
* [NodeJS Interview Questions](https://www.techbeamers.com/top-30-node-js-interview-questions-answers/)
* [HTML/HTML5 Interview Questions](https://www.techbeamers.com/latest-html-interview-questions/)
* [PHP Interview Questions-1](https://www.techbeamers.com/latest-php-interview-questions-answers/)
* [PHP Interview Questions-2](https://www.techbeamers.com/php-interview-questions-and-answers-experienced/)

TECHBEAMERS

* [About Us](https://www.techbeamers.com/about-techbeamers/)

LEGAL

* [Disclaimer](https://www.techbeamers.com/disclaimer/)
* [Privacy Policy](https://www.techbeamers.com/privacy-policy/)

© Copyright 2015 - 2021 | TechBeamers

[▲](https://www.techbeamers.com/java-interview-questions/)

[Skip to content](https://fdk.codes/some-java-interview-questions/#skip-nav)

[fdk](https://fdk.codes/)

[Blog](https://fdk.codes/blog/)[About](https://fdk.codes/about/)

[Twitter](https://twitter.com/Frederiiiiik)

**Some Java interview questions**

11.08.2020 — [java](https://fdk.codes/tags/java/) — 1 min read

Currently, I am looking for a new job. Ideally, I'd like to work for something that is really meaningful to me. Anyways, after a one-year break from daily programming business, I thought it's time to get myself prepared again. Of course, I did not stop programming and learning - maybe I learned even more than I would if I were employed. But it is under other circumstances and very broad, I like to try out all kinds of things I read about and found interesting, be it different programming languages, new concepts, techniques, libraries or frameworks.

When you look at job postings though, most companies are looking for specialists. There are also some, that are searching for generalists, but expect them to be specialists for everything. Since I think that this is not a healthy expectation that one can and should live up to, I have thought about what I am most likely to be a specialist in, although I see myself more as a generalist.

I still don't know the answer, but I know that I worked for the most time with Java. That is what companies would also see when they look at my resume. So, I decided to refresh my java knowledge by starting to go through some interview question that circulate on the web. Finally, I made a collection of questions I liked and added some more details to the answers, but also some basic questions, or questions that led me to other questions I liked to have an answer for. I am pretty sure that there are a many more interesting questions and whenever I'll come across them, I'll extend my collection. And these questions I'd like to share with you.

**Is Java pass by reference or pass by value?**

Java is **always** pass by value.

It means the argument is copied to the parameter variable, so that the method operates on the copy. When the value of an object is passed, the **reference** to it is passed. If a reference points to a mutable object like an ArrayList and gets changed inside a method, the originally passed variable is also affected. Since passing an object as argument also means that the reference to the object is copied. As a result, both variables(original and parameter) refer to the same object.

1public class PassByPrimitive {

2

3 public static void main(String[] args) {

4 int n = 1;

5 increaseAndMultiply(n);

6 System.out.println(n); // still 1

7 }

8

9 public static void increaseAndMultiply(int n) {

10 n++;

11 System.out.println(n \* n); // 4

12 }

13

14}

1import java.util.ArrayList;

2import java.util.List;

3

4public class PassByObject {

5

6 public static void main(String[] args) {

7 var list = new ArrayList<>(List.of("foo", "bar", "foobar"));

8 mutate(list);

9 System.out.println(list); // prints [foo, bar]

10 }

11

12 // object reference is passed by

13 public static void mutate(List<String> list) {

14 list.remove("foobar");

15 }

16

17}

**What is unboxing and autoboxing in java?**

Autoboxing is the automatic conversion the java compiler makes between the primitive types and their corresponding object wrapper classes. For example, converting an int to Integer or a double to Double. If the conversion goes the other way, its called unboxing.

**What is the difference between an abstract class and an interface?**

Abstract classes specify **what an object is**, by defining characteristics of an object type. They can have a constructor and can hold a state.

Interfaces are used to establish a contract about **what an object can do**. They define capabilities that are promised to be provided by an implementing object.

**What are the different access modifiers available in Java?**

There are four types of Access modifiers:

* public – accessible from everywhere in the application
* protected – accessible within the package and the subclasses in any package
* package private (Default) – accessible strictly within the package
* private – accessible only within the same class where it is declared

**What is a final class? What is a final method?**

A final class **cannot** be extended.

A final method **cannot** be overridden.

**What is a defender Method?**

Defender Methods are default methods, which were added in Java 8 to interfaces. With them, it is possible to add new methods to interfaces without breaking existing implementations by defining a default behavior for all of them.

**What is the purpose of garbage collection in Java and when is it used?**

The purpose of garbage collection is to identify and discard those objects, that are no longer needed by the application, in order for the resources to be reclaimed and reused.

**Describe and compare fail-fast and fail-safe iterators.**

**fail-fast**: Operates directly on the collection itself. Whenever the collection is modified while iterating it throws a ConcurrentModificationException. (e.g. ArrayList, HashSet, HashMap)

**fail-save**: Operates on a cloned copy of the collection. Does not throw an exception when it gets modified while iterating. (e.g. ConcurrentHashMap, CopyOnWriteArrayList)

**What are Generics in Java ? What are advantages of using Generics?**

They were designed to extend Java's type system to allow a type or method to operate on objects of various types while providing compile-time type safety.

**What is type erasure?**

Generics are checked at compile-time for type-correctness. The generic type information is removed in a process called type erasure. For example List<Integer> will be converted to the non-generic type List containing arbitrary Objects. Because of that, type parameters cannot be determined at run-time.

**What is invariance, covariance and contravariance?**

Java’s generic type parameters are invariant. This means for any distinct types A and B, G<A> is not a subtype or supertype of G<B>. As a real world example, List<Cat> is not a supertype or subtype of List<Animal>.

1import java.util.List;

2

3public class Invariance {

4

5 public static void main(String[] args) {

6 List<Animal> animals = List.of(new Cat(), new Dog());

7 List<Cat> cats = List.of(new Cat());

8 List<Dog> dogs = List.of(new Dog());

9

10 print(animals); // works

11 print(cats); // does not work

12 print(dogs); // does not work

13 }

14

15 interface Animal {}

16

17 static class Cat implements Animal {}

18 static class Dog implements Animal {}

19

20 static void print(List<Animal> animals) {

21 animals.forEach(System.out::println);

22 }

23

24}

To achieve covariance the wildcard operator combined with an extends clause is used. The type parameter T is covariant in the generic type List<T> when A is a subtype of B and List<A> is a subtype of List<B>.

1import java.util.List;

2

3public class Covariance {

4

5 public static void main(String[] args) {

6 List<Animal> animals = List.of(new Cat(), new Dog());

7 List<Cat> cats = List.of(new Cat());

8 List<Dog> dogs = List.of(new Dog());

9

10 print(animals); // works

11 print(cats); // works

12 print(dogs); // works

13 }

14

15 interface Animal {}

16

17 static class Cat implements Animal {}

18 static class Dog implements Animal {}

19

20 static void print(List<? extends Animal> animals) {

21 animals.forEach(System.out::println);

22 }

23

24}

To achieve contravariance the wildcard operator combined with an super clause is used. The type parameter T is contravariant in the generic type List<T> when A is a subtype of B and List<B> is a subtype of List<A>.

1import java.util.List;

2

3public class Contravariance {

4

5 public static void main(String[] args) {

6 List<Animal> animals = List.of(new Cat(), new Dog());

7 List<Cat> cats = List.of(new Cat());

8 List<Dog> dogs = List.of(new Dog());

9

10 print(animals); // works

11 print(cats); // works

12 print(dogs); // does not work

13 }

14

15 interface Animal {}

16

17 static class Cat implements Animal {}

18 static class Dog implements Animal {}

19

20 static void print(List<? super Cat> cats) {

21 cats.forEach(System.out::println);

22 }

23

24}

**How does the Java memory model work?**

The JVM divides memory between **thread stacks** and the **heap**.

Each thread has its own thread stack. It contains the call stack, divided into stack frames for each method, that has been called, to reach the current point of execution. These stack frames store all local variables. A thread can only access its own thread stack.

The heap contains all Objects created in your Java application regardless of what thread created the object, including objects of wrapper classes for primitive times (e.g. Integer, Double, String).

All local variables of primitive types are fully stored on the thread stack and are not visible to other threads. A local variable may also be a reference to an object, then the reference is stored on the thread stack, but the object itself is stored on the heap. An Object may contain methods that also contain local variables, which are also stored on the thread stack, even if the object the method belongs to is stored on the heap.

An Object's member variables (fields) no matter if primitive type or object reference are stored on the heap along with the object itself. Static class variables are also stored on the heap along with the class definition. Objects on the heap can be accessed by all threads, that have a reference to it.

**What are static initializers and when would you use them?**

A static initializer gives you the opportunity to run code during the initial loading of a class and it guarantees, that this code will only run once and will finish before your class can be accessed in any way.

1import java.util.Collections;

2import java.util.HashMap;

3import java.util.Map;

4

5public class StaticInit {

6

7 private static final Map<String, Boolean> FEATURE\_FLAGS;

8

9 static {

10 final Map<String, Boolean> flags = new HashMap<>();

11 flags.put("cool-feature", true);

12 flags.put("lame-feature", false);

13 FEATURE\_FLAGS = Collections.unmodifiableMap(flags);

14 }

15

16 public static void main(String[] args) {

17 // {lame-feature=false, cool-feature=true}

18 System.out.println(FEATURE\_FLAGS);

19 }

20

21}

**What is the volatile keyword? How and why would you use it?**

The volatile modifier guarantees, that any thread that reads a field, will see the **most recently written value**.

These variables are directly written to and read from the main memory instead of the CPU cache. Reading and writing to main memory is more expensive.

If two thread are reading and writing to a shared variable, where the new value is generated based on the previous value (needs a read before writing), volatile is not enough. The short time gap between reading and writing creates a race condition, where multiple threads might read the same value and overwrite each others value. In that case you need to use synchronized keyword (or AtomicReference, AtomicInteger, ...) to guarantee atomic reads and writes.

One common use-case for using volatile is for a flag to terminate a thread.

1public class ThreadWithVolatileFlag extends Thread {

2

3 private volatile boolean close = false;

4

5 public void run() {

6 while(!close) {

7 // do work

8 }

9 }

10

11 public void close() {

12 close = true;

13 // interrupt here if needed

14 }

15}

**What is the synchronized keyword?**

Synchronized blocks can only be executed by a single thread at a time. They can thus be used to avoid **race conditions**.

Without the synchronized keyword there are no guarantees about when a variable, that is kept in a CPU register by one thread, is written to or refreshed by reading from the main memory.

By using the synchronized keyword, all variables visible to the thread are refreshed before entering the block and all changes to variables will be committed back to the main memory when leaving the block.

For more advanced locking semantics Read/Write Locks are used. There are Lock interface implementations, which can be used for reading and/or writing with an optional fairness policy for the acquisition order enabled.

**What is the ThreadLocal class? How and why would you use it?**

A ThreadLocal instance is used to individually manage a state/value per thread. Whenever it is used inside a thread, it accesses its own independently initialized copy of the variable. Each thread holds an implicit reference of a ThreadLocal variable as long as the thread is alive. It provides a simple way to make code thread safe.

1public class ThreadLocalSample {

2

3 public static void main(String[] args) throws InterruptedException {

4 var threadLocal = new ThreadLocal<Integer>();

5

6 final Runnable runnable = () -> {

7 threadLocal.set((int) (Math.random() \* 100D));

8

9 try {

10 Thread.sleep(2000);

11 } catch (InterruptedException ignored) { }

12

13 System.out.println(threadLocal.get());

14 };

15

16 final Thread t1 = new Thread(runnable);

17 final Thread t2 = new Thread(runnable);

18

19 t1.start();

20 t2.start();

21

22 t1.join();

23 t2.join();

24 }

25

26}

**What is a deadlock?**

A condition that occurs when two processes waiting for the other to complete, before they proceed. As a result, both processes wait endlessly.

**What is singleton class and how can you make a class singleton?**

An implementation of the singleton pattern must:

* ensure that only one instance of the singleton class ever exists
* provide global access to that instance

Typically, this is done by:

* declaring all constructors of the class to be private
* providing a static method that returns a reference to the instance

1public class Singleton {

2

3 private Singleton() {

4 // initialization code

5 }

6

7 private static class SingletonHolder {

8 public static final Singleton instance = new Singleton();

9 }

10

11 public static Singleton getInstance() {

12 return SingletonHolder.instance;

13 }

14

15}

The Bill Pugh singleton implementation from above, is the most widely used implementation to provide a thread safe singleton. It uses an static inner class for holding an instance of the enclosing class. This SingletonHolder class is not loaded until the getInstance() method is called. Before the calling thread is given access to it, the static instance is created as part of class loading. This means safe lazy loading without any need for synchronization/locks.

**What is the Java Producer-Consumer Problem and how can you solve it?**

The consumer-producer problem, also knows as the bounded-buffer problem, is a classic example of a multi-process synchronization problem. It describes two processes, the producer and the consumer, which share a common, fixed-size buffer used as a queue. The producers job is to generate data, put it into the buffer and start again. At the same time, the consumer is consuming the data and removing it from the buffer, one piece at a time. The problem is to make sure, that the producer won't try to add data to a full buffer and that the consumer won't try to remove data from an empty buffer.

It can be solved by synchronized access to a Queue implementation like LinkedList and the use of wait() and notify() for inter-process communication.

1import java.lang.System.Logger;

2import java.util.LinkedList;

3import java.util.Queue;

4import java.util.stream.IntStream;

5

6import static java.lang.System.Logger.Level.INFO;

7import static java.lang.System.getLogger;

8

9public class SynchronizedQueueCPP {

10

11 private static final Logger LOG = getLogger(SynchronizedQueueCPP.class.getName());

12 private static final int SIZE = 10;

13

14 public static void main(String[] args) throws InterruptedException {

15 final Queue<Integer> queue = new LinkedList<>();

16

17 var producer = new Thread(() -> {

18 IntStream.rangeClosed(1, 100)

19 .forEach(i -> {

20 synchronized (queue) {

21 while (queue.size() == SIZE) {

22 try {

23 queue.wait();

24 } catch (InterruptedException e) {

25 e.printStackTrace();

26 }

27 }

28 queue.offer(i);

29 LOG.log(INFO, "Produced: {0}", i);

30 queue.notifyAll();

31 }

32 });

33 });

34

35 var consumer = new Thread(() -> {

36 while (true) {

37 try {

38 synchronized (queue) {

39 while (queue.isEmpty()) {

40 queue.wait();

41 }

42 LOG.log(INFO, "Consumed: {0}", queue.poll());

43 queue.notifyAll();

44 }

45 } catch (InterruptedException e) {

46 e.printStackTrace();

47 }

48 }

49 });

50

51 producer.start();

52 consumer.start();

53

54 producer.join();

55 consumer.join();

56 }

57

58}

Another easier approach could be to use a BlockingQueue, that is already handling synchronization and communication internally.

1import java.lang.System.Logger;

2import java.util.concurrent.ArrayBlockingQueue;

3import java.util.concurrent.BlockingQueue;

4import java.util.stream.IntStream;

5

6import static java.lang.System.Logger.Level.INFO;

7import static java.lang.System.getLogger;

8

9public class BlockingQueueCPP {

10

11 private static final Logger LOG = getLogger(BlockingQueueCPP.class.getName());

12 private static final int SIZE = 10;

13

14 public static void main(String[] args) throws InterruptedException {

15 final BlockingQueue<Integer> queue = new ArrayBlockingQueue<>(SIZE);

16

17 var producer = new Thread(() -> {

18 IntStream.rangeClosed(1, 100)

19 .forEach(i -> {

20 try {

21 queue.put(i);

22 LOG.log(INFO, "Produced: {0}", i);

23 } catch (InterruptedException e) {

24 e.printStackTrace();

25 }

26 });

27 });

28

29 var consumer = new Thread(() -> {

30 while (true) {

31 try {

32 LOG.log(INFO, "Consumed: {0}", queue.take());

33 } catch (InterruptedException e) {

34 e.printStackTrace();

35 }

36 }

37 });

38

39 producer.start();

40 consumer.start();

41

42 producer.join();

43 consumer.join();

44 }

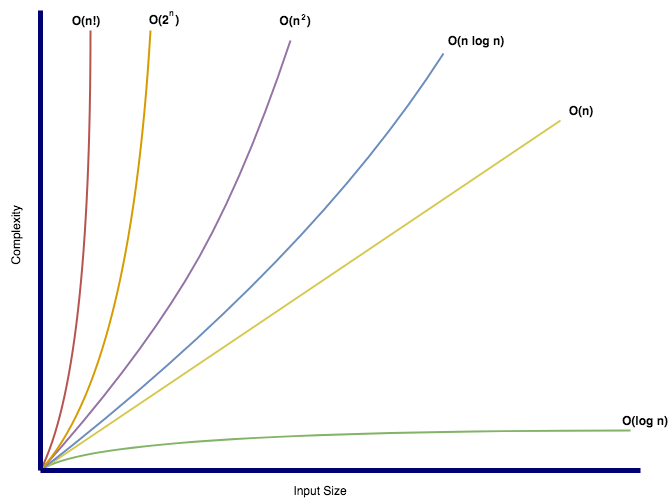
45

46}

**What do you know about the Big O notation?**

The Big O notation, also called Bachmann-Landau notation, **is a relative** **representation of the complexity of an algorithm**.

It is used for comparing algorithms according to how their run time or space requirements grow as the input size grows. Big O always assumes the worst-case. Regardless of the hardware, O(1) is always going to complete faster than O(n!).



**List**

| **Data structure** | **Access** | **Prepend** | **Insert** | **Append** | **Delete** | **Search** | **Traverse** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ArrayList | O(1) | O(n) | O(n) | O(1) / O(n)\* | O(n) | O(n) | O(n) |
| LinkedList | O(n/2) | O(1) | O(n) | O(1) | O(n) | O(n) | O(n) |
| CopyOnWriteArrayList | O(1) | O(2n) | O(2n) | O(n) | O(2n) | O(n) | O(n) |

\* when resizing is needed

**Queue**

| **Data structure** | **Peek** | **Offer** | **Poll** | **Size** |
| --- | --- | --- | --- | --- |
| LinkedList | O(1) | O(1) | O(1) | O(1) |
| PriorityQueue | O(1) | O(log(n)) | O(log(n)) | O(1) |
| ArrayDeque | O(1) | O(1) / O(n)\* | O(1) | O(1) |
| ConcurrentLinkedQueue | O(1) | O(1) | O(1) | O(n) |
| ArrayBlockingQueue | O(1) | O(1) | O(1) | O(1) |

\* when resizing is needed

**Map**

| **Data structure** | **Access** | **Insert** | **Delete** | **Next** | **Search** |
| --- | --- | --- | --- | --- | --- |
| HashMap | O(1) | O(1) / O(n)\* | O(1) | O(c/n)\*\* | O(c+n)\*\* |
| LinkedHashMap | O(1) | O(1) / O(n)\* | O(1) | O(1) | O(n) |
| TreeMap | O(log(n)) | O(log(n)) | O(log(n)) | O(log(n)) | O(log(n) |

\* when resizing is needed  
\*\* c is the table capacity

**Set**

| **Data structure** | **Insert** | **Delete** | **Next** | **Search** | **Traverse** |
| --- | --- | --- | --- | --- | --- |
| HashSet | O(1) / O(n)\* | O(1) | O(c/n)\*\* | O(c+n)\*\* | O(c+n)\*\* |
| LinkedHashSet | O(1) / O(n)\* | O(1) | O(1) | O(n) | O(n) |
| TreeSet | O(log(n)) | O(log(n)) | O(log(n)) | O(log(n)) | O(n) |

\* when resizing is needed  
\*\* c is the table capacity

**Where would you use LinkedList and where an ArrayList?**

ArrayList is to be preferred, when you have a lot of random access via List.get(n). Even more, when elements are appended instead of being inserted at a specific index. Both operations have a complexity of O(1)\* for ArrayList.

LinkedList has a higher complexity of O(n) for random access via List.get(n). But prepending elements has a lower complexity of O(1) for LinkedList compared to O(n) for ArrayList. Same applies to the removal of the first or last element in the list. Also, insertions and removals of elements at a specific index are on average faster for LinkedList.

Therefore a LinkedList is preferable when you rather have prepends, head/tail removals or removals and insertions at a specific index, while you do not have a lot of random access.

\* amortized for appending, when resizing is not necessary

**What is the Java class loader? List and explain the purpose of the three types of class loaders.**

The Java ClassLoader is part of the JRE and is used to load classes at runtime on demand (lazy-loading) into the JVM. These classes can be loaded from a local or remote file system or even from the web. When the JVM is started **3** kinds of ClassLoader are used.

1. **Bootstrap class loader** - The JVM built-in class loader, which defines the classes in a handful of critical modules, such as java.base.
2. **Platform class loader** - All classes in the Java SE Platform are guaranteed to be visible through the platform class loader. In addition, the classes in modules that are standardized under the Java Community Process but not part of the Java SE Platform are guaranteed to be visible through the platform class loader. (e.g. java.net.http)
3. **System class loader**, also known as **application class loader**, that defines classes on the application class path and module path. It is the default loader for classes in modules that are neither Java SE nor JDK modules. The platform class loader is a parent or ancestor of it, so it can load platform classes by delegating to its parent.

**Which language features have been added since Java 8? Can you name some?**

**9**

* [JPMS - Java Module Platform System](https://openjdk.java.net/jeps/261)
* [Private interface methods](https://openjdk.java.net/jeps/213)
* [Allow try-with-resources on effectively final variables](https://openjdk.java.net/jeps/213)
* [Diamond operator for anonymous classes](https://openjdk.java.net/jeps/213)
* [Collection Factory Methods](https://openjdk.java.net/jeps/269)
* [Reactive Streams Interfaces](https://openjdk.java.net/jeps/266)
* [Platform logging](https://openjdk.java.net/jeps/264)

**10**

* [Local-Variable Type Inference](https://openjdk.java.net/jeps/286)

**11**

* [HTTP/2 Client](https://openjdk.java.net/jeps/321)

**12**

* [Switch expressions](https://openjdk.java.net/jeps/325)

**13**

* [Text blocks](https://openjdk.java.net/jeps/355)

**14**

* [Records](https://openjdk.java.net/jeps/359)
* [Helpful NullPointerExceptions](https://openjdk.java.net/jeps/358)
* [Pattern Matching for instanceof](https://openjdk.java.net/jeps/305)

**15**

* [Sealed Types](https://openjdk.java.net/jeps/360)

**How to clone an object in Java?**

To clone an object in java it is necessary to implement the Cloneable marker interface and to override the Object.clone() method to make the protected clone method accessible. Inside the method the value of super.clone() is returned. By default java is doing a shallow copy of the object. That means all fields of a primitive type are getting copied, but for objects only the reference is copied. When a deep copy is created, all values of the object are copied to a newly created object, regardless of how deeply nested they are.

Shallow copy:

1public class ShallowCopy {

2

3 public static void main(String[] args) throws CloneNotSupportedException {

4 final Foo foo1 = new Foo("foo", new Bar("bar"));

5 final Foo foo2 = foo1.clone();

6

7 System.out.println(foo1.equals(foo2)); // false

8 System.out.println(foo1.bar.equals(foo2.bar)); // true

9 System.out.println(foo1.foo.equals(foo2.foo)); // true

10 System.out.println(foo1.bar.bar.equals(foo2.bar.bar)); // true

11 }

12

13 private static class Foo implements Cloneable {

14 private final String foo;

15 private final Bar bar;

16

17 private Foo(String foo, Bar bar) {

18 this.foo = foo;

19 this.bar = bar;

20 }

21

22 @Override

23 protected Foo clone() throws CloneNotSupportedException {

24 return (Foo) super.clone();

25 }

26 }

27

28 private static class Bar {

29 private final String bar;

30

31 private Bar(String bar) {

32 this.bar = bar;

33 }

34 }

35

36}

Deep copy:

1public class DeepCopy {

2

3 public static void main(String[] args) throws CloneNotSupportedException {

4 final Foo foo1 = new Foo("foo", new Bar("bar"));

5 final Foo foo2 = foo1.clone();

6

7 System.out.println(foo1.equals(foo2)); // false

8 System.out.println(foo1.bar.equals(foo2.bar)); // false

9 System.out.println(foo1.foo.equals(foo2.foo)); // true

10 System.out.println(foo1.bar.bar.equals(foo2.bar.bar)); // true

11 }

12

13 private static class Foo implements Cloneable {

14 private final String foo;

15 private Bar bar;

16

17 private Foo(String foo, Bar bar) {

18 this.foo = foo;

19 this.bar = bar;

20 }

21

22 @Override

23 protected Foo clone() throws CloneNotSupportedException {

24 final Foo clone = (Foo) super.clone();

25 clone.bar = clone.bar.clone();

26 return clone;

27 }

28 }

29

30 private static class Bar implements Cloneable {

31 private final String bar;

32

33 private Bar(String bar) {

34 this.bar = bar;

35 }

36

37 @Override

38 protected Bar clone() throws CloneNotSupportedException {

39 return (Bar) super.clone();

40 }

41 }

42

43}

**What is the JIT Compiler?**

The Just-In-Time Compiler is a component of the JVM runtime environment, that improves the performance of java application by optimizing code and compiling bytecodes to native machine code at run time.

**What is the native keyword in Java? When is it used?**

The native keyword in java is applied to a method to indicate, that a method is implemented in platform native code using JNI (Java Native Interface) / JNA (Java Native Access). They are implemented in other languages, not in java. For example, if you need to call a library from Java that is written in another language or if you need to access system or hardware resources that are only accessible from other languages (typically C).

**References**

* <https://www.toptal.com/java/interview-questions>
* <https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-3/>
* <https://medium.com/@yuhuan/covariance-and-contravariance-in-java-6d9bfb7f6b8e>
* <http://tutorials.jenkov.com/java-concurrency/java-memory-model.html>
* <http://tutorials.jenkov.com/java-concurrency/volatile.html>
* <http://tutorials.jenkov.com/java-concurrency/synchronized.html>
* <https://en.wikipedia.org/wiki/Singleton_pattern>
* <https://en.wikipedia.org/wiki/Big_O_notation>
* <https://stackoverflow.com/questions/479142/when-to-use-an-interface-instead-of-an-abstract-class-and-vice-versa>

© 2021 by fdk. All rights reserved.

Deactivate Google Analytics

[Theme](https://github.com/LekoArts/gatsby-themes/tree/master/themes/gatsby-theme-minimal-blog) by [LekoArts](https://www.lekoarts.de/en)



* [Introduction](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/index.html)
* [**1.** Object Oriented Programming (OOP)](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/oop.html)
* [**2.** General Questions about Java](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/general.html)
* [**3.** Java Threads](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/threads.html)
* [**4.** Java Collections](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/collections.html)
* [**5.** Garbage Collectors](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/collectors.html)
* [**6.** Exception Handling](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/exception.html)
* [**7.** Java Applets](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/applets.html)
* [**8.** Swing](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/swing.html)
* [**9.** JDBC](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/jdbc.html)
* [**10.** Remote Method Invocation (RMI)](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/rmi.html)
* [**11.** Servlets](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/servlets.html)
* [**12.** JSP](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/jsp.html)
* [Published using GitBook](http://www.gitbook.io/)

[115-Java-Interview-Questions-and-Answers](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/)

**Object Oriented Programming (OOP)**

Java is a computer programming language that is concurrent, class-based and object-oriented. The advantages of object oriented software development are shown below:

* Modular development of code, which leads to easy maintenance and modification.
* Reusability of code.
* Improved reliability and flexibility of code.
* Increased understanding of code.

Object-oriented programming contains many significant features, such as encapsulation, inheritance, polymorphism and [abstraction](http://www.javacodegeeks.com/2014/07/abstraction-in-java.html). We analyze each feature separately in the following sections.

**Encapsulation**

Encapsulation provides objects with the ability to hide their internal characteristics and behavior. Each object provides a number of methods, which can be accessed by other objects and change its internal data. In Java, there are three access modifiers: public, private and protected. Each modifier imposes different access rights to other classes, either in the same or in external packages. Some of the advantages of using encapsulation are listed below:

* The internal state of every objected is protected by hiding its attributes.
* It increases usability and maintenance of code, because the behavior of an object can be independently changed or extended.
* It improves modularity by preventing objects to interact with each other, in an undesired way.

You can refer to our tutorial here for more details and examples on encapsulation.

**Polymorphism**

Polymorphism is the ability of programming languages to present the same interface for differing underlying data types. A polymorphic type is a type whose operations can also be applied to values of some other type.

**Inheritance**

Inheritance provides an object with the ability to acquire the fields and methods of another class, called base class. Inheritance provides re-usability of code and can be used to add additional features to an existing class, without modifying it.

**Abstraction**

[Abstraction](http://www.javacodegeeks.com/2014/04/why-abstraction-is-really-important.html) is the process of separating ideas from specific instances and thus, develop classes in terms of their own functionality, instead of their implementation details. Java supports the creation and existence of abstract classes that expose interfaces, without including the actual implementation of all methods. The abstraction technique aims to separate the implementation details of a class from its behavior.

**Differences between Abstraction and Encapsulation**

Abstraction and encapsulation are complementary concepts. On the one hand, abstraction focuses on the behavior of an object. On the other hand, encapsulation focuses on the implementation of an object’s behavior. Encapsulation is usually achieved by hiding information about the internal state of an object and thus, can be seen as a strategy used in order to provide abstraction.



* [Introduction](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/index.html)
* [**1.** Object Oriented Programming (OOP)](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/oop.html)
* [**2.** General Questions about Java](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/general.html)
* [**3.** Java Threads](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/threads.html)
* [**4.** Java Collections](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/collections.html)
* [**5.** Garbage Collectors](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/collectors.html)
* [**6.** Exception Handling](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/exception.html)
* [**7.** Java Applets](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/applets.html)
* [**8.** Swing](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/swing.html)
* [**9.** JDBC](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/jdbc.html)
* [**10.** Remote Method Invocation (RMI)](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/rmi.html)
* [**11.** Servlets](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/servlets.html)
* [**12.** JSP](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/jsp.html)
* [Published using GitBook](http://www.gitbook.io/)

[115-Java-Interview-Questions-and-Answers](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/)

**General Questions about Java**

**1. What is JVM ? Why is Java called the “Platform Independent Programming Language” ?**

A Java virtual machine (JVM) is a process [virtual machine](http://www.javacodegeeks.com/2013/12/part-1-of-3-synopsis-of-articles-videos-on-performance-tuning-jvm-gc-in-java-mechanical-sympathy-et-al.html) that can execute Java [bytecode](http://www.javacodegeeks.com/2013/12/mastering-java-bytecode.html). Each Java source file is compiled into a bytecode file, which is executed by the JVM. Java was designed to allow application programs to be built that could be run on any platform, without having to be rewritten or recompiled by the programmer for each separate platform. A Java virtual machine makes this possible, because it is aware of the specific instruction lengths and other particularities of the underlying hardware platform.

**2. What is the Difference between JDK and JRE ?**

The Java Runtime Environment (JRE) is basically the Java Virtual Machine (JVM) where your Java programs are being executed. It also includes browser plugins for applet execution. The Java Development Kit (JDK) is the full featured Software Development Kit for Java, including the JRE, the compilers and tools (like [JavaDoc](http://docs.oracle.com/javase/7/docs/technotes/tools/windows/javadoc.html), and [Java Debugger](http://docs.oracle.com/javase/7/docs/technotes/tools/windows/jdb.html)), in order for a user to develop, compile and execute Java applications.

**3. What does the “static” keyword mean ? Can you override private or static method in Java ?**

The static keyword denotes that a member variable or method can be accessed, without requiring an instantiation of the class to which it belongs. A user cannot override [static methods in Java](http://www.javacodegeeks.com/2012/05/java-static-methods-can-be-code-smell.html), because method overriding is based upon dynamic binding at runtime and static methods are statically binded at compile time. A static method is not associated with any instance of a class so the concept is not applicable.

**4. Can you access non static variable in static context ?**

A static variable in Java belongs to its class and its value remains the same for all its instances. A static variable is initialized when the class is loaded by the JVM. If your code tries to access a non-static variable, without any instance, the compiler will complain, because those variables are not created yet and they are not associated with any instance.

**5. What are the Data Types supported by Java ? What is Autoboxing and Unboxing ?**

The eight primitive data types supported by the Java programming language are:

* byte
* short
* int
* long
* float
* double
* boolean
* char

Autoboxing is [the automatic conversion made by the Java compiler](http://www.javacodegeeks.com/2013/07/java-generics-tutorial-example-class-interface-methods-wildcards-and-much-more.html) between the primitive types and their corresponding object wrapper classes. For example, the compiler converts an int to an [Integer](http://docs.oracle.com/javase/7/docs/api/java/lang/Integer.html?is-external=true), a double to a [Double](http://docs.oracle.com/javase/7/docs/api/java/lang/Double.html), and so on. If the conversion goes the other way, this operation is called unboxing.

**6. What is Function Overriding and Overloading in Java ?**

Method overloading in Java occurs when two or more methods in the same class have the exact same name, but different parameters. On the other hand, method overriding is defined as the case when a child class redefines the same method as a parent class. Overridden methods must have the same name, argument list, and return type. The overriding method may not limit the access of the method it overrides.

**7. What is a Constructor, Constructor Overloading in Java and Copy-Constructor ?**

A constructor gets invoked when a new object is created. Every class [has a constructor](http://www.javacodegeeks.com/2014/01/which-is-better-option-cloning-or-copy-constructors.html). In case the programmer does not provide a constructor for a class, the Java compiler (Javac) creates a default constructor for that class. The constructor overloading is similar to method overloading in Java. Different constructors can be created for a single class. Each constructor must have its own unique parameter list. Finally, Java does support copy constructors like C++, but the difference lies in the fact that Java doesn’t create a default copy constructor if you don’t write your own.

**8. Does Java support multiple inheritance ?**

No, Java does not support multiple inheritance. Each class is able to extend only on one class, but is able to implement more than one interfaces.

**9. What is the difference between an Interface and an Abstract class ?**

Java provides and supports the creation both of [abstract classes](http://examples.javacodegeeks.com/java-basics/java-abstract-class-example/) and interfaces. Both implementations share some common characteristics, but they differ in the following features:

* All methods in an interface are implicitly abstract. On the other hand, an abstract class may contain both abstract and non-abstract methods.
* A class may implement a number of Interfaces, but can extend only one abstract class.
* In order for a class to implement an interface, it must implement all its declared methods. However, a class may not implement all declared methods of an abstract class. Though, in this case, the sub-class must also be declared as abstract.
* Abstract classes can implement interfaces without even providing the implementation of interface methods.
* Variables declared in a Java interface is by default final. An abstract class may contain non-final variables.
* Members of a Java interface are public by default. A member of an abstract class can either be private, protected or public.
* An interface is absolutely abstract and cannot be instantiated. An abstract class also cannot be instantiated, but can be invoked if it contains a main method.
* Also check out the [Abstract class and Interface differences for JDK 8](http://www.javacodegeeks.com/2014/04/abstract-class-versus-interface-in-the-jdk-8-era.html).

**10. What are pass by reference and pass by value ?**

When an object is passed by value, this means that a copy of the object is passed. Thus, even if changes are made to that object, it doesn’t affect the original value. When an object is passed by reference, this means that the actual object is not passed, rather a reference of the object is passed. Thus, any changes made by the external method, are also reflected in all places.



* [Introduction](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/index.html)
* [**1.** Object Oriented Programming (OOP)](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/oop.html)
* [**2.** General Questions about Java](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/general.html)
* [**3.** Java Threads](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/threads.html)
* [**4.** Java Collections](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/collections.html)
* [**5.** Garbage Collectors](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/collectors.html)
* [**6.** Exception Handling](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/exception.html)
* [**7.** Java Applets](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/applets.html)
* [**8.** Swing](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/swing.html)
* [**9.** JDBC](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/jdbc.html)
* [**10.** Remote Method Invocation (RMI)](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/rmi.html)
* [**11.** Servlets](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/servlets.html)
* [**12.** JSP](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/jsp.html)
* [Published using GitBook](http://www.gitbook.io/)

[115-Java-Interview-Questions-and-Answers](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/)

**Java Threads**

**11. What is the difference between processes and threads ?**

A process is an execution of a program, while a [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) is a single execution sequence within a process. A process can contain multiple threads. A [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) is sometimes called a lightweight process.

**12. Explain different ways of creating a thread.Which one would you prefer and why ?**

There are three ways that can be used in order for a [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) to be created:

* A class may extend the [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) class.
* A class may implement the [Runnable](http://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html) interface.
* An application can use the [Executor](http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/Executor.html) framework, in order to create a thread pool.

The [Runnable](http://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html) interface is preferred, as it does not require an object to inherit the [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) class. In case your application design requires multiple inheritance, only interfaces can help you. Also, the thread pool is very efficient and can be implemented and used very easily.

**13. Explain the available thread states in a high-level.**

During its execution, a thread can reside in one of the following states:

* Runnable: A thread becomes ready to run, but does not necessarily start running immediately.
* Running: The processor is actively executing the thread code.
* Waiting: A thread is in a blocked state waiting for some external processing to finish.
* Sleeping: The thread is forced to sleep.
* Blocked on I/O: Waiting for an I/O operation to complete.
* Blocked on Synchronization: Waiting to acquire a lock.
* Dead: The thread has finished its execution.

**14. What is the difference between a synchronized method and a synchronized block ?**

In Java programming, each object has a lock. A thread can acquire the lock for an object by using the synchronized keyword. The synchronized keyword can be applied in a method level (coarse grained lock) or block level of code (fine grained lock).

**15. How does thread synchronization occurs inside a monitor ? What levels of synchronization can you apply ?**

The JVM uses locks in conjunction with monitors. A monitor is basically a guardian that watches over a sequence of synchronized code and ensuring that only one thread at a time executes a synchronized piece of code. Each monitor is associated with an object reference. The thread is not allowed to execute the code until it obtains the lock.

**16. What’s a deadlock ?**

A condition that occurs when [two processes are waiting for each other to complete](http://www.javacodegeeks.com/2013/01/java-deadlock-example-how-to-analyze-deadlock-situation.html), before proceeding. The result is that both processes wait endlessly.

**17. How do you ensure that N threads can access N resources without deadlock ?**

A very simple way to avoid deadlock while using N threads is to impose an ordering on the locks and force each thread to follow that ordering. Thus, if all threads lock and unlock the mutexes in the same order, no deadlocks can arise.



* [Introduction](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/index.html)
* [**1.** Object Oriented Programming (OOP)](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/oop.html)
* [**2.** General Questions about Java](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/general.html)
* [**3.** Java Threads](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/threads.html)
* [**4.** Java Collections](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/collections.html)
* [**5.** Garbage Collectors](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/collectors.html)
* [**6.** Exception Handling](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/exception.html)
* [**7.** Java Applets](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/applets.html)
* [**8.** Swing](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/swing.html)
* [**9.** JDBC](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/jdbc.html)
* [**10.** Remote Method Invocation (RMI)](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/rmi.html)
* [**11.** Servlets](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/servlets.html)
* [**12.** JSP](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/jsp.html)
* [Published using GitBook](http://www.gitbook.io/)

[115-Java-Interview-Questions-and-Answers](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/)

**Java Collections**

**18. What are the basic interfaces of Java Collections Framework ?**

[Java Collections Framework](http://docs.oracle.com/javase/7/docs/technotes/guides/collections/overview.html) provides a well designed set of interfaces and classes that support operations on a collections of objects. The most basic interfaces that reside in the Java Collections Framework are:

* [Collection](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html), which represents a group of objects known as its elements.
* [Set](http://docs.oracle.com/javase/7/docs/api/java/util/Set.html), which is a collection that cannot contain duplicate elements.
* [List](http://docs.oracle.com/javase/7/docs/api/java/util/List.html), which is an ordered collection and can contain duplicate elements.
* [Map](http://docs.oracle.com/javase/7/docs/api/java/util/Map.html), which is an object that maps keys to values and cannot contain duplicate keys.

**19. Why Collection doesn’t extend Cloneable and Serializable interfaces ?**

The [Collection](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html) interface specifies groups of objects known as elements. Each concrete implementation of a [Collection](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html) can choose its own way of how to maintain and order its elements. Some collections allow duplicate keys, while some other collections don’t. The semantics and the implications of either cloning or serialization come into play when dealing with actual implementations. Thus, the concrete implementations of collections should decide how they can be cloned or serialized.

**20. What is an Iterator ?**

The [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) interface provides a number of methods that are able to iterate over any [Collection](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html). Each Java [Collection](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html) contains the [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) method that returns an [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) instance. Iterators are capable of removing elements from the underlying collection during the iteration.

**21. What differences exist between Iterator and ListIterator ?**

The differences of these elements are listed below:

* An [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) can be used to traverse the [Set](http://docs.oracle.com/javase/7/docs/api/java/util/Set.html) and [List](http://docs.oracle.com/javase/7/docs/api/java/util/List.html) collections, while the [ListIterator](http://docs.oracle.com/javase/7/docs/api/java/util/ListIterator.html) can be used to iterate only over [List](http://docs.oracle.com/javase/7/docs/api/java/util/List.html) .
* The [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) can traverse a collection only in forward direction, while the [ListIterator](http://docs.oracle.com/javase/7/docs/api/java/util/ListIterator.html) can traverse a [List](http://docs.oracle.com/javase/7/docs/api/java/util/List.html) in both directions.
* The [ListIterator](http://docs.oracle.com/javase/7/docs/api/java/util/ListIterator.html) implements the [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) interface and contains extra functionality, such as adding an element, replacing an element, getting the index position for previous and next elements, etc.

**22. What is difference between fail-fast and fail-safe ?**

The [Iterator's](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) fail-safe property works with the clone of the underlying collection and thus, it is not affected by any modification in the collection. All the collection classes in java.util package are fail-fast, while the collection classes in java.util.concurrent are fail-safe. Fail-fast iterators throw a [ConcurrentModificationException](http://examples.javacodegeeks.com/java-basics/exceptions/java-util-concurrentmodificationexception-how-to-handle-concurrent-modification-exception/), while fail-safe iterator never throws such an exception.

**23. How HashMap works in Java ?**

[A HashMap in Java stores key-value pairs](http://www.javacodegeeks.com/2014/03/how-hashmap-works-in-java.html). The [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) requires a hash function and uses hashCode and equals methods, in order to put and retrieve elements to and from the collection respectively. When the put method is invoked, the [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) calculates the hash value of the key and stores the pair in the appropriate index inside the collection. If the key exists, its value is updated with the new value. Some important characteristics of a [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) are its capacity, its load factor and the threshold resizing.

**24. What is the importance of hashCode() and equals() methods ?**

A [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) in Java uses the hashCode and equals methods to determine the index of the key-value pair. These methods are also used when we request the value of a specific key. If these methods are not implemented correctly, two different keys might produce the same hash value and thus, will be considered as equal by the collection. Furthermore, these methods are also used to detect duplicates. Thus, the implementation of both methods is crucial to the accuracy and correctness of the [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html).

**25. What differences exist between HashMap and Hashtable ?**

Both the [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) and [Hashtable](http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html) classes implement the Map interface and thus, have very similar characteristics. However, they differ in the following features:

* A [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) allows the existence of null keys and values, while a [Hashtable](http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html) doesn’t allow neither null keys, nor null values.
* A [Hashtable](http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html) is synchronized, while a [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) is not. Thus, [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) is preferred in single-threaded environments, while a [Hashtable](http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html) is suitable for multi-threaded environments.
* A [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) provides its set of keys and a Java application can iterate over them. Thus, a [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) is fail-fast. On the other hand, a [Hashtable](http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html) provides an [Enumeration](http://docs.oracle.com/javase/7/docs/api/java/util/Enumeration.html) of its keys.
* The [Hashtable](http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html) class is considered to be a legacy class.

**26. What is difference between Array and ArrayList ? When will you use Array over ArrayList ?**

The Array and ArrayList classes differ on the following features:

* [Arrays](http://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html) can contain primitive or objects, while an [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html) can contain only objects.
* [Arrays](http://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html) have fixed size, while an [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html) is dynamic.
* An [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html) provides more methods and features, such as addAll, removeAll, iterator, etc.
* For a list of primitive data types, the collections use autoboxing to reduce the coding effort. However, this approach makes them slower when working on fixed size primitive data types.

**27. What is difference between ArrayList and LinkedList ?**

Both the [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html) and [LinkedList](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html) classes implement the List interface, but they differ on the following features:

* An [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html) is an index based data structure backed by an [Array](http://docs.oracle.com/javase/7/docs/api/java/lang/reflect/Array.html). It provides random access to its elements with a performance equal to O(1). On the other hand, a [LinkedList](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html) stores its data as list of elements and every element is linked to its previous and next element. In this case, the search operation for an element has execution time equal to O(n).
* The Insertion, addition and removal operations of an element are faster in a [LinkedList](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html) compared to an [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html), because there is no need of resizing an array or updating the index when an element is added in some arbitrary position inside the collection.
* A [LinkedList](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html) consumes more memory than an [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html), because every node in a [LinkedList](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html) stores two references, one for its previous element and one for its next element. Check also our article [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html) vs. [LinkedList](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html).

**28. What is Comparable and Comparator interface ?**

List their differences. Java provides the [Comparable](http://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html) interface, which contains only one method, called [compareTo](http://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html#compareTo(T)). This method compares two objects, in order to impose an order between them. Specifically, it returns a negative integer, zero, or a positive integer to indicate that the input object is less than, equal or greater than the existing object. Java provides the [Comparator](http://docs.oracle.com/javase/7/docs/api/java/util/Comparator.html) interface, which contains two methods, called [compare](http://docs.oracle.com/javase/7/docs/api/java/util/Comparator.html#compare(T,%20T)) and [equals](http://docs.oracle.com/javase/7/docs/api/java/util/Comparator.html#equals(java.lang.Object)). The first method compares its two input arguments and imposes an order between them. It returns a negative integer, zero, or a positive integer to indicate that the first argument is less than, equal to, or greater than the second. The second method requires an object as a parameter and aims to decide whether the input object is equal to the comparator. The method returns true, only if the specified object is also a comparator and it imposes the same ordering as the comparator.

**29. What is Java Priority Queue ?**

The [PriorityQueue](http://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html) is an unbounded queue, based on a priority heap and its elements are ordered in their natural order. At the time of its creation, we can provide a Comparator that is responsible for ordering the elements of the [PriorityQueue](http://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html). A [PriorityQueue](http://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html) doesn’t allow null values, those objects that doesn’t provide natural ordering, or those objects that don’t have any comparator associated with them. Finally, the Java [PriorityQueue](http://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html) is not thread-safe and it requires O(log(n)) time for its enqueing and dequeing operations.

**30. What do you know about the big-O notation and can you give some examples with respect to different data structures ?**

The Big-O notation simply describes how well an algorithm scales or performs in the worst case scenario as the number of elements in a data structure increases. The Big-O notation can also be used to describe other behavior such as memory consumption. Since the collection classes are actually data structures, we usually use the Big-O notation to chose the best implementation to use, based on time, memory and performance. Big-O notation can give a good indication about performance for large amounts of data.

**31. What is the tradeoff between using an unordered array versus an ordered array ?**

The major advantage of an ordered array is that the search times have time complexity of O(log n), compared to that of an unordered array, which is O (n). The disadvantage of an ordered array is that the insertion operation has a time complexity of O(n), because the elements with higher values must be moved to make room for the new element. Instead, the insertion operation for an unordered array takes constant time of O(1).

**32. What are some of the best practices relating to the Java Collection framework ?**

* Choosing the right type of the collection to use, based on the application’s needs, is very crucial for its performance. For example if the size of the elements is fixed and know a priori, we shall use an [Array](http://docs.oracle.com/javase/7/docs/api/java/lang/reflect/Array.html), instead of an [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html).
* Some collection classes allow us to specify their initial capacity. Thus, if we have an estimation on the number of elements that will be stored, we can use it to avoid rehashing or resizing.
* Always use Generics for type-safety, readability, and robustness. Also, by using Generics you avoid the [ClassCastException](http://docs.oracle.com/javase/7/docs/api/java/lang/ClassCastException.html) during runtime.
* Use immutable classes provided by the Java Development Kit (JDK) as a key in a Map, in order to avoid the implementation of the [hashCode](http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#hashCode%28%29) and equals methods for our custom class.
* Program in terms of interface not implementation.
* Return zero-length collections or arrays as opposed to returning a null in case the underlying collection is actually empty.

**33. What’s the difference between Enumeration and Iterator interfaces ?**

[Enumeration](http://docs.oracle.com/javase/7/docs/api/java/util/Enumeration.html) is twice as fast as compared to an Iterator and uses very less memory. However, the [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) is much safer compared to [Enumeration](http://docs.oracle.com/javase/7/docs/api/java/util/Enumeration.html), because other threads are not able to modify the collection object that is currently traversed by the iterator. Also, [Iterators](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) allow the caller to remove elements from the underlying collection, something which is not possible with [Enumeration](http://docs.oracle.com/javase/7/docs/api/java/util/Enumeration.html).

**34. What is the difference between HashSet and TreeSet ?**

The [HashSet](http://docs.oracle.com/javase/7/docs/api/java/util/HashSet.html) is Implemented using a hash table and thus, its elements are not ordered. The add, remove, and contains methods of a [HashSet](http://docs.oracle.com/javase/7/docs/api/java/util/HashSet.html) have constant time complexity O(1). On the other hand, a [TreeSet](http://docs.oracle.com/javase/7/docs/api/java/util/TreeSet.html) is implemented using a tree structure. The elements in a [TreeSet](http://docs.oracle.com/javase/7/docs/api/java/util/TreeSet.html) are sorted, and thus, the add, remove, and contains methods have time complexity of O(logn).



* [Introduction](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/index.html)
* [**1.** Object Oriented Programming (OOP)](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/oop.html)
* [**2.** General Questions about Java](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/general.html)
* [**3.** Java Threads](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/threads.html)
* [**4.** Java Collections](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/collections.html)
* [**5.** Garbage Collectors](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/collectors.html)
* [**6.** Exception Handling](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/exception.html)
* [**7.** Java Applets](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/applets.html)
* [**8.** Swing](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/swing.html)
* [**9.** JDBC](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/jdbc.html)
* [**10.** Remote Method Invocation (RMI)](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/rmi.html)
* [**11.** Servlets](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/servlets.html)
* [**12.** JSP](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/jsp.html)
* [Published using GitBook](http://www.gitbook.io/)

[115-Java-Interview-Questions-and-Answers](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/)

**Garbage Collectors**

**35. What is the purpose of garbage collection in Java, and when is it used ?**

The purpose of garbage collection is to identify and discard those objects that are no longer needed by the application, in order for the resources to be reclaimed and reused.

**36. What does System.gc() and Runtime.gc() methods do ?**

These methods can be used as a hint to the JVM, in order to start a garbage collection. However, this it is up to the Java Virtual Machine (JVM) to start the garbage collection immediately or later in time.

**37. When is the finalize() called ? What is the purpose of finalization ?**

The finalize method is called by the garbage collector, just before releasing the object’s memory. It is normally advised to release resources held by the object inside the finalize method.

**38. If an object reference is set to null, will the Garbage Collector immediately free the memory held by that object ?**

No, the object will be available for garbage collection in the next cycle of the garbage collector.

**39. What is structure of Java Heap ? What is Perm Gen space in Heap ?**

The JVM has a heap that is the runtime data area from which memory for all class instances and arrays is allocated. It is created at the JVM start-up. Heap memory for objects is reclaimed by an automatic memory management system which is known as a garbage collector. Heap memory consists of live and dead objects. Live objects are accessible by the application and will not be a subject of garbage collection. Dead objects are those which will never be accessible by the application, but have not been collected by the garbage collector yet. Such objects occupy the heap memory space until they are eventually collected by the garbage collector.

**40. What is the difference between Serial and Throughput Garbage collector ?**

The throughput garbage collector uses a parallel version of the young generation collector and is meant to be used with applications that have medium to large data sets. On the other hand, the serial collector is usually adequate for most small applications (those requiring heaps of up to approximately 100MB on modern processors).

**41. When does an Object becomes eligible for Garbage collection in Java ?**

A Java object is subject to garbage collection when it becomes unreachable to the program in which it is currently used.

**42. Does Garbage collection occur in permanent generation space in JVM ?**

Garbage Collection does occur in PermGen space and if PermGen space is full or cross a threshold, it can trigger a full garbage collection. If you look carefully at the output of the garbage collector, you will find that PermGen space is also garbage collected. This is the reason why correct sizing of PermGen space is important to avoid frequent full garbage collections. Also check our article [Java 8: PermGen to Metaspace](http://www.javacodegeeks.com/2013/02/java-8-from-permgen-to-metaspace.html).



* [Introduction](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/index.html)
* [**1.** Object Oriented Programming (OOP)](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/oop.html)
* [**2.** General Questions about Java](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/general.html)
* [**3.** Java Threads](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/threads.html)
* [**4.** Java Collections](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/collections.html)
* [**5.** Garbage Collectors](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/collectors.html)
* [**6.** Exception Handling](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/exception.html)
* [**7.** Java Applets](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/applets.html)
* [**8.** Swing](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/swing.html)
* [**9.** JDBC](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/jdbc.html)
* [**10.** Remote Method Invocation (RMI)](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/rmi.html)
* [**11.** Servlets](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/servlets.html)
* [**12.** JSP](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/jsp.html)
* [Published using GitBook](http://www.gitbook.io/)

[115-Java-Interview-Questions-and-Answers](http://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/)

**Exception Handling**

**43. What are the two types of Exceptions in Java ? Which are the differences between them ?**

Java has two types of exceptions: checked exceptions and unchecked exceptions. Unchecked exceptions do not need to be declared in a method or a constructor’s throws clause, if they can be thrown by the execution of the method or the constructor, and propagate outside the method or constructor boundary. On the other hand, checked exceptions must be declared in a method or a constructor’s throws clause. See here for tips on [Java exception handling](http://www.javacodegeeks.com/2013/07/java-exception-handling-tutorial-with-examples-and-best-practices.html).

**44. What is the difference between Exception and Error in java ?**

[Exception](http://docs.oracle.com/javase/7/docs/api/java/lang/Exception.html) and [Error](http://docs.oracle.com/javase/7/docs/api/java/lang/Error.html) classes are both subclasses of the [Throwable](http://docs.oracle.com/javase/7/docs/api/java/lang/Throwable.html) class. The [Exception](http://docs.oracle.com/javase/7/docs/api/java/lang/Exception.html) class is used for exceptional conditions that a user’s program should catch. The [Error](http://docs.oracle.com/javase/7/docs/api/java/lang/Error.html) class defines exceptions that are not excepted to be caught by the user program.

**45. What is the difference between throw and throws ?**

The throw keyword is used to explicitly raise a exception within the program. On the contrary, the throws clause is used to indicate those exceptions that are not handled by a method. Each method must explicitly specify which exceptions does not handle, so the callers of that method can guard against possible exceptions. Finally, multiple exceptions are separated by a comma.

**45. What is the importance of finally block in exception handling ?**

A finally block will always be executed, whether or not an exception is actually thrown. Even in the case where the catch statement is missing and an exception is thrown, the finally block will still be executed. Last thing to mention is that the finally block is used to release resources like I/O buffers, database connections, etc.

**46. What will happen to the Exception object after exception handling ?**

The [Exception](http://docs.oracle.com/javase/7/docs/api/java/lang/Exception.html) object will be garbage collected in the next garbage collection.

**47. How does finally block differ from finalize() method ?**

A finally block will be executed whether or not an exception is thrown and is used to release those resources held by the application. Finalize is a protected method of the Object class, which is called by the Java Virtual Machine (JVM) just before an object is garbage collected.

* [**Scaler AcademyNew**](https://www.scaler.com/academy/?utm_source=ib&utm_medium=headermenu&utm_campaign=/java-interview-questions)
* [**Practice**](https://www.interviewbit.com/practice/)
* [**Referrals**](https://www.interviewbit.com/referrals/)
* [**Contests**](https://www.interviewbit.com/contests/)
* [**Log In**](https://www.interviewbit.com/users/sign_in/)
* [**Sign Up**](https://www.interviewbit.com/users/sign_up/)
* **Looking to hire?** [We can help](https://www.interviewbit.com/hire/)



*Free Academy Experience*

**Preparing for tech interviews?** Learn from the best instructors in the field.

Try Academy for **FREE!**

[**Join Now!**](https://scaler.com/academy?utm_medium=nudge_modal&utm_campaign=free_academy&utm_source=ib&utm_content=/java-interview-questions)

**Java Interview Questions**

Share with:



Copied

* [**Java Basic Interview Questions**](https://www.interviewbit.com/java-interview-questions/#java-basic-questions)
  + [1.Why is Java a platform independent language?](https://www.interviewbit.com/java-interview-questions/#java-platform-independent)
  + [2.Why is Java not a pure object oriented language?](https://www.interviewbit.com/java-interview-questions/#java-not-purely-object-oriented-language)
  + [3.Pointers are used in C/ C++. Why does Java not make use of pointers?](https://www.interviewbit.com/java-interview-questions/#why-dont-we-use-pointers-in-java)
  + [4.What do you understand by an instance variable and a local variable?](https://www.interviewbit.com/java-interview-questions/#instance-variables-and-local-variables-in-java)
  + [5.What do you mean by data encapsulation?](https://www.interviewbit.com/java-interview-questions/#what-is-data-encapsulation)
  + [6.Tell us something about JIT compiler.](https://www.interviewbit.com/java-interview-questions/#just-in-time-compiler-explained)
  + [7.Can you tell the difference between equals() method and equality operator (==) in Java?](https://www.interviewbit.com/java-interview-questions/#difference-between-equality-operator-and-equals-method-java)
  + [8.How is an infinite loop declared in Java?](https://www.interviewbit.com/java-interview-questions/#infinite-loops-in-java)
  + [9.Briefly explain the concept of constructor overloading](https://www.interviewbit.com/java-interview-questions/#constructor-overloading-in-java)
  + [10.Comment on method overloading and overriding by citing relevant examples.](https://www.interviewbit.com/java-interview-questions/#method-overloading-and-overriding-in-java-with-example)
  + [11.A single try block and multiple catch blocks can co-exist in a Java Program. Explain.](https://www.interviewbit.com/java-interview-questions/#single-try-block-and-multiple-catch-blocks-coexist-in-java)
  + [12.Explain the use of final keyword in variable, method and class.](https://www.interviewbit.com/java-interview-questions/#final-keyword-in-java)
  + [13.Do final, finally and finalize keywords have the same function?](https://www.interviewbit.com/java-interview-questions/#difference-between-final-finally-and-finalize)
  + [14.When can you use super keyword?](https://www.interviewbit.com/java-interview-questions/#super-keyword-in-java)
  + [15.Can the static methods be overloaded?](https://www.interviewbit.com/java-interview-questions/#can-we-overload-static-methods)
  + [16.Can the static methods be overridden?](https://www.interviewbit.com/java-interview-questions/#can-static-methods-be-overridden)
  + [17.What is the main objective of garbage collection?](https://www.interviewbit.com/java-interview-questions/#main-objective-of-java-garbage-collection)
  + [18.What part of memory - Stack or Heap - is cleaned in garbage collection process?](https://www.interviewbit.com/java-interview-questions/#does-garbage-collector-clean-stack-or-heap)
* [**Java Intermediate Interview Questions**](https://www.interviewbit.com/java-interview-questions/#java-intermediate-questions)
  + [19.Apart from the security aspect, what are the reasons behind making strings immutable in Java?](https://www.interviewbit.com/java-interview-questions/#reasons-behind-making-strings-immutable-in-java)
  + [20.How would you differentiate between a String, StringBuffer, and a StringBuilder?](https://www.interviewbit.com/java-interview-questions/#string-vs-stringbuffer-vs-stringbuilder-in-java)
  + [21.Using relevant properties highlight the differences between interfaces and abstract classes.](https://www.interviewbit.com/java-interview-questions/#difference-between-abstract-classses-and-interfaces)
  + [22.In Java, static as well as private method overriding is possible. Comment on the statement.](https://www.interviewbit.com/java-interview-questions/#can-we-override-a-static-or-private-method-in-java)
  + [23.What makes a HashSet different from a TreeSet?](https://www.interviewbit.com/java-interview-questions/#hashset-vs-treeset-in-java)
  + [24.Why is the character array preferred over string for storing confidential information?](https://www.interviewbit.com/java-interview-questions/#why-char-array-preferred-to-store-confidential-data-in-java)
  + [25.What are the differences between JVM, JRE and JDK in Java?](https://www.interviewbit.com/java-interview-questions/#jvm-vs-jre-vs-jdk-in-java)
  + [26.What are the differences between HashMap and HashTable in Java?](https://www.interviewbit.com/java-interview-questions/#difference-between-hashmap-and-hashtable-in-java)
  + [27.What is the importance of reflection in Java?](https://www.interviewbit.com/java-interview-questions/#importance-of-reflection-in-java)
  + [28.What are the different ways of threads usage?](https://www.interviewbit.com/java-interview-questions/#different-ways-of-java-threads-usage)
  + [29.What are the differences between constructor and method of a class in Java?](https://www.interviewbit.com/java-interview-questions/#difference-between-constructor-and-method-in-java)
  + [30.Java works as “pass by value” or “pass by reference” phenomenon?](https://www.interviewbit.com/java-interview-questions/#does-java-pass-by-value-or-pass-by-reference)
  + [31.Which among String or String Buffer should be preferred when there are lot of updates required to be done in the data?](https://www.interviewbit.com/java-interview-questions/#string-or-stringbuffer-is-preferred-when-alot-of-updates-has-to-be-done-in-the-data)
  + [32.How to not allow serialization of attributes of a class in Java?](https://www.interviewbit.com/java-interview-questions/#how-to-avoid-serialization-of-attributes-of-a-class-in-java)
  + [33.What happens if the static modifier is not included in the main method signature in Java?](https://www.interviewbit.com/java-interview-questions/#what-if-the-static-modifier-is-removed-from-the-main-method-signature-in-java)
  + [34.What happens if there are multiple main methods inside one class in Java?](https://www.interviewbit.com/java-interview-questions/#what-happens-if-there-are-multiple-main-methods-inside-one-class-in-java)
  + [35.What do you understand by Object Cloning and how do you achieve it in Java?](https://www.interviewbit.com/java-interview-questions/#object-cloning-in-java)
  + [36.How does an exception propagate in the code?](https://www.interviewbit.com/java-interview-questions/#exception-propagation-java)
  + [37.Is it mandatory for a catch block to be followed after a try block?](https://www.interviewbit.com/java-interview-questions/#is-it-mandatory-for-a-catch-block-to-be-followed-after-a-try-block-in-java)
  + [38.Will the finally block get executed when the return statement is written at the end of try block and catch block as shown below?](https://www.interviewbit.com/java-interview-questions/#will-finally-block-get-executed-when-the-return-statement-is-written-at-the-end-of-try-block-and-catch-block)
  + [39.Can you call a constructor of a class inside the another constructor?](https://www.interviewbit.com/java-interview-questions/#can-a-constructor-of-a-class-inside-another-constructor-be-called-in-java)
  + [40.Contiguous memory locations are usually used for storing actual values in an array but not in ArrayList. Explain.](https://www.interviewbit.com/java-interview-questions/#why-contiguous-memory-locations-store-actual-values-in-array-and-not-in-arraylist)
* [**Java Advanced Interview Questions**](https://www.interviewbit.com/java-interview-questions/#java-advanced-questions)
  + [41.Although inheritance is a popular OOPs concept, it is less advantageous than composition. Explain.](https://www.interviewbit.com/java-interview-questions/#although-inheritance-is-a-popular-oop-concept-it-is-less-advantageous-than-composition-explain)
  + [42.How is the creation of a String using new() different from that of a literal?](https://www.interviewbit.com/java-interview-questions/#string-literal-vs-new-string-object-in-java)
  + [43.Is exceeding the memory limit possible in a program despite having a garbage collector?](https://www.interviewbit.com/java-interview-questions/#possible-to-increase-the-memory-limit-despite-having-garbage-collector-in-java-program)
  + [44.Why is synchronization necessary? Explain with the help of a relevant example.](https://www.interviewbit.com/java-interview-questions/#explain-why-synchronization-is-important-in-java)
  + [45.In the given code below, what is the significance of ... ?](https://www.interviewbit.com/java-interview-questions/#what-is-the-significance-of-below-code-in-java)
  + [46.Can you explain the Java thread lifecycle?](https://www.interviewbit.com/java-interview-questions/#explain-java-thread-cycle)
  + [47.What could be the tradeoff between the usage of an unordered array versus the usage of an ordered array?](https://www.interviewbit.com/java-interview-questions/#reason-for-tradeoff-between-usage-of-unordered-array-vs-usage-of-ordered-array)
  + [48.Is it possible to import the same class or package twice in Java and what happens to it during runtime?](https://www.interviewbit.com/java-interview-questions/#can-we-import-same-class-or-package-twice-in-java-and-what-happens-during-runtime)
  + [49.In case a package has sub packages, will it suffice to import only the main package? e.g. Does importing of com.myMainPackage.\* also import com.myMainPackage.mySubPackage.\*?](https://www.interviewbit.com/java-interview-questions/#if-a-package-has-subpackages-will-it-suffice-to-import-only-the-main-package)
  + [50.Will the finally block be executed if the code System.exit(0) is written at the end of try block?](https://www.interviewbit.com/java-interview-questions/#will-finally-block-be-executed-if-code-system-exit-0-is-written-at-the-end-of-try-block)
  + [51.What do you understand by marker interfaces in Java?](https://www.interviewbit.com/java-interview-questions/#java-marker-interfaces)
  + [52.Explain the term “Double Brace Initialisation” in Java?](https://www.interviewbit.com/java-interview-questions/#explain-double-brace-initialization-in-java)
  + [53.Why is it said that the length() method of String class doesn't return accurate results?](https://www.interviewbit.com/java-interview-questions/#why-is-string-length-not-accurate)
  + [54.What is the output of the below code and why?](https://www.interviewbit.com/java-interview-questions/#explain-output-of-below-java-code)
  + [55.What are the possible ways of making object eligible for garbage collection (GC) in Java?](https://www.interviewbit.com/java-interview-questions/#how-can-we-make-objects-eligible-for-garbage-collection-in-java)
* [**Java Interview Programs**](https://www.interviewbit.com/java-interview-questions/#java-coding-interview-questions)
  + [56.Check if a given string is palindrome using recursion.](https://www.interviewbit.com/java-interview-questions/#java-program-to-check-palindrome-string-using-recursion)
  + [57.Write a Java program to check if the two strings are anagrams.](https://www.interviewbit.com/java-interview-questions/#java-program-to-check-if-two-strings-are-anagrams-or-not)
  + [58.Write a Java Program to find the factorial of a given number.](https://www.interviewbit.com/java-interview-questions/#java-program-to-find-the-factorial-of-a-number.)
  + [59.Given an array of non-duplicating numbers from 1 to n where one number is missing, write an efficient java program to find that missing number.](https://www.interviewbit.com/java-interview-questions/#java-program-to-find-missing-number-in-an-array)
  + [60.Write a Java Program to check if any number is a magic number or not. A number is said to be a magic number if after doing sum of digits in each step and inturn doing sum of digits of that sum, the ultimate result (when there is only one digit left) is 1.](https://www.interviewbit.com/java-interview-questions/#java-program-to-check-if-any-number-is-magic-number-or-not)
* [**Conclusion**](https://www.interviewbit.com/java-interview-questions/#conclusion)
  + [61.Conclusion](https://www.interviewbit.com/java-interview-questions/#conclusion)
* [**Java interview MCQs**](https://www.interviewbit.com/java-interview-questions/#java-interview-mcqs)

Do you have what it takes to ace a Java Interview? We are here to help you in consolidating your knowledge and concepts in Java. The following article will cover all the popular Java interview questions for freshers as well as experienced candidates in depth.

Go through all the questions to enhance your chances of performing well in the interviews. The questions will revolve around the basic and core fundamentals of Java.

So, let’s dive deep into the plethora of useful interview questions on Java.

**Java Basic Interview Questions**

**1. Why is Java a platform independent language?**

Java language was developed in such a way that it does not depend on any hardware or software due to the fact that the compiler compiles the code and then converts it to platform-independent byte code which can be run on multiple systems.

* The only condition to run that byte code is for the machine to have a runtime environment (JRE) installed in it.

**2. Why is Java not a pure object oriented language?**

Java supports primitive data types - byte, boolean, char, short, int, float, long, and double and hence it is not a pure object-oriented language.

**3. Pointers are used in C/ C++. Why does Java not make use of pointers?**

Pointers are quite complicated and unsafe to use by beginner programmers. Java focuses on code simplicity, and the usage of pointers can make it challenging. Pointer utilization can also cause potential errors. Moreover, security is also compromised if pointers are used because the users can directly access memory with the help of pointers.

Thus, a certain level of abstraction is furnished by not including pointers in Java. Moreover, the usage of pointers can make the procedure of garbage collection quite slow and erroneous. Java makes use of references as these cannot be manipulated, unlike pointers.

**4. What do you understand by an instance variable and a local variable?**

**Instance variables** are those variables that are accessible by all the methods in the class. They are declared outside the methods and inside the class. These variables describe the properties of an object and remain bound to it at any cost.

All the objects of the class will have their copy of the variables for utilization. If any modification is done on these variables, then only that instance will be impacted by it, and all other class instances continue to remain unaffected.

**Example:**

**class** **Athlete** {

**public** String athleteName;

**public** **double** athleteSpeed;

**public** **int** athleteAge;

}

**Local variables** are those variables present within a block, function, or constructor and can be accessed only inside them. The utilization of the variable is restricted to the block scope. Whenever a local variable is declared inside a method, the other class methods don’t have any knowledge about the local variable.

**Example:**

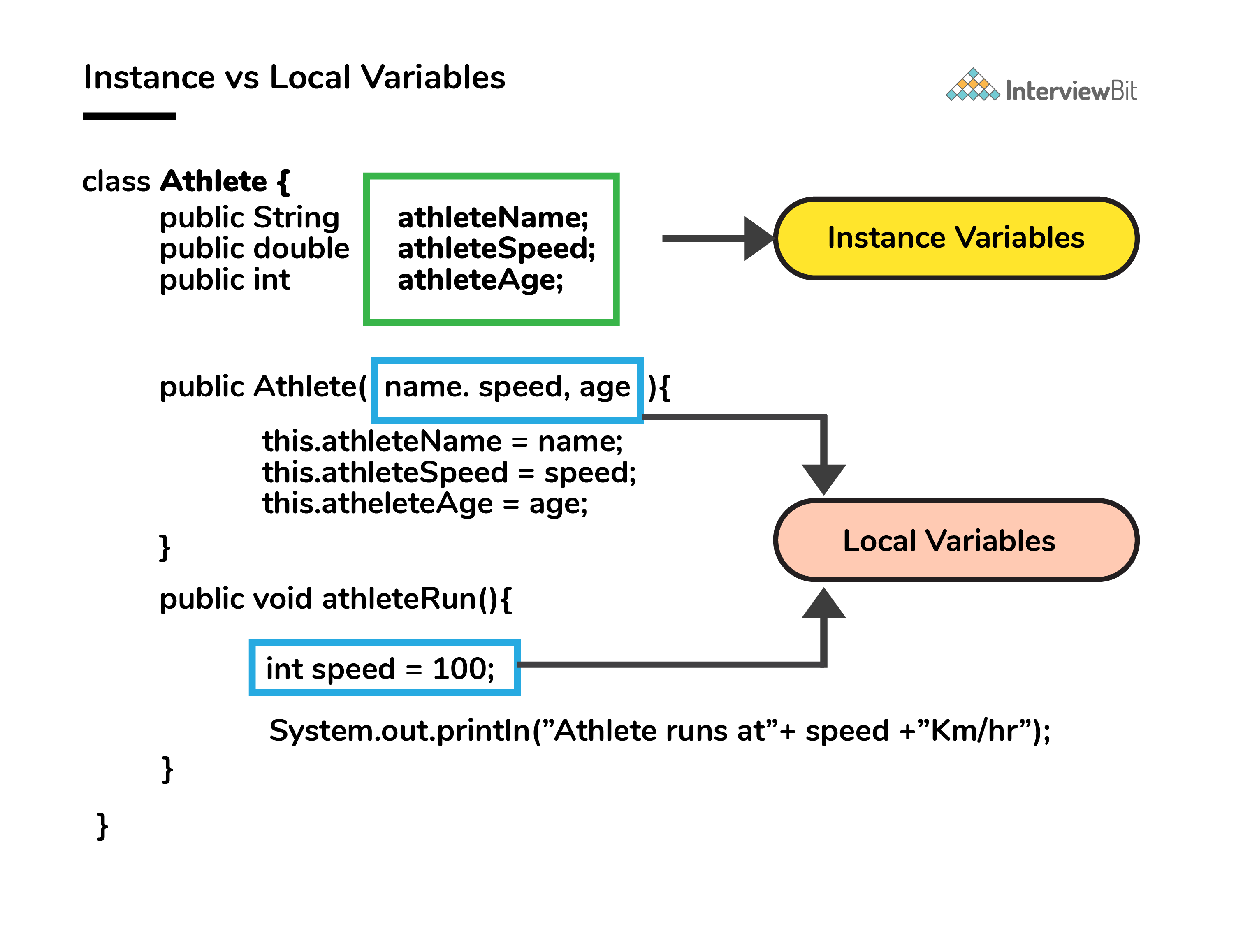
**public** **void** **athlete**() {

String athleteName;

**double** athleteSpeed;

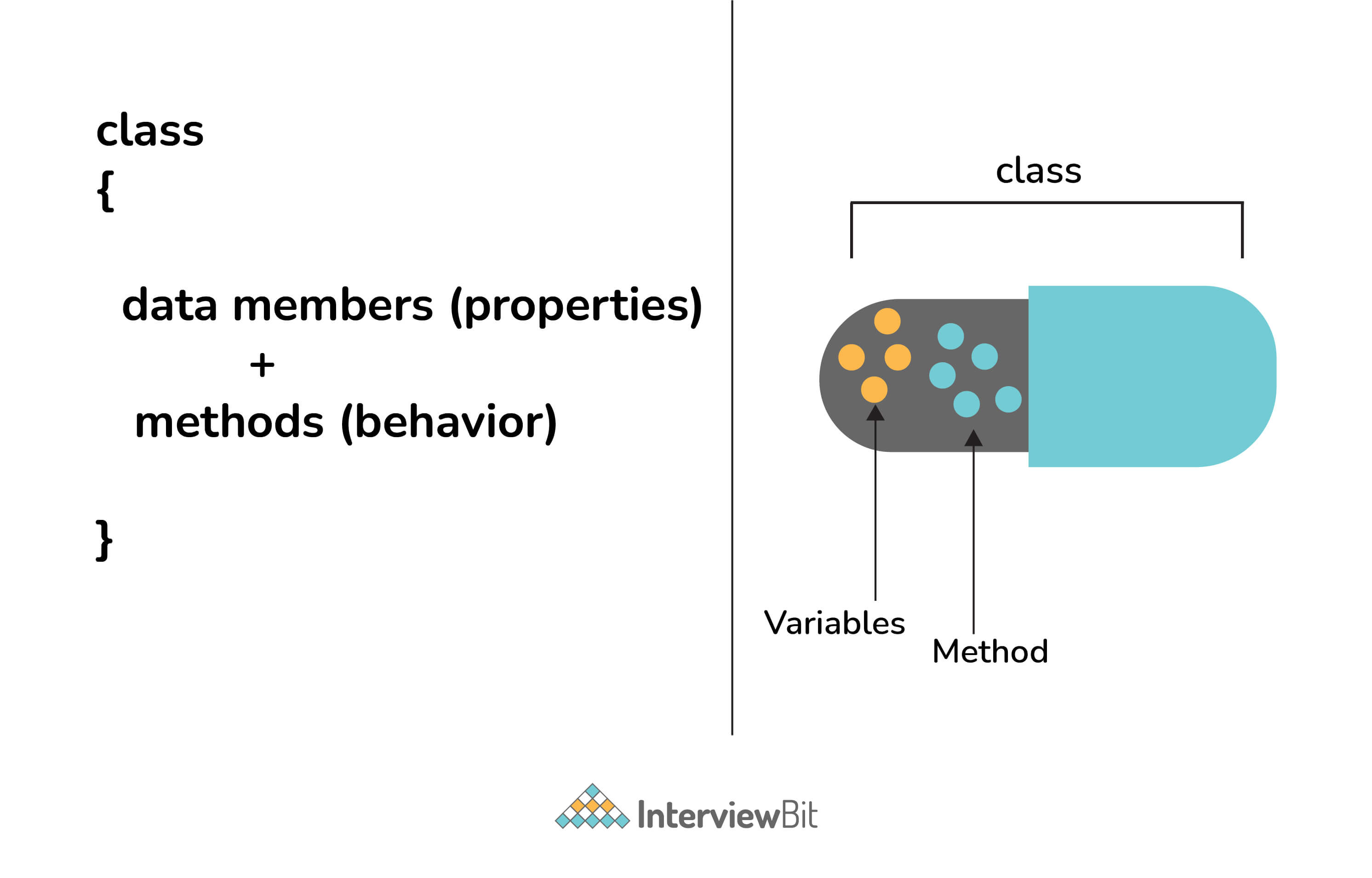
**int** athleteAge;

}



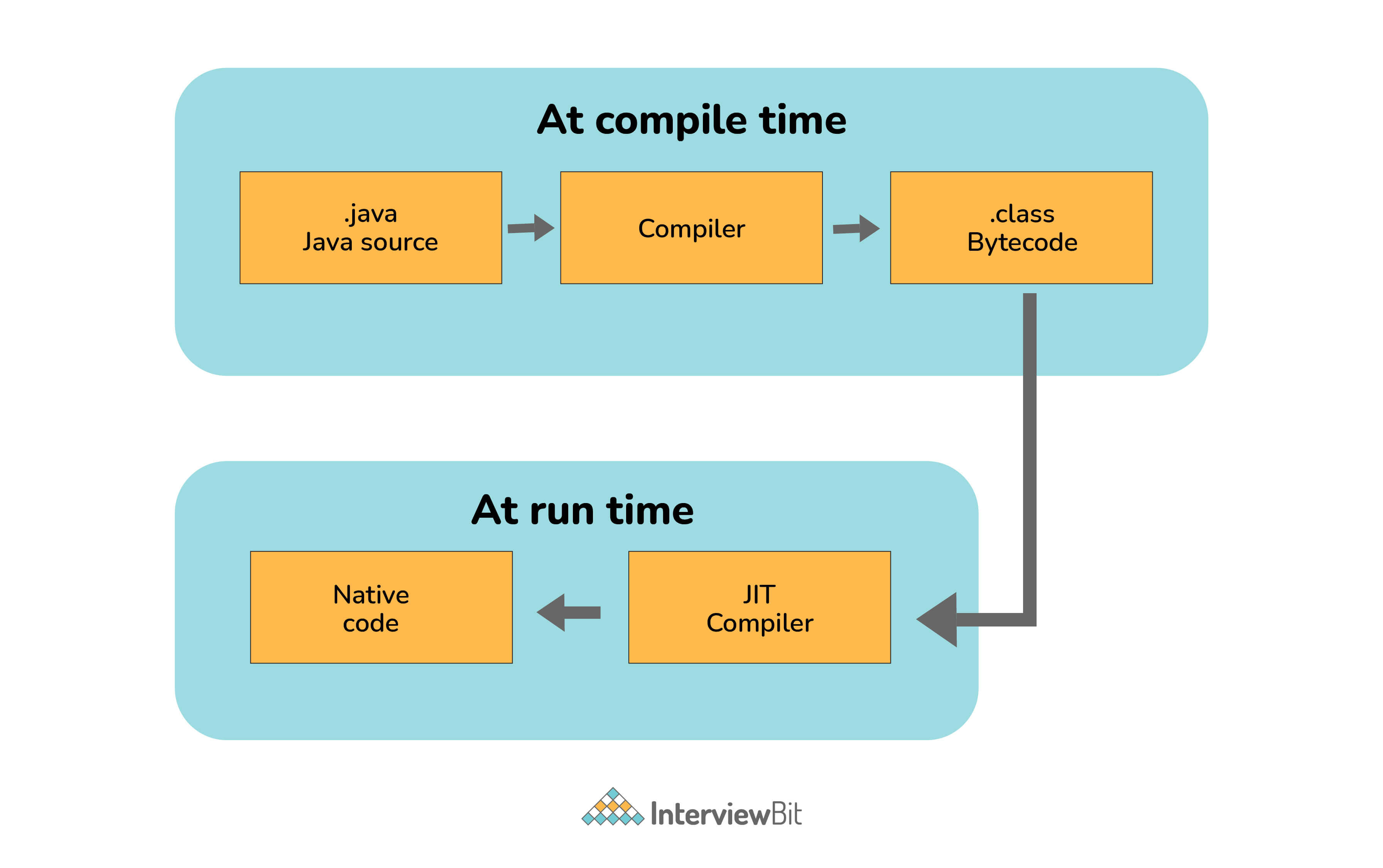
**5. What do you mean by data encapsulation?**

* Data Encapsulation is an [Object-Oriented Programming](https://www.interviewbit.com/oops-interview-questions/) concept of hiding the data attributes and their behaviors in a single unit.
* It helps developers to follow modularity while developing software by ensuring that each object is independent of other objects by having its own methods, attributes, and functionalities.
* It is used for the security of the private properties of an object and hence serves the purpose of data hiding.



**6. Tell us something about JIT compiler.**

* JIT stands for Just-In-Time and it is used for improving the performance during run time. It does the task of compiling parts of byte code having similar functionality at the same time thereby reducing the amount of compilation time for the code to run.
* The compiler is nothing but a translator of source code to machine-executable code. But what is special about the JIT compiler? Let us see how it works:
  + First, the Java source code (.java) conversion to byte code (.class) occurs with the help of the javac compiler.
  + Then, the .class files are loaded at run time by JVM and with the help of an interpreter, these are converted to machine understandable code.
  + JIT compiler is a part of JVM. When the JIT compiler is enabled, the JVM analyzes the method calls in the .class files and compiles them to get more efficient and native code. It also ensures that the prioritized method calls are optimized.
  + Once the above step is done, the JVM executes the optimized code directly instead of interpreting the code again. This increases the performance and speed of the execution.



**7. Can you tell the difference between equals() method and equality operator (==) in Java?**

| **equals()** | **==** |
| --- | --- |
| This is a method defined in the Object class. | It is a binary operator in Java. |
| This method is used for checking the equality of contents between two objects as per the specified business logic. | This operator is used for comparing addresses (or references), i.e checks if both the objects are pointing to the same memory location. |

**Note:**

* In the cases where the equals method is not overridden in a class, then the class uses the default implementation of the equals method that is closest to the parent class.
* Object class is considered as the parent class of all the java classes. The implementation of the equals method in the Object class uses the == operator to compare two objects. This default implementation can be overridden as per the business logic.

**8. How is an infinite loop declared in Java?**

Infinite loops are those loops that run infinitely without any breaking conditions. Some examples of consciously declaring infinite loop is:

* Using For Loop:

**for** (;;)

{

// Business logic

// Any break logic

}

* Using while loop:

**while**(**true**){

// Business logic

// Any break logic

}

* Using do-while loop:

**do**{

// Business logic

// Any break logic

}**while**(**true**);

**9. Briefly explain the concept of constructor overloading**

Constructor overloading is the process of creating multiple constructors in the class consisting of the same name with a difference in the constructor parameters. Depending upon the number of parameters and their corresponding types, distinguishing of the different types of constructors is done by the compiler.

**class** **Hospital** {

**int** variable1, variable2;

**double** variable3;

**public** **Hospital**(**int** doctors, **int** nurses) {

variable1 = doctors;

variable2 = nurses;

}

**public** **Hospital**(**int** doctors) {

variable1 = doctors;

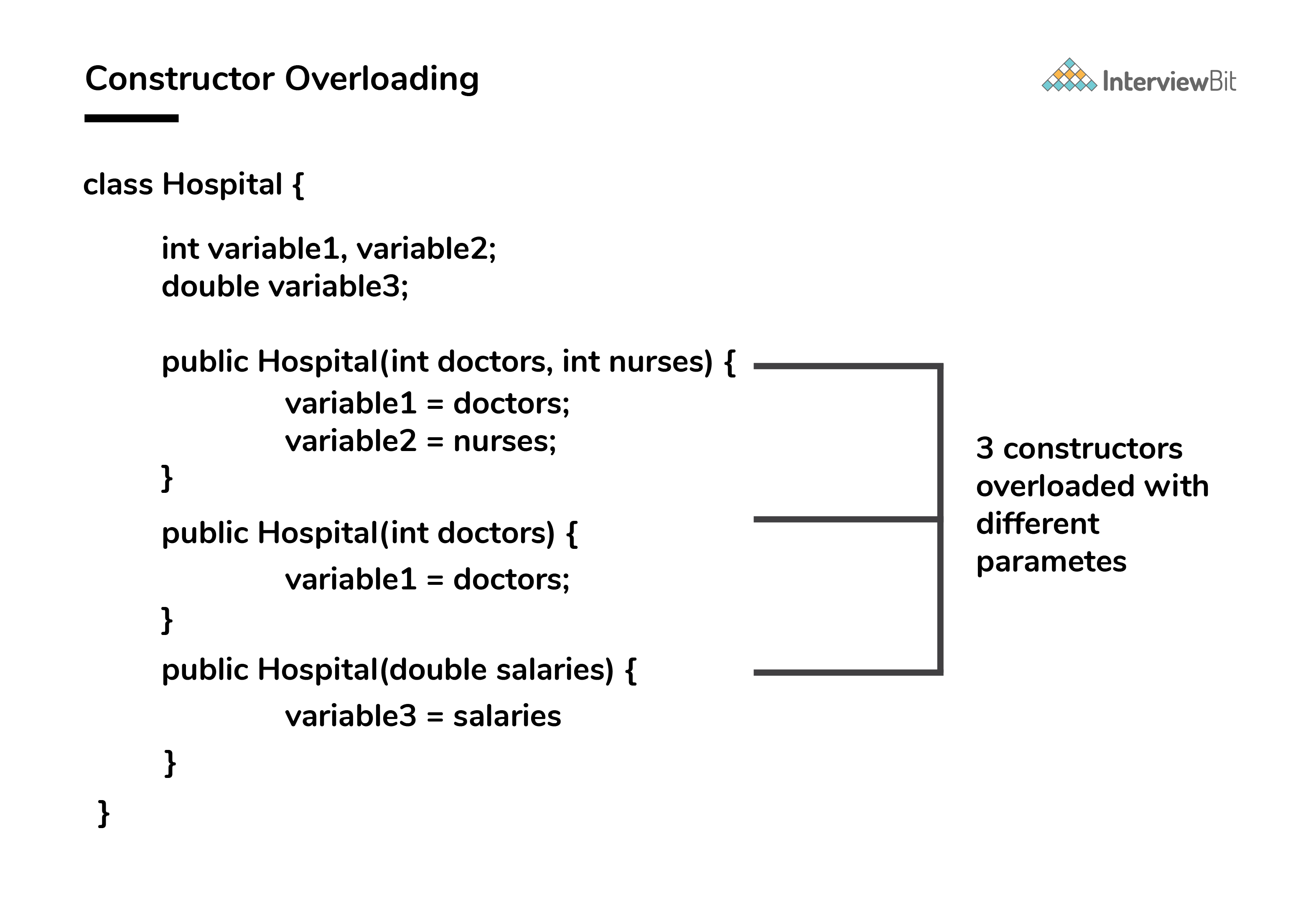
}

**public** **Hospital**(**double** salaries) {

variable3 = salaries

}

}



Three constructors are defined here but they differ on the basis of parameter type and their numbers.

**10. Comment on method overloading and overriding by citing relevant examples.**

In Java, **method overloading** is made possible by introducing different methods in the same class consisting of the same name. Still, all the functions differ in the number or type of parameters. It takes place inside a class and enhances program readability.

The only difference in the return type of the method does not promote method overloading. The following example will furnish you with a clear picture of it.

**class** **OverloadingHelp** {

**public** **int** **findarea** (**int** l, **int** b) {

**int** var1;

var1 = l \* b;

**return** var1;

}

**public** **int** **findarea** (**int** l, **int** b, **int** h) {

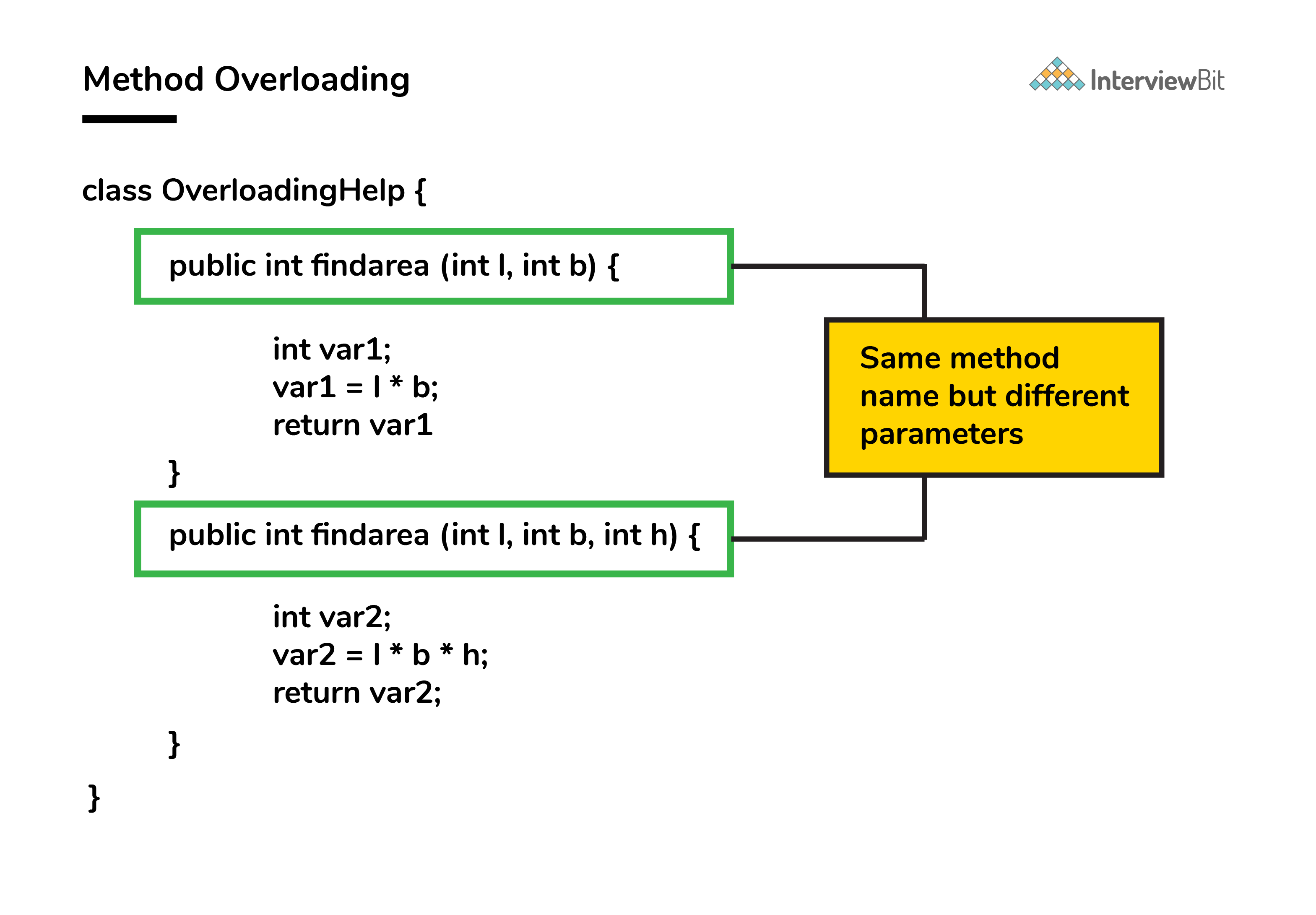
**int** var2;

var2 = l \* b \* h;

**return** var2;

}

}



Both the functions have the same name but differ in the number of arguments. The first method calculates the area of the rectangle, whereas the second method calculates the area of a cuboid.

**Method overriding** is the concept in which two methods having the same method signature are present in two different classes in which an inheritance relationship is present. A particular method implementation (already present in the base class) is possible for the derived class by using method overriding.  
Let’s give a look at this example:

**class** **HumanBeing** {

**public** **int** **walk** (**int** distance, **int** time) {

**int** speed = distance / time;

**return** speed;

}

}

**class** **Athlete** **extends** **HumanBeing** {

**public** **int** **walk**(**int** distance, **int** time) {

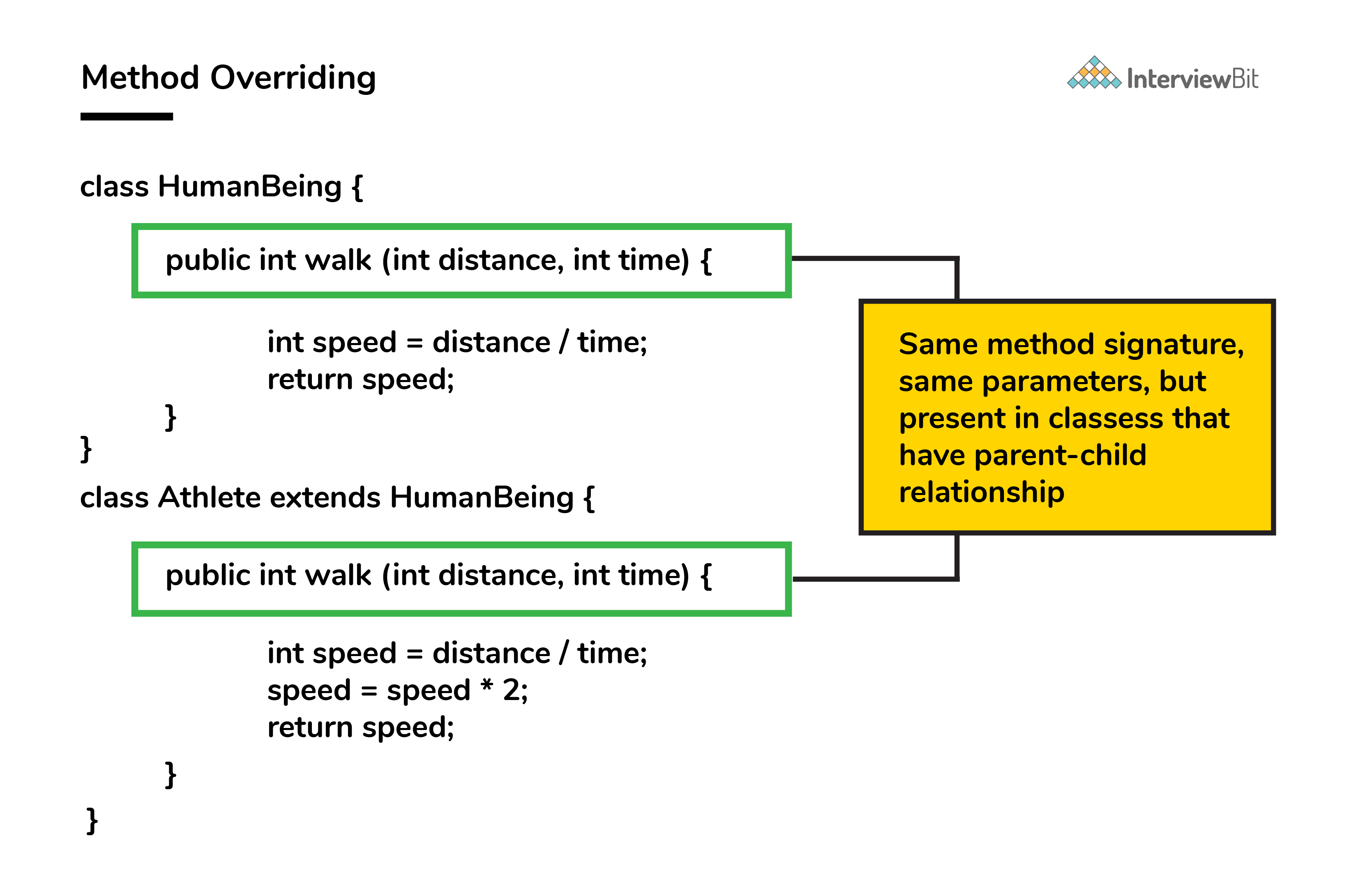
**int** speed = distance / time;

speed = speed \* 2;

**return** speed;

}

}



Both class methods have the name walk and the same parameters, distance, and time. If the derived class method is called, then the base class method walk gets overridden by that of the derived class.

**11. A single try block and multiple catch blocks can co-exist in a Java Program. Explain.**

Yes, multiple catch blocks can exist but specific approaches should come prior to the general approach because only the first catch block satisfying the catch condition is executed. The given code illustrates the same:

**public** **class** **MultipleCatch** {

**public** **static** **void** **main**(String args[]) {

**try** {

**int** n = 1000, x = 0;

**int** arr[] = **new** **int**[n];

**for** (**int** i = 0; i <= n; i++) {

arr[i] = i / x;

}

}

**catch** (ArrayIndexOutOfBoundsException exception) {

System.out.println("1st block = ArrayIndexOutOfBoundsException");

}

**catch** (ArithmeticException exception) {

System.out.println("2nd block = ArithmeticException");

}

**catch** (Exception exception) {

System.out.println("3rd block = Exception");

}

}

}

Here, the second catch block will be executed because of division by 0 (i / x). In case x was greater than 0 then the first catch block will execute because for loop runs till i = n and array index are till n-1.

**12. Explain the use of final keyword in variable, method and class.**

In Java, the final keyword is used as defining something as constant /final and represents the non-access modifier.

* **final variable:**
  + When a variable is declared as final in Java, the value can’t be modified once it has been assigned.
  + If any value has not been assigned to that variable, then it can be assigned only by the constructor of the class.
* **final method:**
  + A method declared as final cannot be overridden by its children's classes.
  + A constructor cannot be marked as final because whenever a class is inherited, the constructors are not inherited. Hence, marking it final doesn't make sense. Java throws compilation error saying - modifier final not allowed here
* **final class:**
  + No classes can be inherited from the class declared as final. But that final class can extend other classes for its usage.

**13. Do final, finally and finalize keywords have the same function?**

All three keywords have their own utility while programming.

**Final:** If any restriction is required for classes, variables, or methods, the final keyword comes in handy. Inheritance of a final class and overriding of a final method is restricted by the use of the final keyword. The variable value becomes fixed after incorporating the final keyword. Example:

**final** **int** a=100;

a = 0; // error

The second statement will throw an error.

**Finally:** It is the block present in a program where all the codes written inside it get executed irrespective of handling of exceptions. Example:

**try** {

**int** variable = 5;

}

**catch** (Exception exception) {

System.out.println("Exception occurred");

}

**finally** {

System.out.println("Execution of finally block");

}

**Finalize:** Prior to the garbage collection of an object, the finalize method is called so that the clean-up activity is implemented. Example:

**public** **static** **void** **main**(String[] args) {

String example = **new** String("InterviewBit");

example = **null**;

System.gc(); // Garbage collector called

}

**public** **void** **finalize**() {

// Finalize called

}

**14. When can you use super keyword?**

* The super keyword is used to access hidden fields and overridden methods or attributes of the parent class.
* Following are the cases when this keyword can be used:
  + Accessing data members of parent class when the member names of the class and its child subclasses are same.
  + To call the default and parameterized constructor of the parent class inside the child class.
  + Accessing the parent class methods when the child classes have overridden them.
* The following example demonstrates all 3 cases when a super keyword is used.

**public** **class** **Parent**{

**private** **int** num = 1;

Parent(){

System.out.println("Parent class default constructor.");

}

Parent(String x){

System.out.println("Parent class parameterised constructor.");

}

**public** **void** **foo**(){

System.out.println("Parent class foo!");

}

}

**public** **class** **Child** **extends** **Parent**{

**private** **int** num = 2;

Child(){

System.out.println("Child class default Constructor");

**super**(); // to call default parent constructor

**super**("Call Parent"); // to call parameterised constructor.

}

**void** **printNum**(){

System.out.println(num);

System.out.println(**super**.num); //prints the value of num of parent class

}

@Override

**public** **void** **foo**(){

System.out.println("Parent class foo!");

**super**.foo(); //Calls foo method of Parent class inside the Overriden foo method of Child class.

}

}

**15. Can the static methods be overloaded?**

Yes! There can be two or more static methods in a class with the same name but differing input parameters.

**16. Can the static methods be overridden?**

* No! Declaration of static methods having the same signature can be done in the subclass but run time polymorphism can not take place in such cases.
* Overriding or dynamic polymorphism occurs during the runtime, but the static methods are loaded and looked up at the compile time statically. Hence, these methods cant be overridden.

**17. What is the main objective of garbage collection?**

The main objective of this process is to free up the memory space occupied by the unnecessary and unreachable objects during the Java program execution by deleting those unreachable objects.

* This ensures that the memory resource is used efficiently, but it provides no guarantee that there would be sufficient memory for the program execution.

**18. What part of memory - Stack or Heap - is cleaned in garbage collection process?**

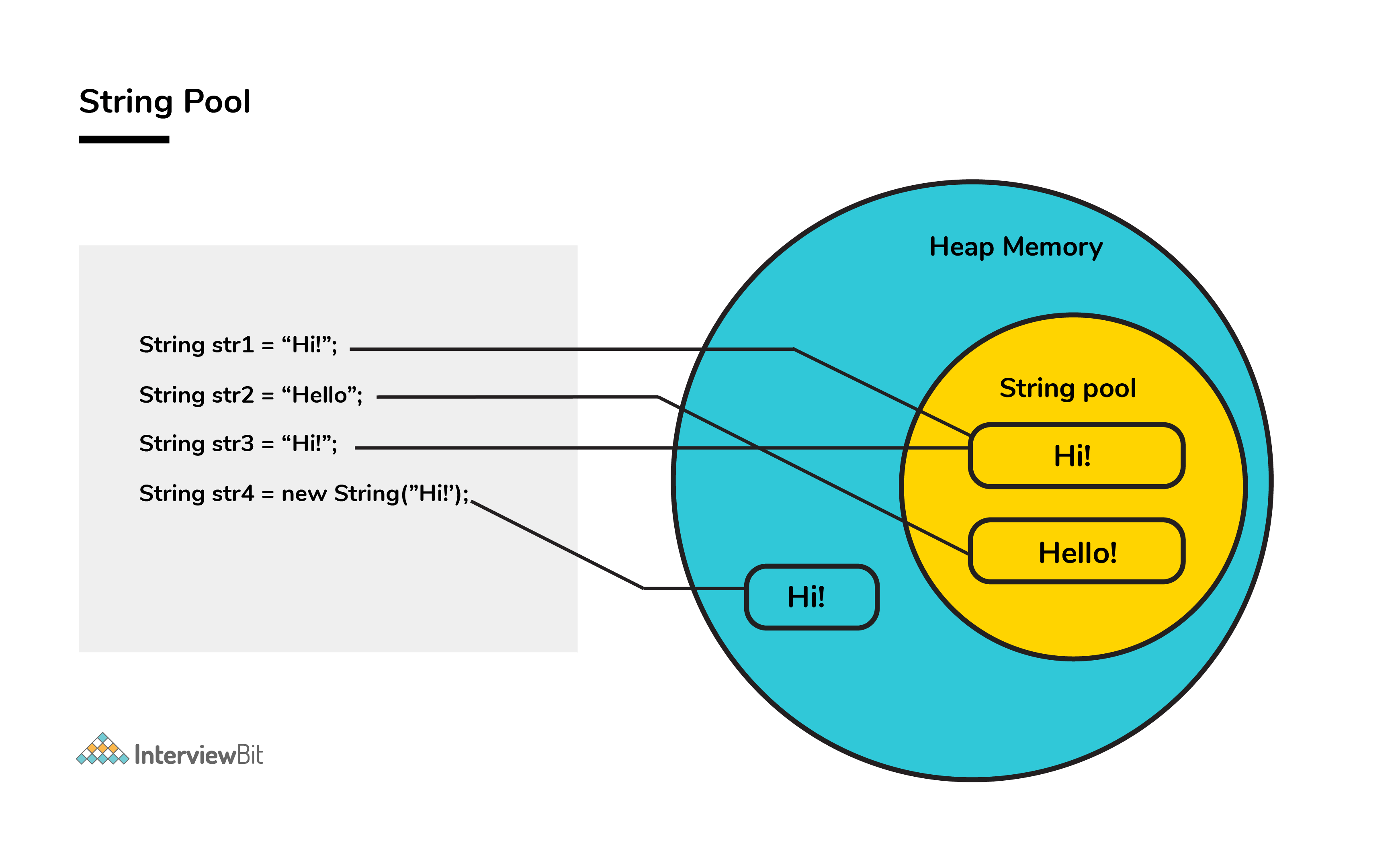
Heap.

**Java Intermediate Interview Questions**

**19. Apart from the security aspect, what are the reasons behind making strings immutable in Java?**

A String is made immutable due to the following reasons:

* **String Pool:** Designers of Java were aware of the fact that String data type is going to be majorly used by the programmers and developers. Thus, they wanted optimization from the beginning. They came up with the notion of using the String pool (a storage area in Java heap) to store the String literals. They intended to decrease the temporary String object with the help of sharing. An immutable class is needed to facilitate sharing. The sharing of the mutable structures between two unknown parties is not possible. Thus, immutable Java String helps in executing the concept of String Pool.



* **Multithreading:** The safety of threads regarding the String objects is an important aspect in Java. No external synchronization is required if the String objects are immutable. Thus, a cleaner code can be written for sharing the String objects across different threads. The complex process of concurrency is facilitated by this method.
* **Collections:** In the case of Hashtables and HashMaps, keys are String objects. If the String objects are not immutable, then it can get modified during the period when it resides in the HashMaps. Consequently, the retrieval of the desired data is not possible. Such changing states pose a lot of risks. Therefore, it is quite safe to make the string immutable.

**20. How would you differentiate between a String, StringBuffer, and a StringBuilder?**

* **Storage area:** In string, the String pool serves as the storage area. For StringBuilder and StringBuffer, heap memory is the storage area.
* **Mutability:** A String is immutable, whereas both the StringBuilder and StringBuffer are mutable.
* **Efficiency:** It is quite slow to work with a String. However, StringBuilder is the fastest in performing operations. The speed of a StringBuffer is more than a String and less than a StringBuilder. (For example appending a character is fastest in StringBuilder and very slow in String because a new memory is required for the new String with appended character.)
* **Thread-safe:** In the case of a threaded environment, StringBuilder and StringBuffer are used whereas a String is not used. However, StringBuilder is suitable for an environment with a single thread, and a StringBuffer is suitable for multiple threads.  
  **Syntax:**

// String

String first = "InterviewBit";

String second = **new** String("InterviewBit");

// StringBuffer

StringBuffer third = **new** StringBuffer("InterviewBit");

// StringBuilder

StringBuilder fourth = **new** StringBuilder("InterviewBit");

**21. Using relevant properties highlight the differences between interfaces and abstract classes.**

* **Availability of methods:** Only abstract methods are available in interfaces, whereas non-abstract methods can be present along with abstract methods in abstract classes.
* **Variable types**: Static and final variables can only be declared in the case of interfaces, whereas abstract classes can also have non-static and non-final variables.
* **Inheritance:**Multiple inheritances are facilitated by interfaces, whereas abstract classes do not promote multiple inheritances.
* **Data member accessibility:** By default, the class data members of interfaces are of the public- type. Conversely, the class members for an abstract class can be protected or private also.
* **Implementation:** With the help of an abstract class, the implementation of an interface is easily possible. However, the converse is not true;

**Abstract class example:**

**public** **abstract** **class** **Athlete** {

**public** **abstract** **void** **walk**();

}

**Interface example:**

**public** **interface** **Walkable** {

**void** **walk**();

}

**22. In Java, static as well as private method overriding is possible. Comment on the statement.**

The statement in the context is completely False. The static methods have no relevance with the objects, and these methods are of the class level. In the case of a child class, a static method with a method signature exactly like that of the parent class can exist without even throwing any compilation error.

The phenomenon mentioned here is popularly known as method hiding, and overriding is certainly not possible. Private method overriding is unimaginable because the visibility of the private method is restricted to the parent class only. As a result, only hiding can be facilitated and not overriding.

**23. What makes a HashSet different from a TreeSet?**

Although both HashSet and TreeSet are not synchronized and ensure that duplicates are not present, there are certain properties that distinguish a HashSet from a TreeSet.

* **Implementation:** For a HashSet, the hash table is utilized for storing the elements in an unordered manner. However, TreeSet makes use of the red-black tree to store the elements in a sorted manner.
* **Complexity/ Performance:** For adding, retrieving, and deleting elements, the time amortized complexity is O(1) for a HashSet. The time complexity for performing the same operations is a bit higher for TreeSet and is equal to O(log n). Overall, the performance of HashSet is faster in comparison to TreeSet.
* **Methods:**hashCode() and equals() are the methods utilized by HashSet for making comparisons between the objects. Conversely, compareTo() and compare() methods are utilized by TreeSet to facilitate object comparisons.
* **Objects type:** Heterogeneous and null objects can be stored with the help of HashSet. In the case of a TreeSet, runtime exception occurs while inserting heterogeneous objects or null objects.

**24. Why is the character array preferred over string for storing confidential information?**

In Java, a string is basically immutable i.e. it cannot be modified. After its declaration, it continues to stay in the string pool as long as it is not removed in the form of garbage. In other words, a string resides in the heap section of the memory for an unregulated and unspecified time interval after string value processing is executed.

As a result, vital information can be stolen for pursuing harmful activities by hackers if a memory dump is illegally accessed by them. Such risks can be eliminated by using mutable objects or structures like character arrays for storing any variable. After the work of the character array variable is done, the variable can be configured to blank at the same instant. Consequently, it helps in saving heap memory and also gives no chance to the hackers to extract vital data.

**25. What are the differences between JVM, JRE and JDK in Java?**

| **Criteria** | **JDK** | **JRE** | **JVM** |
| --- | --- | --- | --- |
| **Abbreviation** | Java Development Kit | Java Runtime Environment | Java Virtual Machine |
| **Definition** | JDK is a complete software development kit for developing Java applications. It comprises JRE, JavaDoc, compiler, debuggers, etc. | JRE is a software package providing Java class libraries, JVM and all the required components to run the Java applications. | JVM is a platform-dependent, abstract machine comprising of 3 specifications - document describing the JVM implementation requirements, computer program meeting the JVM requirements and instance object for executing the Java byte code and provide the runtime environment for execution. |
| **Main Purpose** | JDK is mainly used for code development and execution. | JRE is mainly used for environment creation to execute the code. | JVM provides specifications for all the implementations to JRE. |
| **Tools provided** | JDK provides tools like compiler, debuggers, etc for code development | JRE provides libraries and classes required by JVM to run the program. | JVM does not include any tools, but instead, it provides the specification for implementation. |
| **Summary** | JDK = (JRE) + Development tools | JRE = (JVM) + Libraries to execute the application | JVM = Runtime environment to execute Java byte code. |

**26. What are the differences between HashMap and HashTable in Java?**

| **HashMap** | **HashTable** |
| --- | --- |
| HashMap is not synchronized thereby making it better for non-threaded applications. | HashTable is synchronized and hence it is suitable for threaded applications. |
| Allows only one null key but any number of null in the values. | This does not allow null in both keys or values. |
| Supports order of insertion by making use of its subclass LinkedHashMap. | Order of insertion is not guaranteed in HashTable. |

**27. What is the importance of reflection in Java?**

* The term reflection is used for describing the inspection capability of a code on other code either of itself or of its system and modify it during runtime.
* Consider an example where we have an object of unknown type and we have a method ‘fooBar()’ which we need to call on the object. The static typing system of Java doesn't allow this method invocation unless the type of the object is known beforehand. This can be achieved using reflection which allows the code to scan the object and identify if it has any method called “fooBar()” and only then call the method if needed.

Method methodOfFoo = fooObject.getClass().getMethod("fooBar", **null**);

methodOfFoo.invoke(fooObject, **null**);

* Using reflection has its own cons:
  + Speed — Method invocations due to reflection are about three times slower than the direct method calls.
  + Type safety — When a method is invoked via its reference wrongly using reflection, invocation fails at runtime as it is not detected at compile/load time.
  + Traceability — Whenever a reflective method fails, it is very difficult to find the root cause of this failure due to a huge stack trace. One has to deep dive into the invoke() and proxy() method logs to identify the root cause.
* Hence, it is advisable to follow solutions that don't involve reflection and use this method as a last resort.

**28. What are the different ways of threads usage?**

* We can define and implement a thread in java using two ways:
  + **Extending the Thread class**

**class** **InterviewBitThreadExample** **extends** **Thread**{

**public** **void** **run**(){

System.out.println("Thread runs...");

}

**public** **static** **void** **main**(String args[]){

InterviewBitThreadExample ib = **new** InterviewBitThreadExample();

ib.start();

}

}

* **Implementing the Runnable interface**

**class** **InterviewBitThreadExample** **implements** **Runnable**{

**public** **void** **run**(){

System.out.println("Thread runs...");

}

**public** **static** **void** **main**(String args[]){

Thread ib = **new** Thread(**new** InterviewBitThreadExample());

ib.start();

}

}

* Implementing a thread using the method of Runnable interface is more preferred and advantageous as Java does not have support for multiple inheritances of classes.
* start() method is used for creating a separate call stack for the thread execution. Once the call stack is created, JVM calls the run() method for executing the thread in that call stack.

**29. What are the differences between constructor and method of a class in Java?**

| **Constructor** | **Method** |
| --- | --- |
| Constructor is used for initializing the object state. | Method is used for exposing the object's behavior. |
| Constructor has no return type. | Method should have a return type. Even if it does not return anything, return type is void. |
| Constructor gets invoked implicitly. | Method has to be invoked on the object explicitly. |
| If the constructor is not defined, then a default constructor is provided by the java compiler. | If a method is not defined, then the compiler does not provide it. |
| The constructor name should be equal to the class name. | The name of the method can have any name or have a class name too. |
| A constructor cannot be marked as final because whenever a class is inherited, the constructors are not inherited. Hence, marking it final doesn't make sense. Java throws compilation error saying - modifier final not allowed here | A method can be defined as final but it cannot be overridden in its subclasses. |
| Final variable instantiations are possible inside a constructor and the scope of this applies to the whole class and its objects. | A final variable if initialised inside a method ensures that the variable cant be changed only within the scope of that method. |

**30. Java works as “pass by value” or “pass by reference” phenomenon?**

Java always works as a “pass by value”. There is nothing called a “pass by reference” in Java. However, when the object is passed in any method, the address of the value is passed due to the nature of object handling in Java. When an object is passed, a copy of the reference is created by Java and that is passed to the method. The objects point to the same memory location. 2 cases might happen inside the method:

* **Case 1:**When the object is pointed to another location: In this case, the changes made to that object do not get reflected the original object before it was passed to the method as the reference points to another location.

For example:

**class** **InterviewBitTest**{

**int** num;

InterviewBitTest(**int** x){

num = x;

}

InterviewBitTest(){

num = 0;

}

}

**class** **Driver** {

**public** **static** **void** **main**(String[] args)

{

//create a reference

InterviewBitTest ibTestObj = **new** InterviewBitTest(20);

//Pass the reference to updateObject Method

updateObject(ibTestObj);

//After the updateObject is executed, check for the value of num in the object.

System.out.println(ibTestObj.num);

}

**public** **static** **void** **updateObject**(InterviewBitTest ibObj)

{

// Point the object to new reference

ibObj = **new** InterviewBitTest();

// Update the value

ibObj.num = 50;

}

}

Output:

20

* **Case 2:** When object references are not modified: In this case, since we have the copy of reference the main object pointing to the same memory location, any changes in the content of the object get reflected in the original object.

For example:

**class** **InterviewBitTest**{

**int** num;

InterviewBitTest(**int** x){

num = x;

}

InterviewBitTest(){

num = 0;

}

}

**class** **Driver**{

**public** **static** **void** **main**(String[] args)

{

//create a reference

InterviewBitTest ibTestObj = **new** InterviewBitTest(20);

//Pass the reference to updateObject Method

updateObject(ibTestObj);

//After the updateObject is executed, check for the value of num in the object.

System.out.println(ibTestObj.num);

}

**public** **static** **void** **updateObject**(InterviewBitTest ibObj)

{

// no changes are made to point the ibObj to new location

// Update the value of num

ibObj.num = 50;

}

}

Output:

50

**31. Which among String or String Buffer should be preferred when there are lot of updates required to be done in the data?**

StringBuffer is mutable and dynamic in nature whereas String is immutable. Every updation / modification of String creates a new String thereby overloading the string pool with unnecessary objects. Hence, in the cases of a lot of updates, it is always preferred to use StringBuffer as it will reduce the overhead of the creation of multiple String objects in the string pool.

**32. How to not allow serialization of attributes of a class in Java?**

* In order to achieve this, the attribute can be declared along with the usage of transient keyword as shown below:

**public** **class** **InterviewBitExample** {

**private** **transient** String someInfo;

**private** String name;

**private** **int** id;

// :

// Getters setters

// :

}

* In the above example, all the fields except someInfo can be serialized.

**33. What happens if the static modifier is not included in the main method signature in Java?**

There wouldn't be any compilation error. But then the program is run, since the JVM cant map the main method signature, the code throws “NoSuchMethodError” error at the runtime.

**34. What happens if there are multiple main methods inside one class in Java?**

The program can't compile as the compiler says that the method has been already defined inside the class.

**35. What do you understand by Object Cloning and how do you achieve it in Java?**

* It is the process of creating an exact copy of any object. In order to support this, a java class has to implement the Cloneable interface of java.lang package and override the clone() method provided by the Object class the syntax of which is:

**protected** Object **clone**() **throws** CloneNotSupportedException{

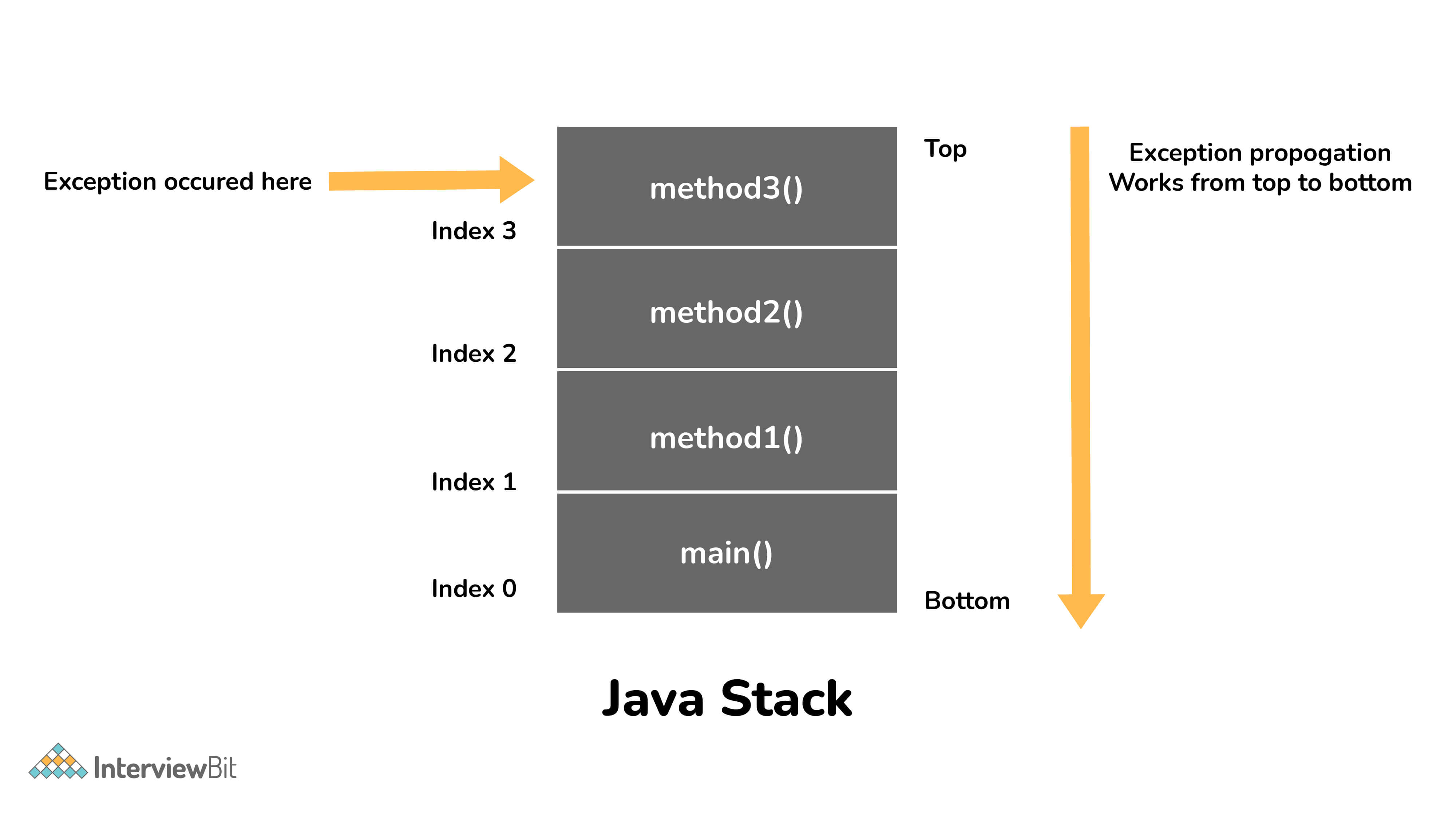
**return** (Object)**super**.clone();

}

* In case the Cloneable interface is not implemented and just the method is overridden, it results in CloneNotSupportedException in Java.

**36. How does an exception propagate in the code?**

When an exception occurs, first it searches to locate the matching catch block. In case, the matching catch block is located, then that block would be executed. Else, the exception propagates through the method call stack and goes into the caller method where the process of matching the catch block is performed. This propagation happens until the matching catch block is found. If the match is not found, then the program gets terminated in the main method.



**37. Is it mandatory for a catch block to be followed after a try block?**

No, it is not necessary for a catch block to be present after a try block. - A try block should be followed either by a catch block or by a finally block. If the exceptions likelihood is more, then they should be declared using the throws clause of the method.

**38. Will the finally block get executed when the return statement is written at the end of try block and catch block as shown below?**

**public** **int** **someMethod**(**int** i){

**try**{

//some statement

**return** 1;

}**catch**(Exception e){

//some statement

**return** 999;

}**finally**{

//finally block statements

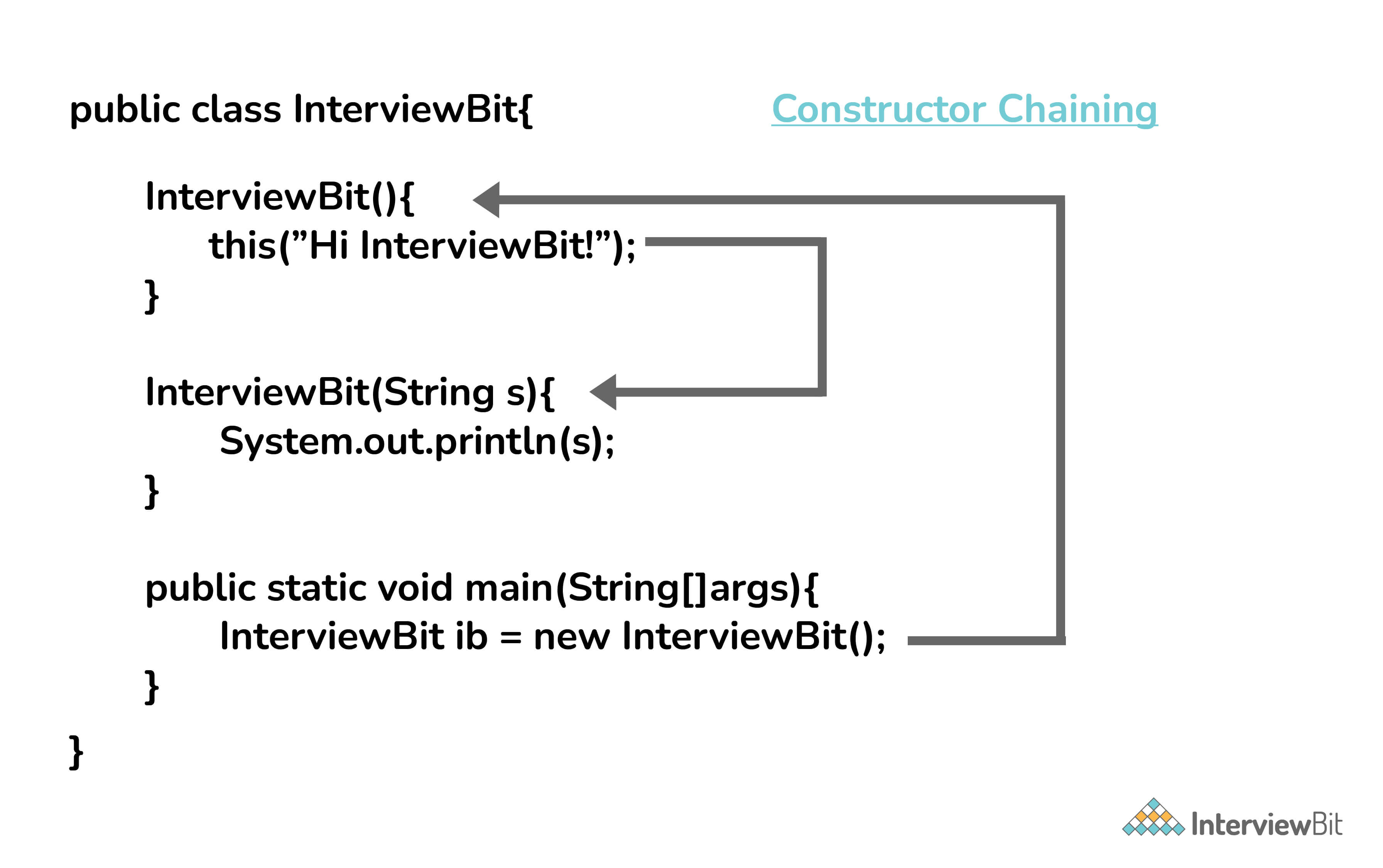
}

}

finally block will be executed irrespective of the exception or not. The only case where finally block is not executed is when it encounters ‘System.exit()’ method anywhere in try/catch block.

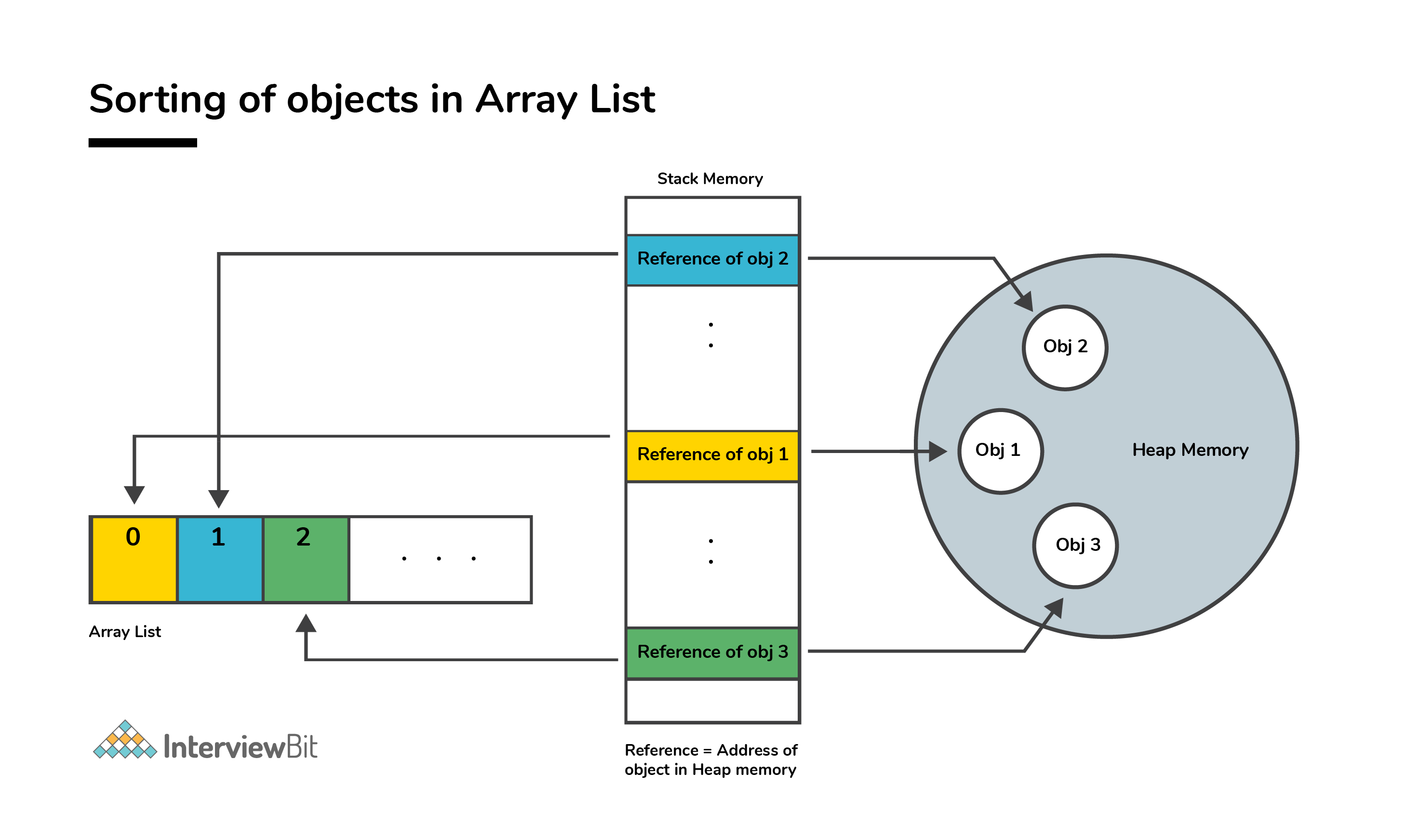
**39. Can you call a constructor of a class inside the another constructor?**

Yes, the concept can be termed as constructor chaining and can be achieved using this().



**40. Contiguous memory locations are usually used for storing actual values in an array but not in ArrayList. Explain.**

In the case of ArrayList, data storing in the form of primitive data types (like int, float, etc.) is not possible. The data members/objects present in the ArrayList have references to the objects which are located at various sites in the memory. Thus, storing of actual objects or non-primitive data types (like Integer, Double, etc.) takes place in various memory locations.



However, the same does not apply to the arrays. Object or primitive type values can be stored in arrays in contiguous memory locations, hence every element does not require any reference to the next element.



**Java Advanced Interview Questions**

**41. Although inheritance is a popular OOPs concept, it is less advantageous than composition. Explain.**

Inheritance lags behind composition in the following scenarios:

* Multiple-inheritance is not possible in Java. Classes can only extend from one superclass. In cases where multiple functionalities are required, for example - to read and write information into the file, the pattern of composition is preferred. The writer, as well as reader functionalities, can be made use of by considering them as the private members.
* Composition assists in attaining high flexibility and prevents breaking of encapsulation.
* Unit testing is possible with composition and not inheritance. When a developer wants to test a class composing a different class, then Mock Object can be created for signifying the composed class to facilitate testing. This technique is not possible with the help of inheritance as the derived class cannot be tested without the help of the superclass in inheritance.
* The loosely coupled nature of composition is preferable over the tightly coupled nature of inheritance.

Let’s take an example:

**package** comparison;

**public** **class** **Top** {

**public** **int** **start**() {

**return** 0;

}

}

**class** **Bottom** **extends** **Top** {

**public** **int** **stop**() {

**return** 0;

}

}

In the above example, inheritance is followed. Now, some modifications are done to the Top class like this:

**public** **class** **Top** {

**public** **int** **start**() {

**return** 0;

}

**public** **void** **stop**() {

}

}

If the new implementation of the Top class is followed, a compile-time error is bound to occur in the Bottom class. Incompatible return type is there for the Top.stop() function. Changes have to be made to either the Top or the Bottom class to ensure compatibility. However, the composition technique can be utilized to solve the given problem:

**class** **Bottom** {

Top par = **new** Top();

**public** **int** **stop**() {

par.start();

par.stop();

**return** 0;

}

}

**42. How is the creation of a String using new() different from that of a literal?**

When a String is formed as a literal with the assistance of an assignment operator, it makes its way into the String constant pool so that String Interning can take place. This same object in the heap will be referenced by a different String if the content is the same for both of them.

**public** bool **checking**() {

String first = "InterviewBit";

String second = "InterviewBit";

**if** (first == second)

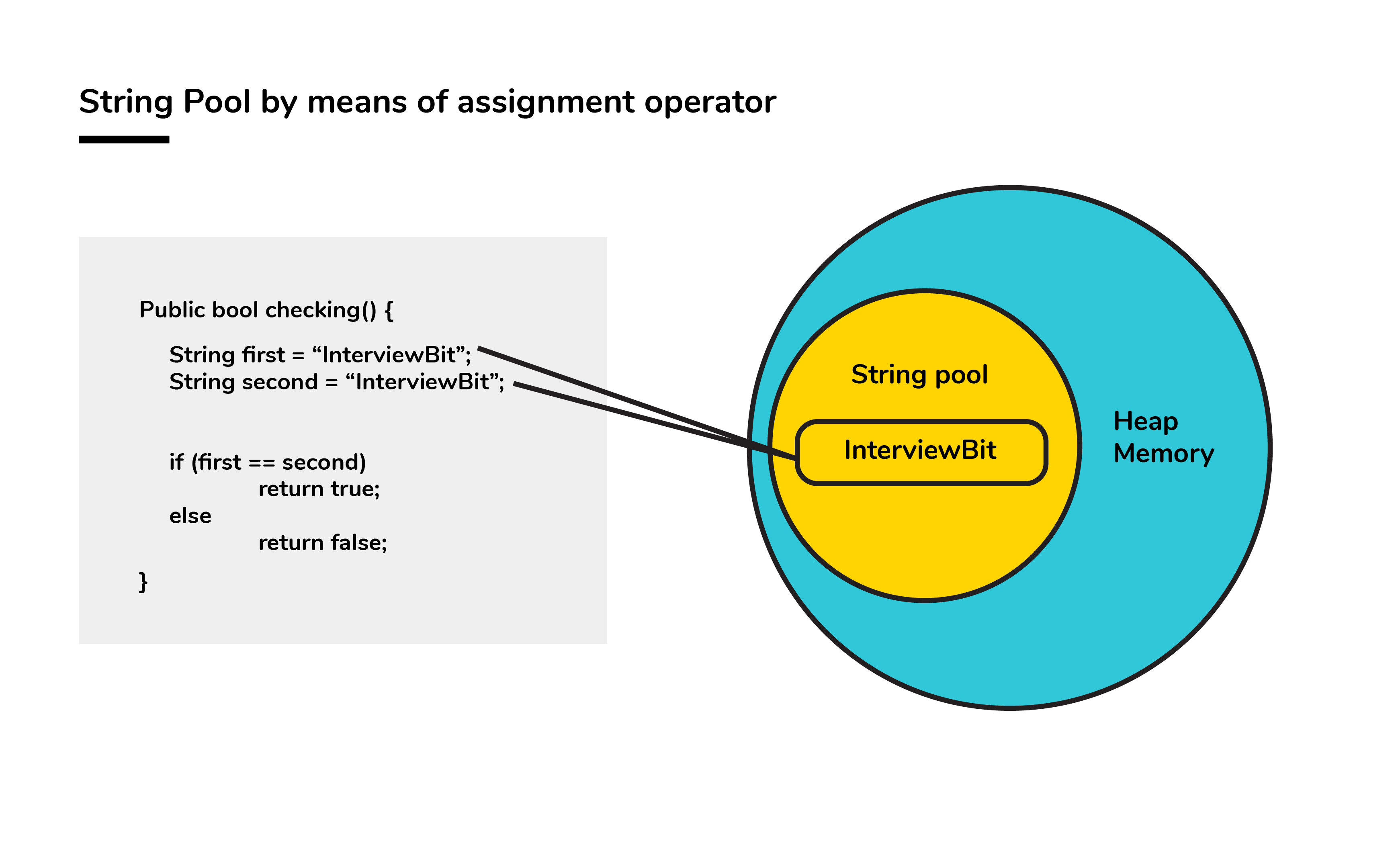
**return** **true**;

**else**

**return** **false**;

}

The checking() function will return true as the same content is referenced by both the variables.



Conversely, when a String formation takes place with the help of a new() operator, interning does not take place. The object gets created in the heap memory even if the same content object is present.

**public** bool **checking**() {

String first = **new** String("InterviewBit");

String second = **new** String("InterviewBit");

**if** (first == second)

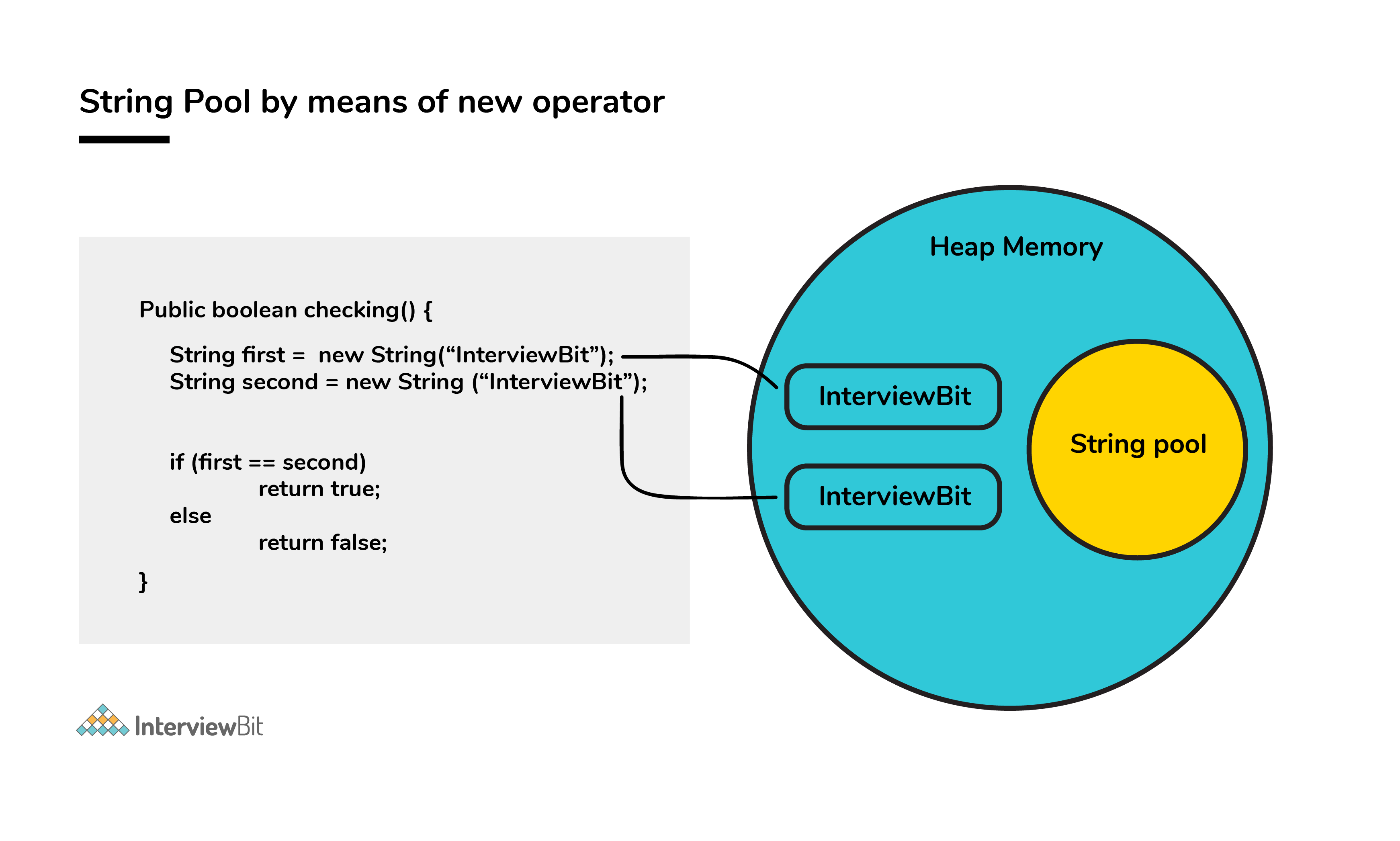
**return** **true**;

**else**

**return** **false**;

}

The checking() function will return false as the same content is not referenced by both the variables.



**43. Is exceeding the memory limit possible in a program despite having a garbage collector?**

Yes, it is possible for the program to go out of memory in spite of the presence of a garbage collector. Garbage collection assists in recognizing and eliminating those objects which are not required in the program anymore, in order to free up the resources used by them.

In a program, if an object is unreachable, then the execution of garbage collection takes place with respect to that object. If the amount of memory required for creating a new object is not sufficient, then memory is released for those objects which are no longer in the scope with the help of a garbage collector. The memory limit is exceeded for the program when the memory released is not enough for creating new objects.

Moreover, exhaustion of the heap memory takes place if objects are created in such a manner that they remain in the scope and consume memory. The developer should make sure to dereference the object after its work is accomplished. Although the garbage collector endeavors its level best to reclaim memory as much as possible, memory limits can still be exceeded.

Let’s take a look at the following example:

List<String> example = **new** LinkedList<String>();

**while**(**true**){

example.add(**new** String("Memory Limit Exceeded"));

}

**44. Why is synchronization necessary? Explain with the help of a relevant example.**

Concurrent execution of different processes is made possible by synchronization. When a particular resource is shared between many threads, situations may arise in which multiple threads require the same shared resource.

Synchronization assists in resolving the issue and the resource is shared by a single thread at a time. Let’s take an example to understand it more clearly. For example, you have a URL and you have to find out the number of requests made to it. Two simultaneous requests can make the count erratic.

**No synchronization:**

**package** anonymous;

**public** **class** **Counting** {

**private** **int** increase\_counter;

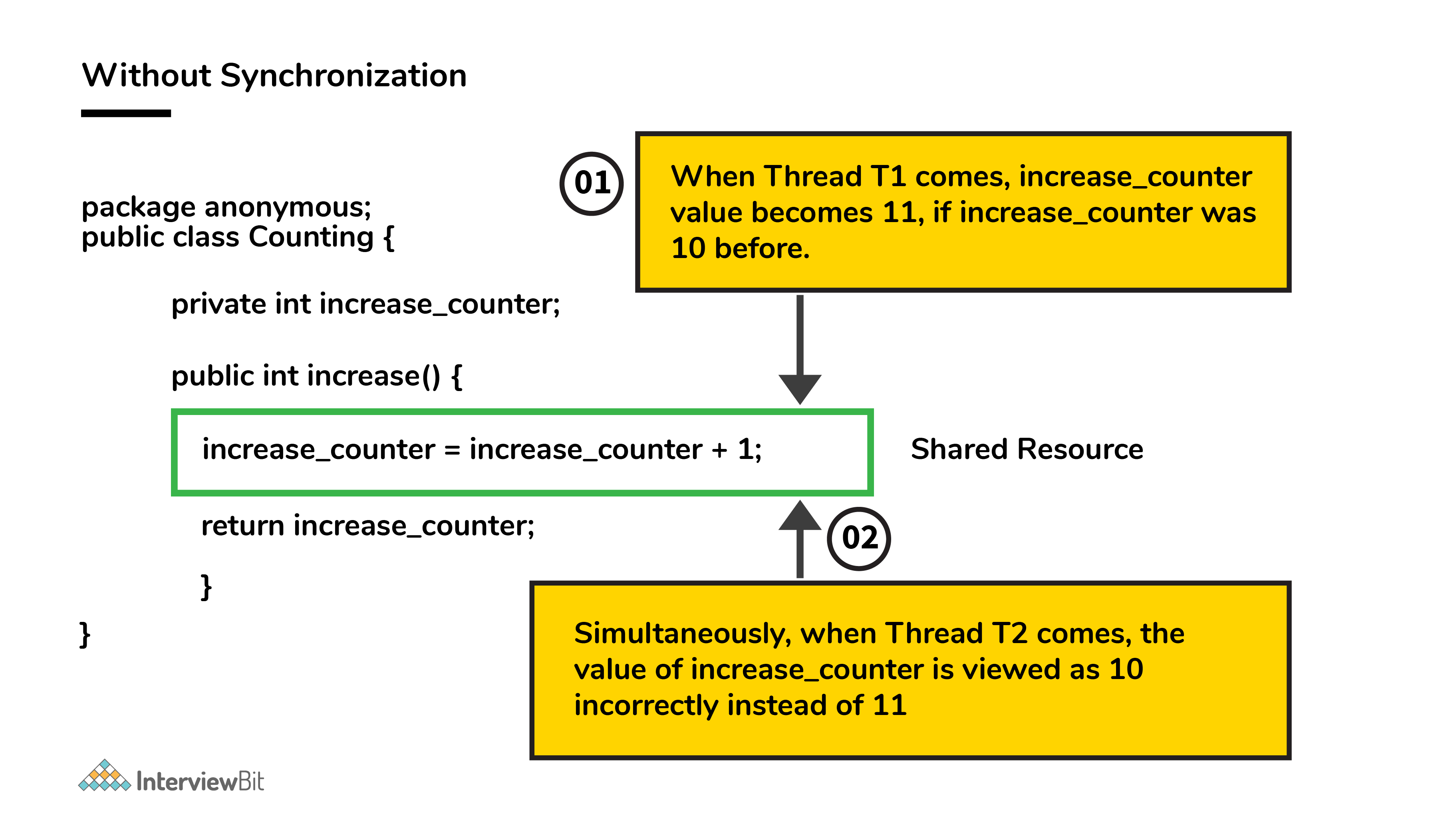
**public** **int** **increase**() {

increase\_counter = increase\_counter + 1;

**return** increase\_counter;

}

}



If a thread Thread1 views the count as 10, it will be increased by 1 to 11. Simultaneously, if another thread Thread2 views the count as 10, it will be increased by 1 to 11. Thus, inconsistency in count values takes place because the expected final value is 12 but the actual final value we get will be 11.

Now, the function increase() is made synchronized so that simultaneous accessing cannot take place.

**With synchronization:**

**package** anonymous;

**public** **class** **Counting** {

**private** **int** increase\_counter;

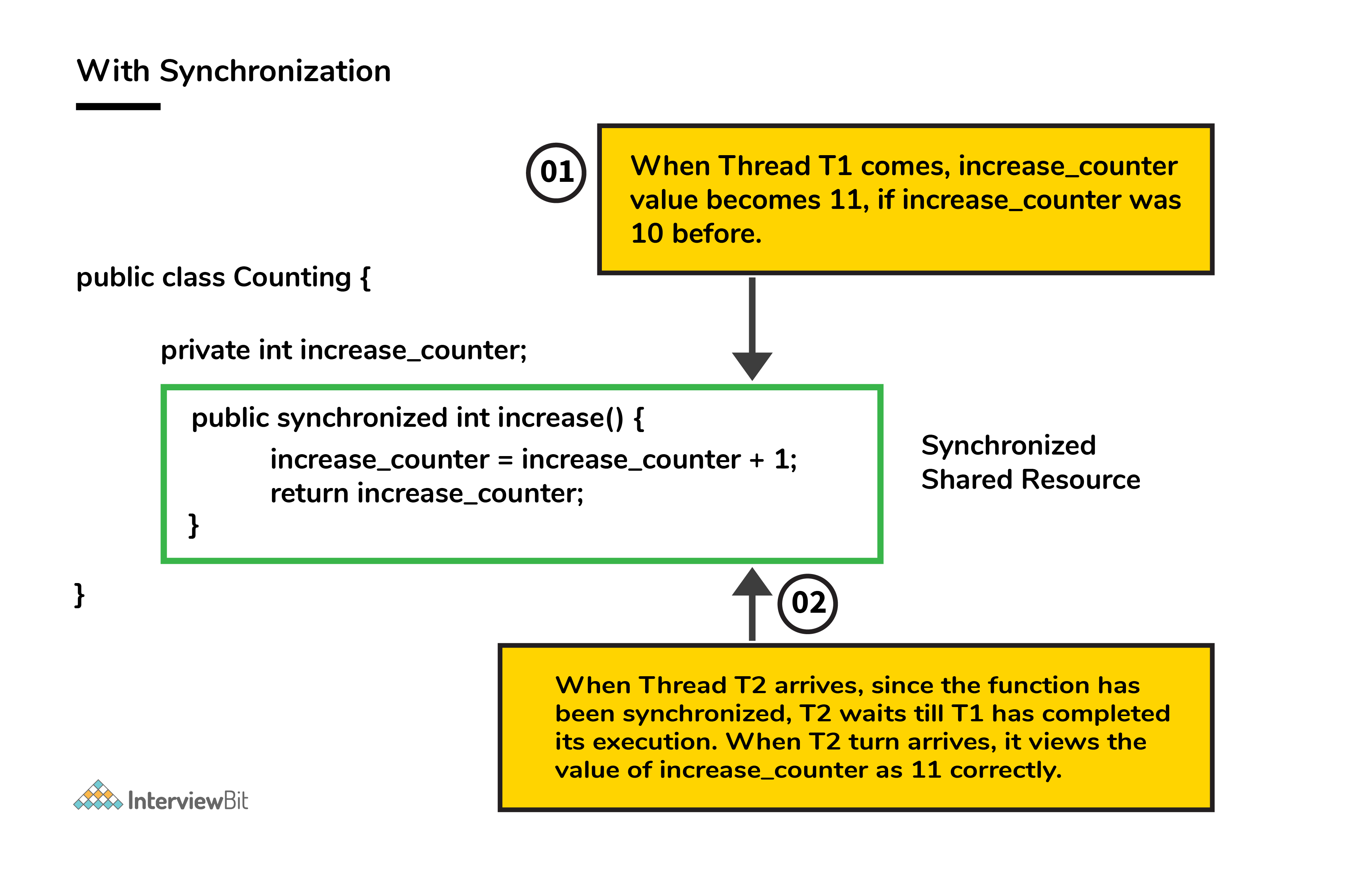
**public** **synchronized** **int** **increase**() {

increase\_counter = increase\_counter + 1;

**return** increase\_counter;

}

}



If a thread Thread1 views the count as 10, it will be increased by 1 to 11, then the thread Thread2 will view the count as 11, it will be increased by 1 to 12. Thus, consistency in count values takes place.

**45. In the given code below, what is the significance of ... ?**

**public** **void** **fooBarMethod**(String... variables){

// method code

}

* Ability to provide ... is a feature called varargs (variable arguments) which was introduced as part of Java 5.
* The function having ... in the above example indicates that it can receive multiple arguments of the datatype String.
* For example, the fooBarMethod can be called in multiple ways and we can still have one method to process the data as shown below:

fooBarMethod("foo", "bar");

fooBarMethod("foo", "bar", "boo");

fooBarMethod(**new** String[]{"foo", "var", "boo"});

**public** **void** **myMethod**(String... variables){

**for**(String variable : variables){

// business logic

}

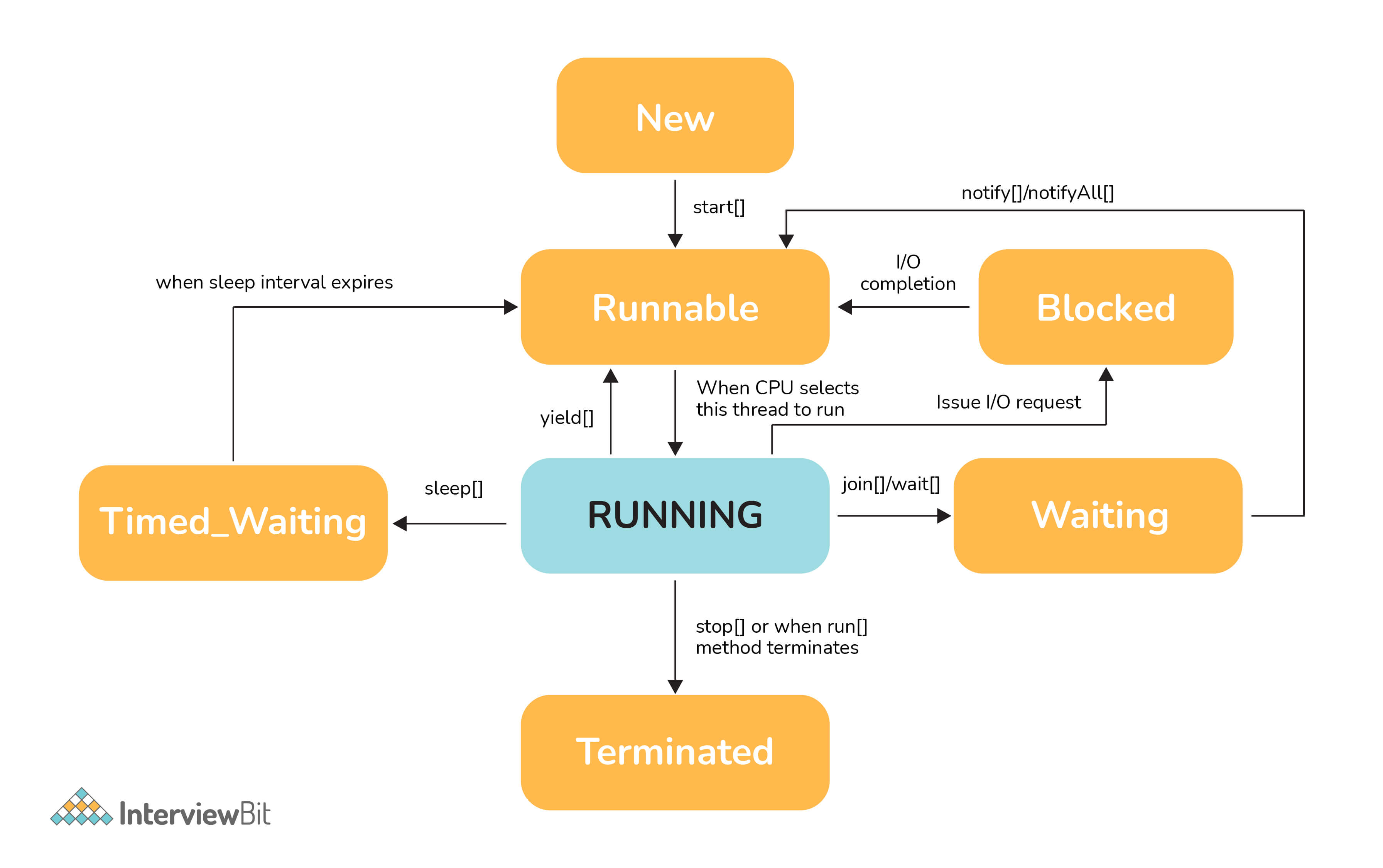
}

**46. Can you explain the Java thread lifecycle?**

Java thread life cycle is as follows:

* **New** – When the instance of the thread is created and the start() method has not been invoked, the thread is considered to be alive and hence in the NEW state.
* **Runnable** – Once the start() method is invoked, before the run() method is called by JVM, the thread is said to be in RUNNABLE (ready to run) state. This state can also be entered from the Waiting or Sleeping state of the thread.
* **Running** – When the run() method has been invoked and the thread starts its execution, the thread is said to be in a RUNNING state.
* **Non-Runnable (Blocked/Waiting)** – When the thread is not able to run despite the fact of its aliveness, the thread is said to be in a NON-RUNNABLE state. Ideally, after some time of its aliveness, the thread should go to a runnable state.
  + A thread is said to be in a Blocked state if it wants to enter synchronized code but it is unable to as another thread is operating in that synchronized block on the same object. The first thread has to wait until the other thread exits the synchronized block.
  + A thread is said to be in a Waiting state if it is waiting for the signal to execute from another thread, i.e it waits for work until the signal is received.
* **Terminated** – Once the run() method execution is completed, the thread is said to enter the TERMINATED step and is considered to not be alive.

The following flowchart clearly explains the lifecycle of the thread in Java.



**47. What could be the tradeoff between the usage of an unordered array versus the usage of an ordered array?**

* The main advantage of having an ordered array is the reduced search time complexity of O(log n) whereas the time complexity in an unordered array is O(n).
* The main drawback of the ordered array is its increased insertion time which is O(n) due to the fact that its element has to reordered to maintain the order of array during every insertion whereas the time complexity in the unordered array is only O(1).
* Considering the above 2 key points and depending on what kind of scenario a developer requires, the appropriate data structure can be used for implementation.

**48. Is it possible to import the same class or package twice in Java and what happens to it during runtime?**

It is possible to import a class or package more than once, however, it is redundant because the JVM internally loads the package or class only once.

**49. In case a package has sub packages, will it suffice to import only the main package? e.g. Does importing of com.myMainPackage.\* also import com.myMainPackage.mySubPackage.\*?**

This is a big NO. We need to understand that the importing of the sub-packages of a package needs to be done explicitly. Importing the parent package only results in the import of the classes within it and not the contents of its child/sub-packages.

**50. Will the finally block be executed if the code System.exit(0) is written at the end of try block?**

NO. The control of the program post System.exit(0) is immediately gone and the program gets terminated which is why the finally block never gets executed.

**51. What do you understand by marker interfaces in Java?**

Marker interfaces, also known as tagging interfaces are those interfaces that have no methods and constants defined in them. They are there for helping the compiler and JVM to get run time-related information regarding the objects.

**52. Explain the term “Double Brace Initialisation” in Java?**

This is a convenient means of initializing any collections in Java. Consider the below example.

**import** java.util.HashSet;

**import** java.util.Set;

**public** **class** **IBDoubleBraceDemo**{

**public** **static** **void** **main**(String[] args){

Set<String> stringSets = **new** HashSet<String>()

{

{

add("set1");

add("set2");

add("set3");

}

};

doSomething(stringSets);

}

**private** **static** **void** **doSomething**(Set<String> stringSets){

System.out.println(stringSets);

}

}

In the above example, we see that the stringSets were initialized by using double braces.

* The first brace does the task of creating an anonymous inner class that has the capability of accessing the parent class’s behavior. In our example, we are creating the subclass of HashSet so that it can use the add() method of HashSet.
* The second braces do the task of initializing the instances.

Care should be taken while initializing through this method as the method involves the creation of anonymous inner classes which can cause problems during the garbage collection or serialization processes and may also result in memory leaks.

**53. Why is it said that the length() method of String class doesn't return accurate results?**

* The length method returns the number of Unicode units of the String. Let's understand what Unicode units are and what is the confusion below.
* We know that Java uses UTF-16 for String representation. With this Unicode, we need to understand the below two Unicode related terms:
  + Code Point: This represents an integer denoting a character in the code space.
  + Code Unit: This is a bit sequence used for encoding the code points. In order to do this, one or more units might be required for representing a code point.
* Under the UTF-16 scheme, the code points were divided logically into 17 planes and the first plane was called the Basic Multilingual Plane (BMP). The BMP has classic characters - U+0000 to U+FFFF. The rest of the characters- U+10000 to U+10FFFF were termed as the supplementary characters as they were contained in the remaining planes.
  + The code points from the first plane are encoded using **one** 16-bit code unit
  + The code points from the remaining planes are encoded using **two** code units.

Now if a string contained supplementary characters, the length function would count that as 2 units and the result of the length() function would not be as per what is expected.

In other words, if there is 1 supplementary character of 2 units, the length of that SINGLE character is considered to be TWO - Notice the inaccuracy here? As per the java documentation, it is expected, but as per the real logic, it is inaccurate.

**54. What is the output of the below code and why?**

**public** **class** **InterviewBit**{

**public** **static** **void** **main**(String[] args)

{

System.out.println('b' + 'i' + 't');

}

}

“bit” would have been the result printed if the letters were used in double-quotes (or the string literals). But the question has the character literals (single quotes) being used which is why concatenation wouldn't occur. The corresponding ASCII values of each character would be added and the result of that sum would be printed.  
The ASCII values of ‘b’, ‘i’, ‘t’ are:

* ‘b’ = 98
* ‘i’ = 105
* ‘t’ = 116

98 + 105 + 116 = 319

Hence 319 would be printed.

**55. What are the possible ways of making object eligible for garbage collection (GC) in Java?**

**First Approach:** Set the object references to null once the object creation purpose is served.

**public** **class** **IBGarbageCollect** {

**public** **static** **void** **main** (String [] args){

String s1 = "Some String";

// s1 referencing String object - not yet eligible for GC

s1 = **null**; // now s1 is eligible for GC

}

}

**Second Approach:** Point the reference variable to another object. Doing this, the object which the reference variable was referencing before becomes eligible for GC.

**public** **class** **IBGarbageCollect** {

**public** **static** **void** **main**(String [] args){

String s1 = "To Garbage Collect";

String s2 = "Another Object";

System.out.println(s1); // s1 is not yet eligible for GC

s1 = s2; // Point s1 to other object pointed by s2

/\* Here, the string object having the content "To Garbage Collect" is not referred by any reference variable. Therefore, it is eligible for GC \*/

}

}

**Third Approach:**Island of Isolation Approach: When 2 reference variables pointing to instances of the same class, and these variables refer to only each other and the objects pointed by these 2 variables don't have any other references, then it is said to have formed an “Island of Isolation” and these 2 objects are eligible for GC.

**public** **class** **IBGarbageCollect** {

IBGarbageCollect ib;

**public** **static** **void** **main**(String [] str){

IBGarbageCollect ibgc1 = **new** IBGarbageCollect();

IBGarbageCollect ibgc2 = **new** IBGarbageCollect();

ibgc1.ib = ibgc2; //ibgc1 points to ibgc2

ibgc2.ib = ibgc1; //ibgc2 points to ibgc1

ibgc1 = **null**;

ibgc2 = **null**;

/\*

\* We see that ibgc1 and ibgc2 objects refer

\* to only each other and have no valid

\* references- these 2 objects for island of isolcation - eligible for GC

\*/

}

}

**Java Interview Programs**

**56. Check if a given string is palindrome using recursion.**

/\*

\* Java program to check if a given inputted string is palindrome or not using recursion.

\*/

**import** java.util.\*;

**public** **class** **InterviewBit** {

**public** **static** **void** **main**(String args[]) {

Scanner s = **new** Scanner(System.in);

String word = s.nextLine();

System.out.println("Is "+word+" palindrome? - "+isWordPalindrome(word));

}

**public** **static** **boolean** **isWordPalindrome**(String word){

String reverseWord = getReverseWord(word);

//if word equals its reverse, then it is a palindrome

**if**(word.equals(reverseWord)){

**return** **true**;

}

**return** **false**;

}

**public** **static** String **getReverseWord**(String word){

**if**(word == **null** || word.isEmpty()){

**return** word;

}

**return** word.charAt(word.length()- 1) + getReverseWord(word.substring(0, word.length() - 1));

}

}

**57. Write a Java program to check if the two strings are anagrams.**

The main idea is to validate the length of strings and then if found equal, convert the string to char array and then sort the arrays and check if both are equal.

**import** java.util.Arrays;

**import** java.util.Scanner;

**public** **class** **InterviewBit** {

**public** **static** **void** **main**(String[] args) {

Scanner s = **new** Scanner(System.in);

//Input from two strings

System.out.print("First String: ");

String string1 = s.nextLine();

System.out.print("Second String: ");

String string2 = s.nextLine();

// check for the length

**if**(string1.length() == string2.length()) {

// convert strings to char array

**char**[] characterArray1 = string1.toCharArray();

**char**[] characterArray2 = string2.toCharArray();

// sort the arrays

Arrays.sort(characterArray1);

Arrays.sort(characterArray2);

// check for equality, if found equal then anagram, else not an anagram

**boolean** isAnagram = Arrays.equals(characterArray1, characterArray2);

System.out.println("Anagram: "+ isAnagram);

}

}

**58. Write a Java Program to find the factorial of a given number.**

**public** **class** **FindFactorial** {

**public** **static** **void** **main**(String[] args) {

**int** num = 10;

**long** factorialResult = 1l;

**for**(**int** i = 1; i <= num; ++i)

{

factorialResult \*= i;

}

System.out.println("Factorial: "+factorialResult);

}

}

**59. Given an array of non-duplicating numbers from 1 to n where one number is missing, write an efficient java program to find that missing number.**

Idea is to find the sum of n natural numbers using the formula and then finding the sum of numbers in the given array. Subtracting these two sums results in the number that is the actual missing number. This results in O(n) time complexity and O(1) space complexity.

**public** **class** **IBMissingNumberProblem** {

**public** **static** **void** **main**(String[] args) {

**int**[] array={4,3,8,7,5,2,6};

**int** missingNumber = findMissingNum(array);

System.out.println("Missing Number is "+ missingNumber);

}

**public** **static** **int** **findMissingNum**(**int**[] array) {

**int** n=array.length+1;

**int** sumOfFirstNNums=n\*(n+1)/2;

**int** actualSumOfArr=0;

**for** (**int** i = 0; i < array.length; i++) {

actualSumOfArr+=array[i];

}

**return** sumOfFirstNNums-actualSumOfArr;

}

}

**60. Write a Java Program to check if any number is a magic number or not. A number is said to be a magic number if after doing sum of digits in each step and inturn doing sum of digits of that sum, the ultimate result (when there is only one digit left) is 1.**

Example, consider the number:

* Step 1: 163 => 1+6+3 = 10
* Step 2: 10 => 1+0 = 1 => Hence 163 is a magic number

**public** **class** **IBMagicNumber**{

**public** **static** **void** **main**(String[] args) {

**int** num = 163;

**int** sumOfDigits = 0;

**while** (num > 0 || sumOfDigits > 9)

{

**if** (num == 0)

{

num = sumOfDigits;

sumOfDigits = 0;

}

sumOfDigits += num % 10;

num /= 10;

}

// If sum is 1, original number is magic number

**if**(sumOfDigits == 1) {

System.out.println("Magic number");

}**else** {

System.out.println("Not magic number");

}

}

}

**Conclusion**

**61. Conclusion**

Java is one of the simple high-level languages that provides powerful tools and impressive standards required for application development. It was also one of the first languages to provide amazing threading support for tackling concurrency-based problems. The easy-to-use syntax and the built-in features of Java combined with the stability it provides to applications are the main reasons for this language to have ever-growing usage in the software community.

Join our [community](https://discuss.interviewbit.com/t/a-cheat-sheet-for-your-java-interview/62512) and share your java interview experiences.

**Recommended tutorials:**

[Java Tutorial](https://www.interviewbit.com/courses/fast-track-java/)

[Puzzles](https://www.interviewbit.com/puzzles/)

[Coding Interview Questions](https://www.interviewbit.com/coding-interview-questions/%20%20)

[Java 8 Interview Questions](https://www.interviewbit.com/java-8-interview-questions/)

**Java Frameworks:**

[Spring](https://www.interviewbit.com/spring-interview-questions/)

[Hibernate](https://www.interviewbit.com/hibernate-interview-questions/)

[**JAVA SE Download**](https://www.oracle.com/in/java/technologies/javase-downloads.html)

**Java interview MCQs**

1.

What is the component used for compiling, debugging, and executing java programs?

JDK

JVM

JRE

JIT

2.

What component does the task of bytecode to machine code conversion?

JDK

JVM

JRE

JIT

3.

Which of the following is the functionality of the java interpreter?

Interpretor is nothing but the JIT compiler.

It acts as medium between JVM and JIT.

It does the conversion of byte code to machine code.

It reads the high level code and executes them.

4.

When an object has its own lifecycle and its child object cant belong to another parent object, what is it called?

Association

Aggregation

Composition

Encapsulation

5.

What is the output of the below piece of code?

**class** **InterviewBit** {

**public** **void** **method1** (**int** num1,**float** num2){

System.out.println("int-float method");

}

**public** **void** **method1**(**float** num1,**int** num2){

System.out.println("float-int method");

}

**public** **static** **void** **main**(String[] args){

InterviewBit interviewBit=**new** InterviewBit();

interviewBit.method1(40,20);

}

}

int-float method

float-int method

Compilation Error

Run Time error

6.

What is the output of the following code?

**class** **InterviewBit**{

**int** **fun** (**int** n)

{

**int** result;

result = fun (n - 1);

**return** result;

}

}

**class** **Driver**{

**public** **static** **void** **main**(String args[])

{

InterviewBit ib = **new** InterviewBit() ;

System.out.print(ib.fun(12));

}

}

0

1

Compilation Error

Run time error

7.

Which of the following happens when the garbage collection process kicks off during the execution of the thread?

Garbage collection does not happen during thread execution.

Thread pauses while the garbage collection process runs.

Both the process takes place simultaneously and does not interfere its execution.

Nothing happens, the thread proceeds with execution.

8.

What is the output of the below code?

**class** **InterviewBit**{

**public** **static** **void** **main**(String args[])

{

String obj = "Hello";

String obj1 = "InterviewBit";

String obj2 = "Hello";

System.out.println(obj.equals(obj1) + " " + obj.equals(obj2));

}

}

false false

true true

true false

false true

9.

What is the functionality of Class.getInstance()?

It invokes the constructor.

It has the same functionality of new operator.

It creates object if the class does not have constructor defined.

None of the above.

10.

What is the output of the below code?

**class** **InterviewBit**{

**public** **int** num1;

**static** **int** num2;

**void** **calculate**(**int** a, **int** b)

{

num1 += a ;

num2 += b;

}

}

**class** **Driver**{

**public** **static** **void** **main**(String args[])

{

InterviewBit obj1 = **new** InterviewBit();

InterviewBit obj2 = **new** InterviewBit();

obj1.num1 = 0;

obj1.num2 = 0;

obj1.calculate(1, 2);

obj2.num1 = 0;

obj2.calculate(2, 3);

System.out.println(obj1.num1 + " " + obj2.num2);

}

}

1 2

1 5

4 2

2 5

11.

What is the output of the following code?

**class** **InterviewBit**{

**int** **calculate**(**int** a, **int** b)

{

**try**{

**return** a-b;

}**catch**(Exception e){

**return** a+b;

}**finally**{

**return** a\*b;

}

}

}

**class** **Driver**{

**public** **static** **void** **main**(String args[])

{

InterviewBit obj1 = **new** InterviewBit();

**int** result = obj1.calculate(2, 3);

System.out.println("Result: " + result);

}

}

5

-1

6

Run time Error

×



Preparing or tech  
interviews?

Learn from the best  
instructors in the field.

Try Academy for FREE

[**Register Now**](https://scaler.com/academy?utm_source=ib&utm_campaign=free_academy&utm_medium=nudge_modal&utm_content=/java-interview-questions)

* [Blog](https://blog.interviewbit.com/)

* [About Us](https://www.interviewbit.com/pages/about_us/)

* [FAQ](https://www.interviewbit.com/pages/faq/)

* [Contact Us](https://www.interviewbit.com/pages/contact_us/)

* [Terms](https://www.interviewbit.com/pages/terms/)

* [Privacy Policy](https://www.interviewbit.com/pages/privacy/)
* [Scaler Academy Review](https://www.interviewbit.com/scaler-academy-review)

* [System Design Interview Questions](https://www.interviewbit.com/courses/system-design/)

* [Google Interview Questions](https://www.interviewbit.com/google-interview-questions/)

* [Facebook Interview Questions](https://www.interviewbit.com/facebook-interview-questions/)

* [Amazon Interview Questions](https://www.interviewbit.com/amazon-interview-questions/)

* [Microsoft Interview Questions](https://www.interviewbit.com/microsoft-interview-questions/)

* [SQL Interview Questions](https://www.interviewbit.com/sql-interview-questions/)

* [Python Interview Questions](https://www.interviewbit.com/python-interview-questions/)

* [Javascript Interview Questions](https://www.interviewbit.com/javascript-interview-questions/)

* [MVC Interview Questions](https://www.interviewbit.com/mvc-interview-questions/)

* [React Interview Questions](https://www.interviewbit.com/react-interview-questions/)

* [jQuery Interview Questions](https://www.interviewbit.com/jquery-interview-questions/)

* [Angular Interview Questions](https://www.interviewbit.com/angular-interview-questions/)

* [Spring Interview Questions](https://www.interviewbit.com/spring-interview-questions/)

* [Data Structure Interview Questions](https://www.interviewbit.com/data-structure-interview-questions/)

* [Selenium Interview Questions](https://www.interviewbit.com/selenium-interview-questions/)

* [HTML Interview Questions](https://www.interviewbit.com/html-interview-questions/)

* [Directi Interview Questions](https://www.interviewbit.com/directi-interview-questions/)

* [Yahoo Interview Questions](https://www.interviewbit.com/yahoo-interview-questions/)

* [LinkedIn Interview Questions](https://www.interviewbit.com/linkedin-interview-questions/)

* [VMware Interview Questions](https://www.interviewbit.com/vmware-interview-questions/)

* [eBay Interview Questions](https://www.interviewbit.com/ebay-interview-questions/)

* [Flipkart Interview Questions](https://www.interviewbit.com/flipkart-interview-questions/)

* [C Interview Questions](https://www.interviewbit.com/c-interview-questions/)

* [Php Interview Questions](https://www.interviewbit.com/php-interview-questions/)

* [C Sharp Interview Questions](https://www.interviewbit.com/c-sharp-interview-questions/)

* [Web Api Interview Questions](https://www.interviewbit.com/web-api-interview-questions/)

* [Hibernate Interview Questions](https://www.interviewbit.com/hibernate-interview-questions/)

* [Node Js Interview Questions](https://www.interviewbit.com/node-js-interview-questions/)

* [Cpp Interview Questions](https://www.interviewbit.com/cpp-interview-questions/)

* [Oops Interview Questions](https://www.interviewbit.com/oops-interview-questions/)

* [Devops Interview Questions](https://www.interviewbit.com/devops-interview-questions/)

* [Machine Learning Interview Questions](https://www.interviewbit.com/machine-learning-interview-questions/)

* [Docker Interview Questions](https://www.interviewbit.com/docker-interview-questions/)

* [Mysql Interview Questions](https://www.interviewbit.com/mysql-interview-questions/)

* [Css Interview Questions](https://www.interviewbit.com/css-interview-questions/)

* [Laravel Interview Questions](https://www.interviewbit.com/laravel-interview-questions/)

* [Asp Net Interview Questions](https://www.interviewbit.com/asp-net-interview-questions/)

* [Django Interview Questions](https://www.interviewbit.com/django-interview-questions/)

* [Dot Net Interview Questions](https://www.interviewbit.com/dot-net-interview-questions/)

* [Kubernetes Interview Questions](https://www.interviewbit.com/kubernetes-interview-questions/)

* [Operating System Interview Questions](https://www.interviewbit.com/operating-system-interview-questions/)

* [React Native Interview Questions](https://www.interviewbit.com/react-native-interview-questions/)

* [Aws Interview Questions](https://www.interviewbit.com/aws-interview-questions/)

* [Git Interview Questions](https://www.interviewbit.com/git-interview-questions/)

* [Java 8 Interview Questions](https://www.interviewbit.com/java-8-interview-questions/)

* [Mongodb Interview Questions](https://www.interviewbit.com/mongodb-interview-questions/)

* [Dbms Interview Questions](https://www.interviewbit.com/dbms-interview-questions/)

* [Spring Boot Interview Questions](https://www.interviewbit.com/spring-boot-interview-questions/)

* [Power Bi Interview Questions](https://www.interviewbit.com/power-bi-interview-questions/)

* [Pl Sql Interview Questions](https://www.interviewbit.com/pl-sql-interview-questions/)

* [Tableau Interview Questions](https://www.interviewbit.com/tableau-interview-questions/)

* [Linux Interview Questions](https://www.interviewbit.com/linux-interview-questions/)

* [Ansible Interview Questions](https://www.interviewbit.com/ansible-interview-questions/)

* [Java Interview Questions](https://www.interviewbit.com/java-interview-questions/)

* [Jenkins Interview Questions](https://www.interviewbit.com/jenkins-interview-questions/)

* [Agile Interview Questions](https://www.interviewbit.com/agile-interview-questions/)

* [Networking Interview Questions](https://www.interviewbit.com/networking-interview-questions/)

* [Azure Interview Questions](https://www.interviewbit.com/azure-interview-questions/)

* [Software Testing Interview Questions](https://www.interviewbit.com/software-testing-interview-questions/)

* [Automation Testing Interview Questions](https://www.interviewbit.com/automation-testing-interview-questions/)
* [Like Us](https://www.facebook.com/interviewbit)

* [Follow Us](https://twitter.com/interview_bit)

* [Email](mailto:hello@interviewbit.com)

[Skip to content](https://github.com/Devinterview-io/java-interview-questions#start-of-content)

* Why GitHub?
* [Team](https://github.com/team)
* [Enterprise](https://github.com/enterprise)
* Explore
* [Marketplace](https://github.com/marketplace)
* Pricing

Top of Form



Bottom of Form

[Sign in](https://github.com/login?return_to=%2FDevinterview-io%2Fjava-interview-questions)[Sign up](https://github.com/join?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F%3Cuser-name%3E%2F%3Crepo-name%3E&source=header-repo&source_repo=Devinterview-io%2Fjava-interview-questions)

[**Devinterview-io**](https://github.com/Devinterview-io)**/**[**java-interview-questions**](https://github.com/Devinterview-io/java-interview-questions)

* [Notifications](https://github.com/login?return_to=%2FDevinterview-io%2Fjava-interview-questions)
* [Star](https://github.com/login?return_to=%2FDevinterview-io%2Fjava-interview-questions)[**1**](https://github.com/Devinterview-io/java-interview-questions/stargazers)
* [Fork](https://github.com/login?return_to=%2FDevinterview-io%2Fjava-interview-questions)[**0**](https://github.com/Devinterview-io/java-interview-questions/network/members)
* [**Code**](https://github.com/Devinterview-io/java-interview-questions)
* [Issues](https://github.com/Devinterview-io/java-interview-questions/issues)
* [Pull requests](https://github.com/Devinterview-io/java-interview-questions/pulls)
* [Actions](https://github.com/Devinterview-io/java-interview-questions/actions)
* [Projects](https://github.com/Devinterview-io/java-interview-questions/projects)
* [Wiki](https://github.com/Devinterview-io/java-interview-questions/wiki)
* [Security](https://github.com/Devinterview-io/java-interview-questions/security)
* [Insights](https://github.com/Devinterview-io/java-interview-questions/pulse)

 main

[**1** branch](https://github.com/Devinterview-io/java-interview-questions/branches)[**0** tags](https://github.com/Devinterview-io/java-interview-questions/tags)

[Go to file](https://github.com/Devinterview-io/java-interview-questions/find/main) Code

**Latest commit**

[@Devinterview-io](https://github.com/Devinterview-io)

[**Devinterview-io**](https://github.com/Devinterview-io/java-interview-questions/commits?author=Devinterview-io) [Update README.md](https://github.com/Devinterview-io/java-interview-questions/commit/a64204240460d4555a2e491dc96cbda22b6269bd)

[a642042on Jan 21](https://github.com/Devinterview-io/java-interview-questions/commit/a64204240460d4555a2e491dc96cbda22b6269bd)

**Git stats**

* [**2** commits](https://github.com/Devinterview-io/java-interview-questions/commits/main)

**Files**

Type

Name

Latest commit message

Commit time

[README.md](https://github.com/Devinterview-io/java-interview-questions/blob/main/README.md)

[Update README.md](https://github.com/Devinterview-io/java-interview-questions/commit/a64204240460d4555a2e491dc96cbda22b6269bd)

4 months ago

[**README.md**](https://github.com/Devinterview-io/java-interview-questions#readme)

**Top 147 Java interview questions and answers in 2021.**

You can check all 147 Java interview questions here 👉 <https://devinterview.io/dev/java-interview-questions>

**🔹 1. What are the two types of Exceptions in Java? Which are the differences between them?**

**Answer:**

Java has two types of exceptions: checked exceptions and unchecked exceptions.

1. **Unchecked exceptions**do not need to be declared in a method or a constructor’s throws clause, if they can be thrown by the execution of the method or the constructor, and propagate outside the method or constructor boundary.
2. On the other hand, **checked exceptions** must be declared in a method or a constructor’s throws clause.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 2. What is JVM? Why is Java called the “Platform Independent Programming Language”?**

**Answer:**

A Java virtual machine (JVM) is a process virtual machine that can execute Java bytecode. Each Java source file is compiled into a bytecode file, which is executed by the JVM. Java was designed to allow application programs to be built that could be run on any platform, without having to be rewritten or recompiled by the programmer for each separate platform. A Java virtual machine makes this possible, because it is aware of the specific instruction lengths and other particularities of the underlying hardware platform.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 3. What is the difference between an Applet and a Java Application?**

**Answer:**

* Applets are executed within a Java enabled browser, but a
* Java application is a standalone Java program that can be executed outside of a browser.

However, they both require the existence of a Java Virtual Machine (JVM). Furthermore, a Java application requires a main method with a specific signature, in order to start its execution. Java applets don’t need such a method to start their execution. Finally, Java applets typically use a restrictive security policy, while Java applications usually use more relaxed security policies.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 4. What is the Difference between JDK and JRE?**

**Answer:**

* **The Java Runtime Environment (JRE)**is basically the Java Virtual Machine (**JVM**) where your Java programs are being executed. It also includes browser plugins for applet execution.
* **The Java Development Kit (JDK)** is the full featured Software Development Kit for Java, including the JRE, the compilers and tools (like JavaDoc, and Java Debugger), in order for a user to develop, compile and execute Java applications.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 5. What is a Servlet?**

**Answer:**

The servlet is a Java programming language class used to process client requests and generate dynamic web content. Servlets are mostly used to process or store data submitted by an HTML form, provide dynamic content and manage state information that does not exist in the stateless HTTP protocol.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 6. What is a JSP Page?**

**Answer:**

A **Java Server Page (JSP)** is a text document that contains two types of text:

* static data and
* JSP elements.

Static data can be expressed in any text-based format, such as HTML or XML. JSP is a technology that mixes static content with dynamically-generated content.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 7. What are Directives?**

**Answer:**

What are the different types of Directives available in JSP ? Directives are instructions that are processed by the JSP engine, when the page is compiled to a servlet. Directives are used to set page-level instructions, insert data from external files, and specify custom tag libraries. Directives are defined between < %@ and % >.The different types of directives are shown below:

* Include directive: it is used to include a file and merges the content of the file with the current page.
* Page directive: it is used to define specific attributes in the JSP page, like error page and buffer.
* Taglib: it is used to declare a custom tag library which is used in the page.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 8. What does System.gc() and Runtime.gc() methods do?**

**Answer:**

These methods can be used as a hint to the JVM, in order to start a garbage collection. However, this it is up to the Java Virtual Machine (JVM) to start the garbage collection immediately or later in time.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 9. What differences exist between HashMap and Hashtable?**

**Answer:**

There are several differences between HashMap and Hashtable in Java:

1. Hashtable is synchronized, whereas HashMap is not. This makes HashMap better for non-threaded applications, as unsynchronized Objects typically perform better than synchronized ones.
2. Hashtable does not allow null keys or values. HashMap allows one null key and any number of null values.
3. One of HashMap's subclasses is LinkedHashMap, so in the event that you'd want predictable iteration order (which is insertion order by default), you could easily swap out the HashMap for a LinkedHashMap. This wouldn't be as easy if you were using Hashtable.

*Source:* [stackoverflow.com](https://stackoverflow.com/questions/40471/differences-between-hashmap-and-hashtable)

**🔹 10. What is JDBC?**

**Answer:**

JDBC is an abstraction layer that allows users to choose between databases. [JDBC enables developers to write database applications in Java](http://www.javacodegeeks.com/2014/03/java-8-friday-java-8-will-revolutionize-database-access.html), without having to concern themselves with the underlying details of a particular database.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 11. What does the “static” keyword mean? Can you override private or static method in Java?**

**Answer:**

The static keyword denotes that a member variable or method can be accessed, *without requiring an instantiation of the class to which it belongs*.

A user cannot override static methods in Java, because method overriding is based upon dynamic binding at runtime and static methods are statically binded at compile time. A static method is not associated with any instance of a class so the concept is not applicable.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 12. What is the importance of finally block in exception handling?**

**Answer:**

A *finally* block will always be executed, whether or not an exception is actually thrown. Even in the case where the catch statement is missing and an exception is thrown, the finally block will still be executed. Last thing to mention is that the finally block is used to release resources like I/O buffers, database connections, etc.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 13. What is the difference between Exception and Error in java?**

**Answer:**

* An **Error** "indicates serious problems that a reasonable application should not try to catch."
* An **Exception** "indicates conditions that a reasonable application might want to catch."

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 14. When does an Object becomes eligible for Garbage collection in Java ?**

**Answer:**

A Java object is subject to garbage collection when it becomes unreachable to the program in which it is currently used.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 15. What is an Iterator?**

**Answer:**

The [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) interface provides a number of methods that are able to iterate over any [Collection](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html). Each Java [Collection](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html) contains the [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) method that returns an [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) instance. Iterators are capable of removing elements from the underlying collection during the iteration.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 16. What are pass by reference and pass by value?**

**Answer:**

* When an object is **passed by value**, this means that a copy of the object is passed. Thus, even if changes are made to that object, it doesn’t affect the original value.
* When an object is **passed by reference**, this means that the actual object is not passed, rather a reference of the object is passed. Thus, any changes made by the external method, are also reflected in all places.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 17. What is an Java Applet?**

**Answer:**

A Java Applet is program that can be included in a HTML page and be executed in a java enabled client browser. Applets are used for creating dynamic and interactive web applications.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 18. How HashMap works in Java?**

**Answer:**

[A HashMap in Java stores key-value pairs](http://www.javacodegeeks.com/2014/03/how-hashmap-works-in-java.html). The [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) requires a hash function and uses hashCode and equals methods, in order to put and retrieve elements to and from the collection respectively. When the put method is invoked, the [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) calculates the hash value of the key and stores the pair in the appropriate index inside the collection. If the key exists, its value is updated with the new value. Some important characteristics of a [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) are its capacity, its load factor and the threshold resizing.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 19. What are the basic interfaces of Java Collections Framework?**

**Answer:**

**Java Collections Framework** provides a well designed set of interfaces and classes that support operations on a collections of objects. The most basic interfaces that reside in the Java Collections Framework are:

* **Collection**, which represents a group of objects known as its elements.
* **Set**, which is a collection that cannot contain duplicate elements.
* **List**, which is an ordered collection and can contain duplicate elements.
* **Map**, which is an object that maps keys to values and cannot contain duplicate keys.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 20. What are the Data Types supported by Java? What is Autoboxing and Unboxing?**

**Answer:**

The eight primitive data types supported by the Java programming language are:

* byte
* short
* int
* long
* float
* double
* boolean
* char

**Autoboxing** is the automatic conversion made by the Java compiler between the primitive types and their corresponding object wrapper classes. If the conversion goes the other way, this operation is called **unboxing**.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 21. What is the difference between processes and threads?**

**Answer:**

The main difference between them is that

* a **Process** is a program which is executing some code and
* a **Thread** is an independent path of execution in the process.

A process can have more than one thread for doing independent task e.g. a thread for reading data from disk, a thread for processing that data and another thread for sending that data over the network.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 22. What will happen to the Exception object after exception handling?**

**Answer:**

The [Exception](http://docs.oracle.com/javase/7/docs/api/java/lang/Exception.html) object will be garbage collected in the next garbage collection.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 23. What is the difference between an Interface and an Abstract class?**

**Answer:**

Java provides and supports the creation both of **abstract** classes and **interfaces**. Both implementations share some common characteristics, but they differ in the following features:

* All methods in an interface are implicitly abstract. On the other hand, an abstract class may contain both abstract and non-abstract methods.
* A class may implement a number of Interfaces, but can extend only one abstract class.
* In order for a class to implement an interface, it must implement all its declared methods. However, a class may not implement all declared methods of an abstract class. Though, in this case, the sub-class must also be declared as abstract.
* Abstract classes can implement interfaces without even providing the implementation of interface methods.
* Variables declared in a Java interface is by default final. An abstract class may contain non-final variables.
* Members of a Java interface are public by default. A member of an abstract class can either be private, protected or public.
* An interface is absolutely abstract and cannot be instantiated. An abstract class also cannot be instantiated, but can be invoked if it contains a main method.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 24. What are Expressions?**

**Answer:**

A JSP expression is used to insert the value of a scripting language expression, converted into a string, into the data stream returned to the client, by the web server. Expressions are defined between <% = and %> tags.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 25. What do you know about the big-O notation and can you give some examples with respect to different data structures?**

**Answer:**

The Big-O notation simply describes how well an algorithm scales or performs in the worst case scenario as the number of elements in a data structure increases. The Big-O notation can also be used to describe other behavior such as memory consumption. Since the collection classes are actually data structures, we usually use the Big-O notation to chose the best implementation to use, based on time, memory and performance. Big-O notation can give a good indication about performance for large amounts of data.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 26. What is Function Overriding and Overloading in Java?**

**Answer:**

* Method **overloading** in Java occurs when two or more methods in the same class have the exact same name, but different parameters.

class Dog{

public void bark(){

System.out.println("woof ");

}

<span class="token cComment">//overloading method</span>

<span class="token cVar">public</span> <span class="token cVar">void</span> <span class="token cMod">bark</span><span class="token cBase">(</span><span class="token cVar">int</span> num<span class="token cBase">)</span><span class="token cBase">{</span>

<span class="token cVar">for</span><span class="token cBase">(</span><span class="token cVar">int</span> i<span class="token cBase">=</span><span class="token cNum">0</span><span class="token cBase">;</span> i<span class="token cBase">&lt;</span>num<span class="token cBase">;</span> i<span class="token cBase">++</span><span class="token cBase">)</span>

<span class="token class-name">System</span><span class="token cBase">.</span>out<span class="token cBase">.</span><span class="token cMod">println</span><span class="token cBase">(</span><span class="token cString">"woof "</span><span class="token cBase">)</span><span class="token cBase">;</span>

<span class="token cBase">}</span>

}

* On the other hand, method **overriding** is defined as the case when a child class redefines the same method as a parent class. Overridden methods must have the same name, argument list, and return type. The overriding method may not limit the access of the method it overrides.

class Dog{

public void bark(){

System.out.println("woof ");

}

}

class Hound extends Dog{

public void sniff(){

System.out.println("sniff ");

}

<span class="token cVar">public</span> <span class="token cVar">void</span> <span class="token cMod">bark</span><span class="token cBase">(</span><span class="token cBase">)</span><span class="token cBase">{</span>

<span class="token class-name">System</span><span class="token cBase">.</span>out<span class="token cBase">.</span><span class="token cMod">println</span><span class="token cBase">(</span><span class="token cString">"bowl"</span><span class="token cBase">)</span><span class="token cBase">;</span>

<span class="token cBase">}</span>

}

public class OverridingTest{

public static void main(String [] args){

Dog dog = new Hound();

dog.bark();

}

}

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 27. How are the JSP requests handled?**

**Answer:**

On the arrival of a JSP request, the browser first requests a page with a .jsp extension. Then, the Web server reads the request and using the JSP compiler, the Web server converts the JSP page into a servlet class. Notice that the JSP file is compiled only on the first request of the page, or if the JSP file has changed.The generated servlet class is invoked, in order to handle the browser’s request. Once the execution of the request is over, the servlet sends a response back to the client. See [how to get Request parameters in a JSP](http://examples.javacodegeeks.com/enterprise-java/jsp/get-request-parameter-in-jsp-page/).

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 28. What is the design pattern that Java uses for all Swing components?**

**Answer:**

The design pattern used by Java for all Swing components is the Model View Controller (MVC) pattern.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 29. What is the purpose Class.forName method?**

**Answer:**

This method is used to method is used to load the driver that will establish a connection to the database.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 30. What is the purpose of garbage collection in Java, and when is it used?**

**Answer:**

The purpose of garbage collection is to identify and discard those objects that are no longer needed by the application, in order for the resources to be reclaimed and reused.

*Source:* [github.com/snowdream](https://github.com/snowdream/115-Java-Interview-Questions-and-Answers)

**🔹 31. What’s the difference between sendRedirect and forward methods?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 32. What are Decalarations?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 33. What are JSP actions?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 34. What is reflection and why is it useful?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 35. Explain the architechure of a Servlet.**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 36. Explain Serialization and Deserialization.**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 37. How does Garbage Collection prevent a Java application from going out of memory?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 38. How do I efficiently iterate over each entry in a Java Map?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 39. What is a Constructor, Constructor Overloading in Java and Copy-Constructor?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 40. What is the role of stub in RMI?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 41. What is the difference between doGet() and doPost()?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 42. What is the difference between GenericServlet and HttpServlet?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 43. Explain the life cycle of a Servlet.**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 44. What is structure of Java Heap?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 45. What differences exist between Iterator and ListIterator?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 46. Explain the role of Driver in JDBC.**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 47. What is the relationship between an event-listener interface and an event-adapter class?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 48. What is the applet security manager, and what does it provide?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 49. What is the difference between throw and throws?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 50. What happens when an applet is loaded?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 51. When is the finalize() called? What is the purpose of finalization?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 52. Can you access non static variable in static context?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 53. What is a Server Side Include (SSI)?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 54. Why Collection doesn’t extend Cloneable and Serializable interfaces?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 55. Explain the life cycle of an Applet.**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 56. Which Swing methods are thread-safe?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 57. What is the tradeoff between using an unordered array versus an ordered array?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 58. What is Comparable and Comparator interface? List their differences.**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 59. What are untrusted applets?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 60. What are the restrictions imposed on Java applets?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 61. What is the difference between an Applet and a Servlet?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 62. How does finally block differ from finalize() method?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 63. What’s a deadlock?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 64. What are Scriptlets?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 65. What’s the difference between Enumeration and Iterator interfaces?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 66. What is difference between ArrayList and LinkedList ?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 67. What are the steps involved to make work a RMI program?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 68. Does Java support multiple inheritance?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 69. If an object reference is set to null, will the Garbage Collector immediately free the memory held by that object?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 70. What is the importance of hashCode() and equals() methods?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 71. Explain different ways of creating a thread. Which one would you prefer and why?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 72. What is Java Priority Queue?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 73. What is difference between fail-fast and fail-safe?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 74. What is difference between Array and ArrayList ? When will you use Array over ArrayList?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 75. What is the advantage of PreparedStatement over Statement?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 76. What is meant by JSP implicit objects and what are they?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 77. How threadsafe is enum in Java?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 78. What are the differences between == and equals?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 79. Is there anything like static class in java?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 80. Compare the sleep() and wait() methods in Java**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 81. How and where are Annotations used in Java?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 82. What is the difference between final, finalize and finally?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 83. What is the main difference between StringBuffer and StringBuilder?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 84. What’s the difference between a ClassNotFoundException and NoClassDefFoundError?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 85. How can I synchornize two Java processes?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 86. How do I break out of nested loops in Java?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 87. What are the advantages of JSP?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 88. What is the JIT?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 89. What is a JavaBean exactly?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 90. Can an enum be extended?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 91. What is the Java Classloader?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 92. What is the volatile keyword useful for?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 93. What is the difference between HashMap, LinkedHashMap and TreeMap in Java?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 94. What is the difference between public, protected, package-private and private in Java?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 95. What's the advantage of using getters and setters?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 96. What is static initializer?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 97. Is Java “pass-by-reference” or “pass-by-value”?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 98. Why does Java have transient fields?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 99. What do the ... dots in the method parameters mean?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

**🔹 100. Can == be used on enum?**

👉🏼 Check [all 147 answers](https://devinterview.io/dev/java-interview-questions)

Thanks 🙌 for reading and good luck on your next tech interview!  
Explore 3800+ dev interview question here 👉 [Devinterview.io](https://devinterview.io/)

**About**

🟣 Java Coding Interview Questions Answered to help you get ready for your next coding interview.

**Topics**

[java](https://github.com/topics/java) [java-interview-questions](https://github.com/topics/java-interview-questions) [java-interview](https://github.com/topics/java-interview)

**Resources**

[Readme](https://github.com/Devinterview-io/java-interview-questions#readme)

[**Releases**](https://github.com/Devinterview-io/java-interview-questions/releases)

No releases published

[**Packages**](https://github.com/users/Devinterview-io/packages?repo_name=java-interview-questions)

No packages published

* © 2021 GitHub, Inc.
* [Terms](https://docs.github.com/en/github/site-policy/github-terms-of-service)
* [Privacy](https://docs.github.com/en/github/site-policy/github-privacy-statement)
* [Security](https://github.com/security)
* [Status](https://www.githubstatus.com/)
* [Docs](https://docs.github.com/)
* [Contact GitHub](https://support.github.com/)
* [Pricing](https://github.com/pricing)
* [API](https://docs.github.com/)
* [Training](https://services.github.com/)
* [Blog](https://github.blog/)
* [About](https://github.com/about)