JAVA INTERVIEW QUESTIONS

Most asked Interview Questions

BY : CODE OF GEEKS

CODE OF GEEKS

Java Interview Questions

This e-book contains **most frequently asked questions** on **Java – Programming language**.

Prepared by **CODE OF GEEKS**

In case of bug-reporting, please mail us at

support@codeofgeeks.com

SET 1 – Q1 – Q5

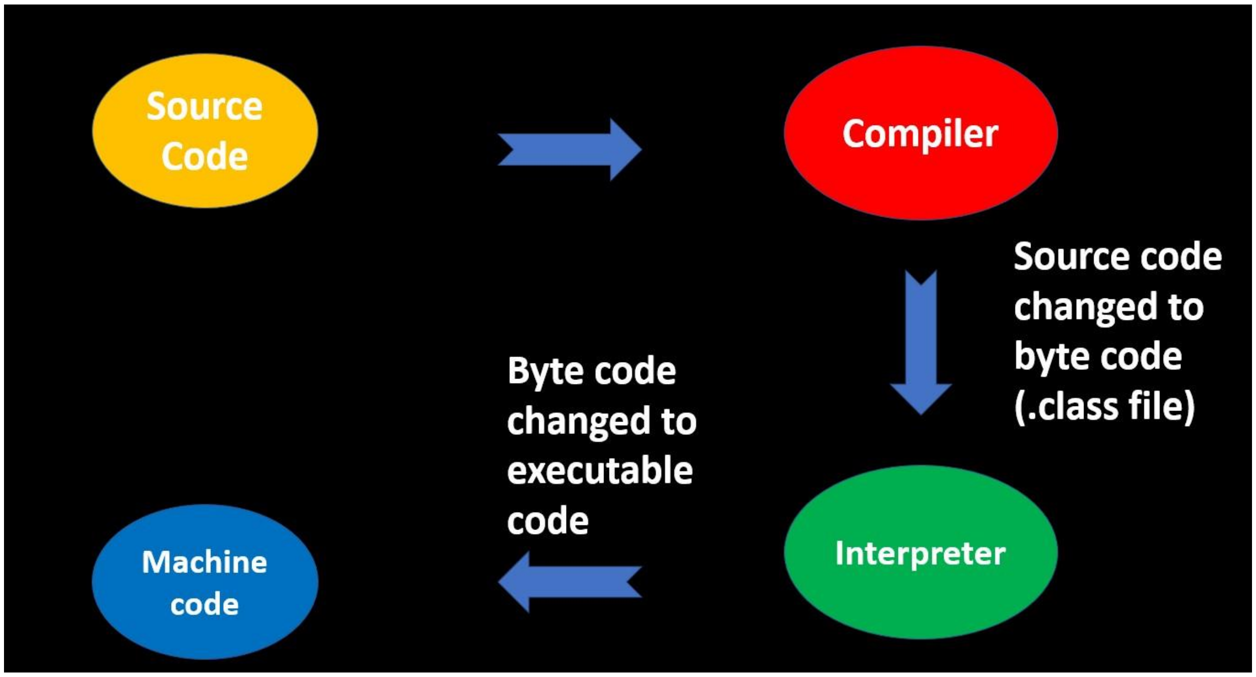
Q. What do you know about Java Programming Language ?
A. Java is a **general purpose**, **object oriented** programming language, designed by **James Gosling** in 1991. First version of Java was launched on **May 23, 1995**. Earlier, Java was named as **Oak**.

Q. Tell me some important features of Java.
A. Features of JAVA :

1. Java is class based **object oriented**, compiled & **Interpreted** programming language.
2. Platform Independent & **Portable**.
3. Java uses the concept of **Multithreading.**
4. Java is **dynamic** – as it can multiple programs and applications dynamically. Ex- Libraries.
5. Java is **Robust** & Secure.

Q. What are Identifiers in Java ?
A. Identifiers are the name given to a class, method, or a variable. Identifiers can start from alphabets, $ sign, but cannot start with numbers or any other sign, **except underscore**.

Q. Explain the procedure of compilation of a Java Code ?
A. For compiling a source code, Java uses both compiler as well as interpreter. Java program takes place at two stages : **Compiler & Interpreter**. Compiler changes a source code to byte code (.class file) and Interpreter changes byte code to machine code.Following is the procedure of compiling a Java Code :


Source Code
Compiler
Source code changed to byte code (.class file)
Interpreter
Byte code changed to executable code
Machine code

Q. What are access modifiers in Java ?
A. In Java, Access modifiers, specifies the accessibility or scope of a field, method, constructor, or a class.

SET 2 – Q6 – Q10

Q. Name some access modifiers in Java ?
A. There are **four access modifiers** in Java Programming Language :

1. public
2. protected
3. default.
4. private.

Q. Name some non access modifiers ?
A. ” **final, abstract, strictfp, synchronized, volatile** ” are non access modifiers.

Q. Explain me the role of JVM.
A. JVM stands for **Java Virtual Machine**. JVM is a software that provides run time environment in which java byte code can be executed.

Q. Explain JRE ?
A. JRE is **Java Runtime Environment**. It is the implementation of JVM.

Q. What is JDK ?
A. JDK is **Java Development kit**. It is the collection of programming tools, JRE, JVM.

SET 3 – Q11 – Q15

Q. What is Applet ?
A. It is a small application written in Java or any other language, that compiles in **Byte code**. It is basically a small internet based program embedded inside a pattern.

Q. What is a Class ?
A. A Class is a specification or a **template** of an object. Class contains variables and methods.

Q. What is an Object ?
A. Object is an instance (dynamic memory allocation) of a class. The process of creating objects out of class is known as instantiation. Object has a state, behaviour and an identity.

Q. Explain OOPS methodology ?
A. The entire OOPS methodology has been derived from a single root called object. In OOPS, all programs involve creation of classes and object. **Four important features** of OOPS are :

1. Encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism

Q. What is Procedural oriented programming ?
A. It is the methodology where programming is done using procedures & functions.

SET 4 – Q16 – Q20

Q. Is Java an object oriented or object based programming language ?
A. Object Based programming language **does not support** all features of OOPs like Polymorphism and Inheritance. Java is an object oriented programming language. Some of object based programming languages are JavaScript, Visual Basics.

Q. Explain Encapsulation with example ?
A. Encapsulation is a mechanism where the data and the code that act on the data will **bind** together.
Example : Class is an example of encapsulation. As class is the collection of data members and member functions, hence class binds them together.

Q. Explain Abstraction with example ?
A. Abstraction refers to the act of **hiding** the background details and providing only necessary details to the user.
Example : Let us the example of a car. While driving car, driver(user) only knows basic operations like gearing system, brakes, but is totally unaware about the internal happenings and processes. This is abstraction.

Q. Explain Polymorphism with example ?
A. Polymorphism represents the **ability** to assume several different forms of a program.
Example : **Overloading** of methods is an example of Polymorphism. Other examples include, abstract classes and interfaces.

Q. How we achieve the concept of polymorphism in Java?
A. Inheritance, Overloading and Overriding are used to achieve Polymorphism in java.

SET 5 – Q21 – Q25

Q. Explain the different forms of Polymorphism.
A. Polymorphism is divided into two forms :

1. Compile time Polymorphism
2. Run time Polymorphism
   Compile time polymorphism is method overloading. Runtime time polymorphism is done using inheritance and interface.

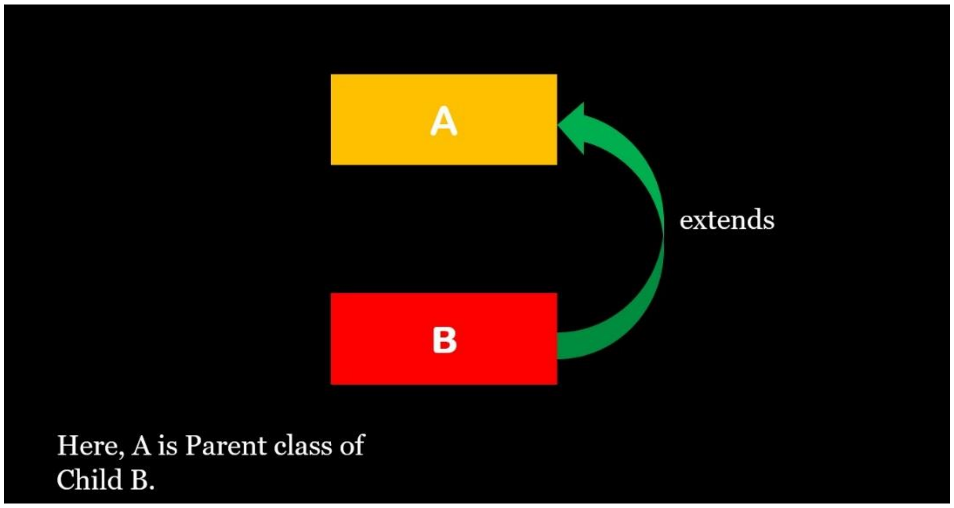
Q. Explain Inheritance with example ?
A. Inheritance refers to the process in which child class inherits the properties of its parent class.
A class that is inherited is called a **superclass**. The class that does the inheriting is called a subclass.
Inheritance is done by using the keyword extends.

Example : Suppose we are having two classes A and B,
class B extends A -> Child class B, is inheriting the property of its parent class.

Q. What are the types of Inheritance shown by Java ?
A. There are four types of Inheritance in Java :

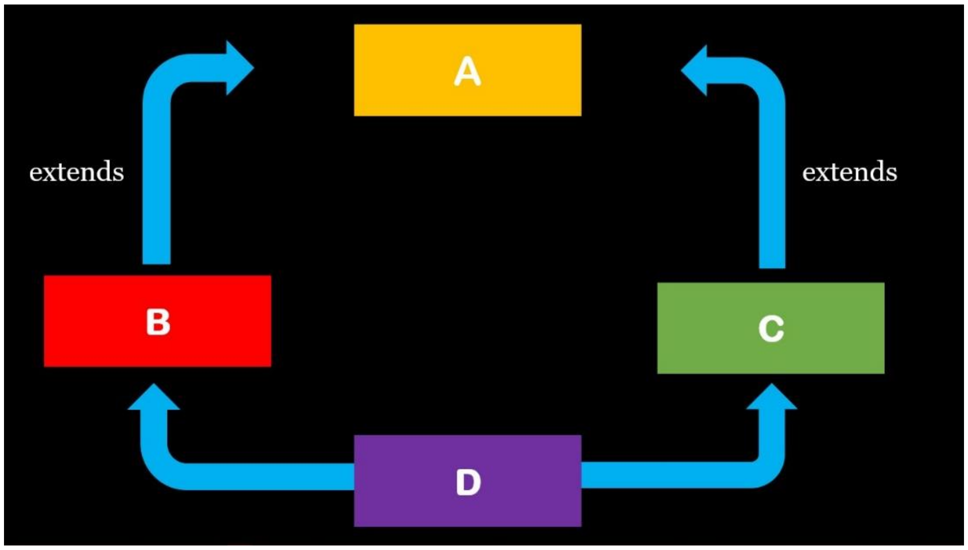
1. Single Inheritance
2. Multilevel Inheritance
3. Hybrid Inheritance
4. Hierarchical Inheritance

Q. Explain different types of Inheritance ?
A. **Single Inheritance** : It is a type of inheritance, where a single child class inherits the property of its parent class.

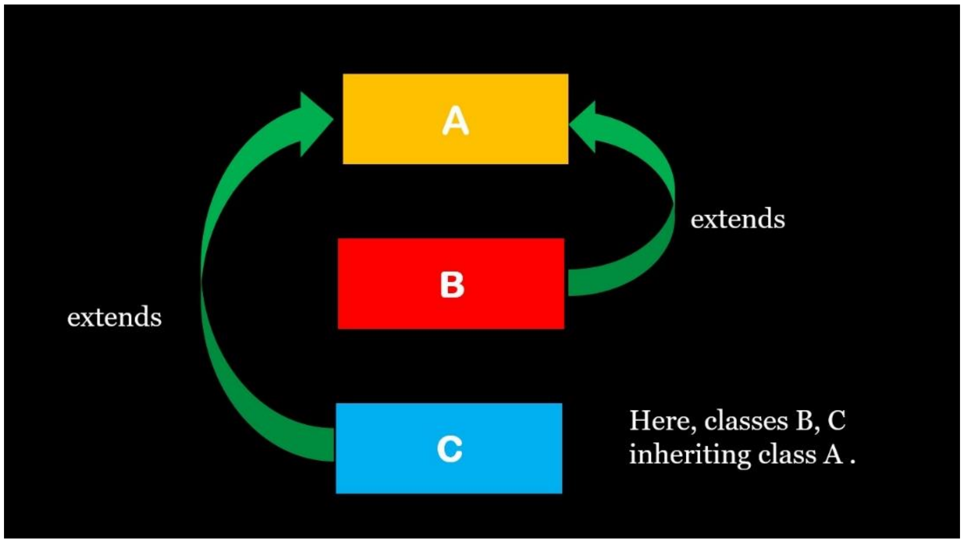


Multilevel Inheritance : A class inherits properties from a class which again has inherits properties.

Hybrid Inheritance : A hybrid inheritance is a combination of more than one types of inheritance.



Hierarchical Interitance : In this type of Inheritance, more than one classes are derived from a single base class or parent class.



Q. Why Java Doesn't Support multiple Inheritance ?
A. This is due to the ambiguity problem.

SET 6 – Q26 – Q30

Q. What happens if super class and sub class having same field name ?
A. Super class fields will be hidden, meanwhile fields of Subclass will be visible. However, you can also access super class fields with super keyword.

Q. Can a children class inherit a constructor of its parent class ?
A. No.

Q. What is Method Overloading ?
A. Method Overloading means to have two or more methods with **same name** in the same class with different arguments. A method can be overloaded in the same class or in a subclass.

Q. What is Method Overriding ?
A. Method overriding occurs when sub class declares a method that has the same type arguments as a method declared by one of its superclass.

Q. Can we override a method marked as final ?
A. **No**, we can not override a method marked with final.

SET 7 – Q31 – Q35

Q. Can we override a method marked as static ?
A. No, we can not override a method marked with static.

Q. Is it possible to override the main method ?
A. No, main is a static method. A static method **can't** be overridden in Java.

Q. Explain the role of implements keyword ?
A. **implements** keyword is used to developed inheritance between interface and class.

Q. Is String a keyword in java?
A. No. **String is not a keyword in java**. String is a final class in java.lang package which is used to represent the set of characters in java.

Q. Can we overload a static method in Java?
A. Yes.

SET 8 – Q36 – Q40

Q. Can we override a static method in Java?
A. No.

Q. Can a class implement more than one interface in Java?
A. Yes.

Q. Can a class extend more than one class in Java?
A. No.

Q. How we can create object without using new operator ?
A. Using factory methods:
NumberFormat obj = **NumberFormat**. getNumberInstance( );
Here, we are creating NumberFormat object using the factory method **getNumberInstance()**.

Q. What are reference variables ?
A. Reference variable usually points to object. In other words, a reference variable is associated with an object. For example :
A ob = new **A()**;
Here, ob is a reference variable for a class A.
Reference variable are like pointers in C.

SET 9 – Q41 – Q45

Q. Explain Mutable and Immutable object in Java ?
A. Mutable objects are the objects which can be modified. For example : **StringBuffer.**
Immutable objects are the objects which can not be modified. For example : **String.**

Q. Where objects are stored in memory ?
A. Objects are stored in **heap** memory.

Q. What is memory leak ?
A. A Memory Leak is a situation when there are objects present in the heap that are no longer used, still, garbage collector is unable to remove them from memory and, thus they are unnecessarily maintained.

Q. Explain the difference between an object and an instance ?
A. An Object **may or may not** have a class definition.
Example – int arr[] where arr is an array.
An Instance should have a class definition.
Example – Solution sol = new Solution(); Here, sol is an instance.

Q. Explain the concept of Singleton Class ?
A. In object-oriented programming, a singleton class is a class that can have only one object at a time.

SET 10 – Q46 – Q50

Q. What is the role of super keyword ?
A. **Super** keyword is used to access the method or member variables from the superclass. If a method overrides one of the methods in its superclass, the method can invoke the overridden method through the use of the super keyword.

Q. What is an Interface ?
A. An **Interface** in the Java programming language is an abstract type that is used to specify a behavior that classes must implement.
Defining an Interface :

```
access interface name {
return-type method-name1();
return-type method-name2();
type final var1 = value;
}
```

An Interface cannot extend anything but another interfaces. Interfaces can not be final.

Q. Can we instantiate an interface ?
A. No, You can't **instantiate** an interface directly, but you can instantiate a class that implements an interface.

Q. What is an abstract class ?
A. Abstract classes are the classes that usually contain **one or more** abstract methods. An abstract method is a method that is declared, but contains no implementation. Whenever abstract keyword is used in front of a class, then it becomes an abstract class.

If even a single method is abstract, the whole class must be declared abstract.

Q. Can we instantiate an abstract class ?
A. No.

SET 11 – Q51 – Q55

Q. Can a class be marked with abstract & final ?
A. No, not possible.

Q. When you declare a method as abstract, can other non abstract methods access it?
A. Yes.

Q. Can there be an abstract class with no abstract methods in it ?
A. Yes.

Q. Explain Constructor ?
A. Constructor is defined as a **special method** that is used for initializing the objects of a class. Constructor has same name as its class. Constructor does not have any return type.

Q. How are this() and super() used with constructors ?
A. Constructors use **this** to refer to another constructor in the same class with different parameters. Constructors use **super** keyword to invoke the superclass's constructor.

SET 12 – Q56 – Q60

Q. What is final modifier?
A. Whenever a variable is marked with final, it means that its value **can not be changed**.

Q. What is static block?
A. Static block is a block of code, which is executed **exactly once** i.e when the class is first loaded into JVM. Before main method the static block executes.
Syntax :
static
{
System.out.println(" This is a static block.");
}

Q. What are Static variables ?
A. It is a variable which belongs to the class and **initialized only once** at the start of the execution.

Q. A static method can only call other static methods. True or False ?
A. True.

Q. A static method can reference to the current object using keywords super or this. True or False ?
A. False.

SET 13 – Q61 – Q65

Q. What are instance variables ?
A. Instance variables are the variables used within a class but **outside** a method. In this, each instance of class, contains its own copies of these variables.

Q. What if we include a return type to a constructor ?
A. No error, as it will be considered as a method.

Q. Can a method name be same as class name ?
A. Yes.

Q. What is Constructor Overloading ?
A. **Constructor overloading** is the process of defining more than one constructors, each one having different parameters.

Q. Do we have destructors in java ?
A. No. Same job is done through Garbage Collection.

SET 14 – Q66 – Q70

Q. Explain the role of garbage collector ?
A. When no reference to an object exists, that object is needed to be no longer needed, and the memory occupied by the object can be reclaimed. There is no need to explicitly destroying the objects. This is the whole process of garbage collection in java.

Q. What are packages in Java ?
A. **Package** in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces. It is achieved through package keyword.

Q. What is S.C.P ?
A. String Constant Pool is the memory space in heap memory specially allocated to store the string objects created using string literals. In String Constant Pool, there will be no two string objects having the same content.

Q. Which is the final class in these three classes – String, StringBuffer and StringBuilder?
A. All three are final.

Q. Which one will you prefer among == and equals() method to compare two string objects?
A. equals() method because it compares two string objects based on their content. If you use == operator, it checks only references of two objects are equal or not.

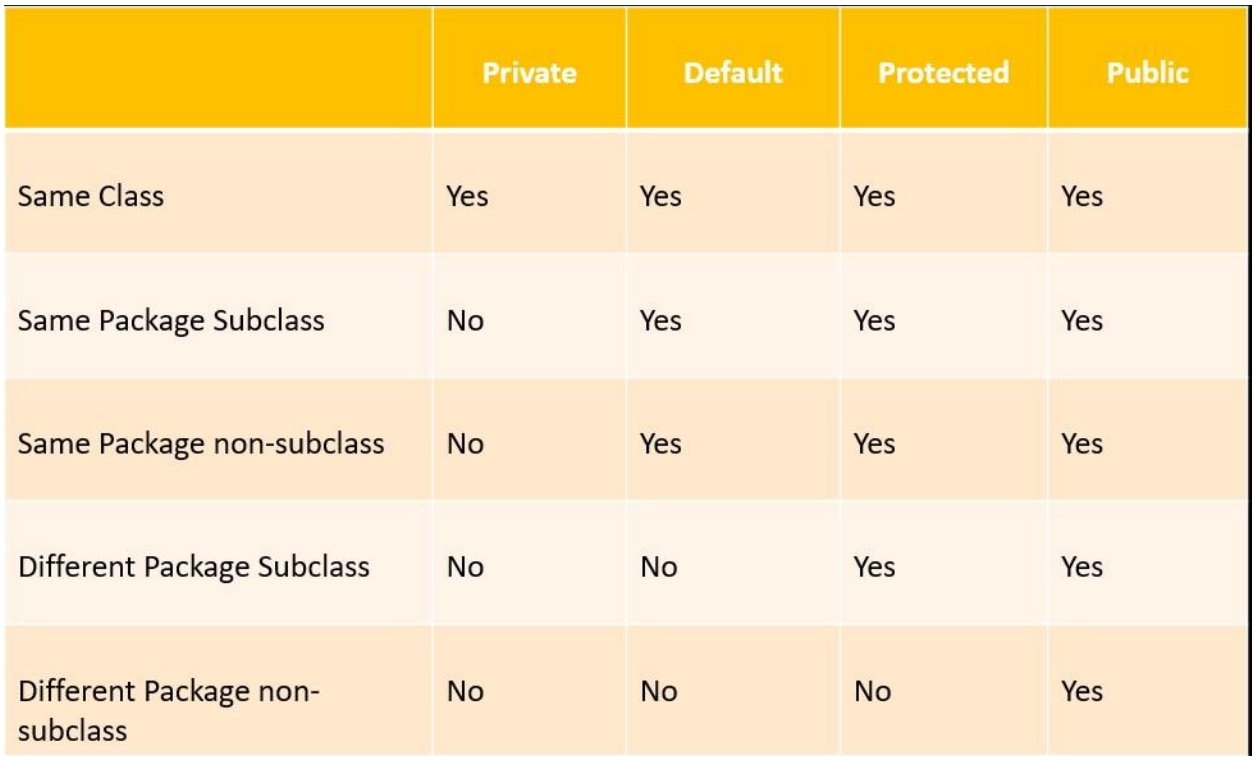
SET 15 – Q71 – Q75

Q. Which class will you recommend among String, StringBuffer and StringBuilder classes if I want mutable and thread safe objects?
A. StringBuffer.

Q. What is the similarity and difference between String and StringBuffer class?
A. The main similarity between **String** and **StringBuffer** class is that both are thread safe. The main difference between them is that String objects are immutable where as StringBuffer objects are mutable.

Q. What is the similarity and difference between StringBuffer and StringBuilder class ?
A. The main similarity between **StringBuffer** and **StringBuilder** class is that both produces mutable string objects. The main difference between them is that StringBuffer class is thread safe where as StringBuilder class is not thread safe.

Q. Explain Class Member Access ?
A.



	Private	Default	Protected	Public
Same Class	Yes	Yes	Yes	Yes
Same Package Subclass	No	Yes	Yes	Yes
Same Package non-subclass	No	Yes	Yes	Yes
Different Package Subclass	No	No	Yes	Yes
Different Package non-subclass	No	No	No	Yes

Q. What is an exception ?
A. An exception is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.

SET 16 – Q76 – Q80

Q. What is an error ?
A. An Error indicates that a non-recoverable condition has occurred that should not be caught.

Q. Which is superclass of Exception ?
A. **Throwable** Class, is the parent class of all exception related classes.

Q. How exceptions can be handled in Java ?
A. Using **try**, **catch**, **throw**, **throws**.

Q. Explain the role of finally block ?
A. It creates a block of code, that will be executed after a try/catch block or before. It will be executed everytime, no matter whether an exception is thrown or not.

Q. What are the types of Exceptions ?
A. **Checked** and **Unchecked**.

SET 17 – Q81 – Q85

Q. Can we have the try block without catch block ?
A. Yes, we can have the try block without catch block, but finally block should follow the try block.

Q. What is a Thread ?
A. A Thread refers to the smallest unit of processing.

Q. How can you gain the concept of multithreading in Java ?
A. In either of two ways :

1. Extending Thread class
2. Implementing Runnable interface.

Q. Explain some methods that are used in Multithreading ?
A.

Method	Meaning
getName()	obtains a thread name.
getPriority()	obtains a thread priority.
isAlive()	determines if thread is still running.
join()	waits for a thread to terminate.
run()	Entry point for a thread.
sleep()	Suspends a thread for a period of time.
start()	Starts a thread by calling run method.

Q. Explain Thread Synchronization ?
A. When two or more threads needs access to a shared resource, they need some way to ensure that the resource will be used by only one thread at a time. This is Synchronization.
It is achieved through synchronized keyword in Java.

SET 18 – Q86 – Q90

Q. What is the difference between yield() and sleep()?
A. yield() allows the current thread to release its lock from the object and scheduler gives the lock of the object to the other thread with same priority. sleep() allows the thread to go to sleep state for x milliseconds. When a thread goes into sleep state it doesn’t release the lock.

Q. What is the difference between wait() and sleep()?
A. wait() is a method of Object class. sleep() is a method of Object class. sleep() allows the thread to go to sleep state for x milliseconds. When a thread goes into sleep state it doesn’t release the lock. wait() allows thread to release the lock and goes to suspended state. The thread is only active when a notify() or notifyAll() method is called for the same object.

Q. What is difference between notify() and notifyAll()?
A. **notify( )** wakes up the first thread that called wait( ) on the same object. **notifyAll( )** wakes up all the threads that called wait( ) on the same object. The highest priority thread will run first.

Q. What is volatile keyword ?
A. **Volatile** keyword is used to modify the value of a variable by different threads. It is also used to make classes thread safe.

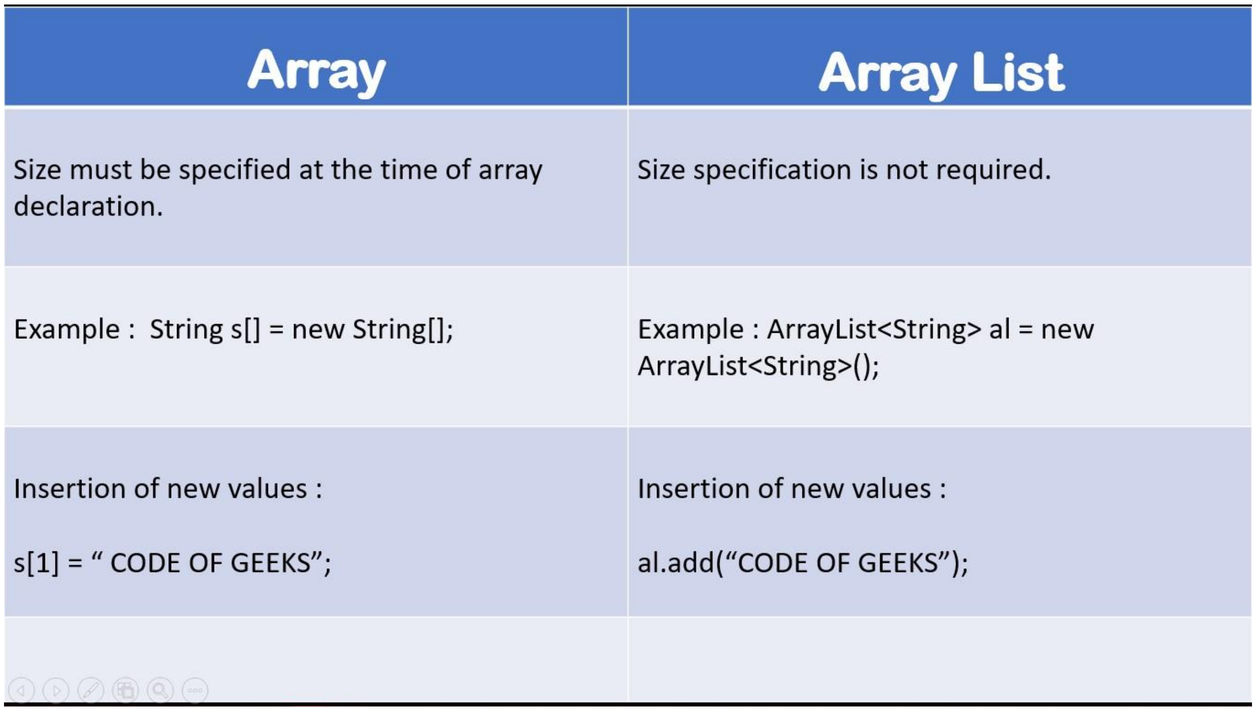
Q. What are the daemon threads?
A. **Daemon threads** are special threads that run in the background, however, these not used to run the application code generally.

SET 19 – Q91 – Q95

Q. How many locks does an object have?
A. Each object has only one lock.

Q. What are Collections in Java ?
A. The Collection in Java is a framework that provides an architecture to store and manipulate the group of objects. Java Collections can achieve all the operations like searching, sorting, insertion, manipulation, and deletion.

Q. Differentiate b/w Array & ArrayList in Java ?
A.



Array	Array List
Size must be specified at the time of array declaration.	Size specification is not required.
Example : String s[] = new String[];	Example : ArrayList<String> al = new ArrayList<String>();
Insertion of new values : s[1] = “ CODE OF GEEKS”;	Insertion of new values : al.add(“CODE OF GEEKS”);

Q. Explain checked & unchecked exceptions ?
A. Checked Exception : Exceptions are checked by the compiler at the time of compilation, are checked exceptions.
Example – ClassNotFound Exception

Unchecked Exception : These exceptions are not checked during the compile time by the compiler.
Example – ArrayIndexOutOfBounds Exception

Q. What is the role of finalize() ?
A. finalize() method is a protected and non-static method of java.lang.Object class. This method is used to perform some final operations or clean up operations on an object before it is removed from the memory.

So, these are most frequently asked interview questions on JAVA.
We, at CODE OF GEEKS, wish you all the best for your upcoming future.
For any assistance, feel free to mail us at support@codeofgeeks.com

Some Important Links

100+ | Java Interview Questions

[illegible]

Here are the Most Important Interview Questions on Java!

Prepare this Question, then you are ready for Companies like Accenture, Capgemini, TCS, Mindtree, Etc.

1. What is OOPs and OOPs Concepts?

Ans :Object-Oriented Programming System (OOps) is model in programming languages based on concepts of objects and classes.

OOps concepts are:

- Abstraction
- Inheritance
- Polymorphism
- Encapsulation

Q2. Explain Abstraction In the real world?

Ans: Abstraction refers to the act of representing essential features without including the background details(complexities).

In simple terms hiding complexities and displaying necessary details only.

In real world take an example of online banking transaction where end user are shown only User interface of Payments gateways and complexities that is how actual transaction and operations is been performed are kept hidden from end user. This what abstraction is.

Q3. Code to Achieve Abstraction

```
/*
code for Abstraction */
abstract class OnlineBank{
abstract int displaybalance();
}
class SavingAccount extends OnlineBank{
int displaybalance(){
return 50000;
}
class CurrentAccount extends Onlinebank{
int displaybalance(){
return 150000;
}
class main{
OnlineBank b =new SavingAccount();
System.out.println("Balance in Saving
Account="+b.displaybalance()+"Rs");
OnlineBank b =new CurrentgAccount();
System.out.println("Balance in Current
Account="+b.displaybalance()+"Rs");
}
}
```

Output:

```
Balance in Saving Account= 50000 Rs

Balance in Current Account= 150000 Rs
```

Q4. Can we declare Abstract method without using abstract class?

Ans:No, if there is a abstract method they must be declare in abstract class.

Q5. Difference between Abstract class and Interface?

Abstract Class	Interface
Abstract class can only have a method body.	Interface can only have abstract methods.
Abstract class can have constructors	Interface cannot have constructors.
Members of Abstract class can be public private protected	Members of Interface are public by default.
Abstract keyword is used to declare class as Abstract.	Interface keyword is used to declare an Interface.

Q6. What is inheritance, list its Types?

Ans: Inheritance is the process by which one class acquires properties of other class. or mechanism of deriving new class from old class is called as Inheritance.

Types of Inheritance-

- Single
- Multilevel
- Hierarchical
- Hybrid
- Multiple

Q7. Is Multiple Inheritance Supported in Java if No Why, And if Yes How?

No, Java not supports Multiple Inheritance. Because it leads to the diamond problem.

Since multiple inheritance is not supported in java classes, but with the help of interface we can achieve multiple inheritance.

Q8. Code to achieve Multiple Inheritance?

```
/* Code To Achieve Multiple Inheritance */
interface A{
void showa();
}
interface B{
void showb();
}
class C implements A,B{
public void showa()
{
System.out.println("I Am In A");
}
public void showb()
{
System.out.println("I Am In B");
}
}
class multiple{
```

```
Public static void main(String args[]){
C c= new C();
c.showa();
c.showb();
}
}
```

Output:

```
I Am In A
I Am In B
```

Q9. What is Polymorphism and its types?

Ans: Polymorphism in java is a concept by which we can perform single action by different means/ways. Polymorphism is the Greek word in which poly means many and morph means form hence many forms.

There Are two Types of Polymorphism:-

1. Compile time- Achieved by Method Overloading.
2. Run Time- Achieved by Method Overriding.

Q10. Code to implement Method Overloading?

```
/*
Java Program to implement Method Overloading */
    class Area{
int area(int side){
return(side*side);
}
double area(double circle){
```

```
return(3.14*radius*radius);
}
int area(int len,int wid){
return(len*wid);
}
class Shape{
public static void main(String args[]){
Area s=new Area();
System.out.println("Area of Square:"+s.area(5));
System.out.println("Area of circle:"+s.area(10.2));
System.out.println("Area of Rectangle:"+s.area(10,2));
}
}
```

Output:

```
Area of Square: 25
Area of circle: 326.685
Area of Square: 20
```

Q11. Code to implement Method Overriding?

```
/*Code to implement Method Overiding */
class A{
public void show(){
System.out.println("I am in A");
}
}
class B extends A
{
public void show()
{
super.show();
System.out.println("I am in B");
}
}
class Inherits{
public static void main(String args[]){
B b1= new B();
b1.show();
}
}
```

Output:

```
I am in A
I am in B
```

Q12. Difference between Method Overloading and Method Overriding?

Method Overloading	Method Overriding
In Method Overloading,Methods of same class shares the same name but each method must have different parameter.	In Method Overriding sub class have the same methods with same name and exactly same number and types of parameters

Method Overloading	Method Overriding
It is a compile time polymorphism.	It is a run time polymorphism.
May or may require Inheritance to implement method overloading.	Always requires Inheritance to implement method overriding.

Q13. What is upcasting and downcasting in java?

i. When a reference variable of parent class refers to the object of child class,it is known as Upcasting.Converting a subclass to super class type.

reference variable of parent class——>object of child class

ii.Converting Super class type to sub class is know as downcasting.

Q14. Write a Code to Explain Upcasting in Java?

```
/*
Java Program to demonstrate Upcasting in Java */

class Parent{
void example(){
System.out.println("In a Method of parent class");
}
}
public class Base extends Parent{
void example(){
System.out.println("In a Method of parent class");
}
public static void main(String args[]){
Parent p=(Parent) new Base();
p.example();
}
}
```

Q15. Write a code to Explain Downcasting in Java?

```
/*
Java Program to demonstrate Downcasting in Java */

class Parent{
void example(){
System.out.println("In a Method of parent class");
}
}
public class Base extends Parent{
void example(){
System.out.println("In a Method of parent class");
}
public static void main(String args[]){
Parent P= new Base()
Sub sub=(Sub) P ;
sub.example();
}
}
```

Q16. Differentiate Between Constructor and Methods?

Constructor	Methods
Constructors are the special methods whose tasks is to initialize object of its class.	Method are the set of instruction that are called at any given point during our program execution.
Constructors don’ t have return type.	It has return type.
Constructor cannot be declared as final.	Method can be declared as final.
The Name of constructor must be same as class name	Method Name can be any.

Q17. What is the Static Variable in Java?

Ans: Static Variables are the variables, we assign memory only once at the time of class loading. Using a Static variable it makes your program efficient as it saves memory.

Q18. Demonstrate the use of Static Variables with the help of Java Program?

```
/*
Java Static Variable */
class StaticExample{
int EmployeeNo;
String EmployeeName;
static string department="Testing";
StaticExample(int eno,String n)
EmployeeNo=eno;
EmployeeName=n;
}
void display(){
System.out.println(EmployeeNo+""+EmployeeName+""+department);
}
public static void main(String args[]){
StaticExample S1= new StaticExample(1000,Manthan);
StaticExample S2= new StaticExample(1001,Nived);
S1.display();
S2.display();
}
```

Output:

```
1000 Manthan Testing
1001 Nived      Testing
```

Q19. What is Static Method?

Ans: Static methods can be called without using objects.Static methods can only call other static method and static variables defined in the class.

Syntax: static return_type method_name(parameter_list)

Q20. Code for to demonstrate the use of Static Method

```
/*
Java Program To demonstrate the use of Static method */

class example
{
```

```
static void show(){
System.out.println("I am in Static Method");
}
public static void main(String args[]){
System.out.println(I am in main);
show();
}
}
```

Output:

```
I am in main
I am in Static Meethod
```

Q21. Difference Between Static Method and Instance Method?

Static Method	Instance Method
To call static method we don’t require to create Objects.	To call Instance method we need to create object for it.
They are efficient as it save space.	They are not efficient in terms of memory, dedicated memory is allocated to each instance method.
Example: public static int{ square(int n){ return n*n;}}	Example: public int square(int n){ return n*n; }

Q22. Can we make Constructor Static ?

Ans: We know that Constructors invokes as soon as we create objects. And Static (method, block, or variable) belongs to class not the object.

Hence that is why there no sense in making Constructor as static. If you try to make Constructor Static Compiler will throw an error.

Q23. Can we Execute Program without Main Method?

Ans: Yes,We can execute Program without main method also.

Q24. Can we Make Abstract Method Static?

Ans: If we make the Abstract method as static in java then they became part of the class. Therefore we can call them from anywhere, then what is the use of abstraction, Hence this is not allowed in java.

Q25. Explain the Significance of this Keyword in Java?

Ans:Many times it is necessary to refers to its own object in a method or constructor.To allow this Java defines 'this' keyword.

this is used inside the method or constructor to refers to its own object.

Q26. Demonstrate the use of this keyword with Example?

```
/* Java Program To Demonstrate The use of This Keyword */
class Data{
int a ,b;
void setdata(){
a=10;
b=20;
void setdata(int a,int b){
this.a=a;
yhis.b=b;
}
void showdata(){
```

```
System.out.println("A="+a);
System.out.println("B="+b);
}
}
class Mydata{
public static void main(String arga[]){
Data d1= new Data();
d1.setdata(100,200);
d1.showdata();
}
}
```

Output:

```
A=100
B=200
```

Q27. What is Constructor Chaining and how it is performed?

Ans: Constructor Chaining is the process of calling one constructor from another constructor of the class with the help of the current class object.

This keyword is used to perform constructor chaining within same class.

Q28. Code To Demonstrate Constructor Chaining?

```
/* Java Program for Constructor Chaining */
class student{
int rollno,age;
String name;
public student(int age){
```

```
this.age=age;
}
public student(int rollno,int age){
this(age);
this.rollno=rollno;
}
public student(int rollno,String name){
this(rollno,age);
this.name=name;
}
public static void main(String args[]){
student std=new student(100,17,"Shyam");
System.out.println("RollNo"+std.rollno+"Age:+age +"Name:"+name);
}
}
```

Output:

```
Rollno:100 Age:17 Name:Shyam
```

Q29. Explain the significance of Super Keyword in Java?

Ans: The Keyword "Super" is a special keyword in java use for inheritance. whenever the subclass needs to refer to its immediate superclass, the "super" keyword is used.

There are two main uses of Super keyword:

1. Accessing members from super class that has been hidden by members of the subclass.
2. Calling super class constructor.

Q30. Code for Super Keyword?

```
/* Java Program to Implement Super Keyword */
Class Shape{
String colour="Black";
Shape(){System.out.println("Shape is Created");}
}
class Square extends Shape{
String colour="Red";
void colourprint(){
System.out.println(colour);
System.out.println(super.colour);
}
class test{
public static void main(String args[]){
Square s= new Square();
s.colourprint();
}

}
```

```
Black
Red
```

Q31 Differentiate Between this and Super Keyword?

This Keyword	Super Keyword
This Keyword is to refer current	Super Keyword is to refer to Parent

This Keyword	Super Keyword
class context.	class context.
Used to invoke current class method.	Used to invoke immediate parent class method.

Q32. What is Object Cloning?

Ans:It is a process of creating exact copy of an object.

Q34. Explain the Significance of Final Keyword?

Ans: Following Are the Uses of Final Keyword

1. **One use of the final keyword is to make constant variables**,i.e we cannot change the value of the variable.
2. **It is used for preventing method overriding**,i.e is method is declared as the final method cannot be overridden.
3. **Used to prevent Inheritance**,i.e we cannot extends the parent class

Q35. Write a code to Explain the Final Keyword?

```
/*
Java Program To demonstrate Use of Final Keyword */
class A{
final void show(){System.out.printlb("I am in A");}
}
class B extends A{
void show(){System.out.printlb("I am in B");}
}
class Example(){
```

```
public static void main(String args[])
B b1 = new B();
b1.show();
}
```

```
show() in B cannot override show() in A; overridden method is final
void show()
1 error
```

Core Java|String Related Questions

Q36. What is String in Java? Is String is a datatype?

Ans: String is a class in java, In Java.lang package. It is not a primitive datatype like: int, float, and double.

String is an important class in java which is used to perform various operations like:

Split, trim, String to uppercase and lowercase and vice versa, concatenation and various other complex operation converting other types to string type, etc.

Q37. What are the different ways of creating a String Object?

Ans: There are two ways of creating String in Java:

1. Creating String using new Keyword.
    1. String str= new String("Noteshacker');
2. Simple declaration using double quotes
    1. String str="NotesHacker";

When we create a String using double quotes, JVM looks in the String pool to find if there is any other String with the same value/name. If same string found it refers to that String object else create a new object in the String pool.

When we use the new operator, Java Virtual Machine will create the new String Object, but initially it is not stored in string pool.

Q38. String is Immutable in Java?

Ans: First of all Immutable means: Once something is created it cannot be changed.

Yes, String is Immutable that is once we create its object, its value cannot be changed.

Q39: What is the Difference Between String and String Buffer?

String	**StringBuffer**
String is a major Class.	StringBuffer is a peer class of String.
The length of the String is fixed.	String Length is flexible.
The contents of objects cannot be modified.	The contents of objects can be modified.
Object can be created by assigning String Constants enclosed in double quotes.	Objects can be created by calling the constructor of the StringBuffer class using new.
Ex: String s=” NotesHacker “;	Ex: StringBuffer s= new StringBuffer(“NotesHacker”);

Q40. Differentiate StringBuilder and StringBuffer?

StringBuilder	**StringBuffer**
StringBuilder are not thread safe.	Stringbuffer is thread safe.
It is not Synchronized	It is synchronized.
They are faster.	StringBuffer is slower as thread is safe.

Q41. How many Objects will be created in the following code?

String s1="NotesHacker";

String s2=new String("NotesHacker");

In the above example Two object will be created,Object created using string literal s1 is stored in string constant pool.

Object created using new operator is stored in the heap section s2.

Q42. Java Program to Check Whether String is Palindrome or Not?

Ans: It is one of the favorite Question of Interview and mostly asked:

Click here to view Solution

Q43 Which class is final among three classes-String, StringBuffer and StringBuilder?

Ans: All the three classes are final,this type of question are asked to confuse.

Q44. What is String intern() method ?

Ans: String Interning is a process of creating only one unique copy of distinct string value that must be immutable.

For example: if a name lets say-Noteshacker appears 10 times by interning you must ensure that only one Noteshacker is allocated with memory.

Q45. Code To demonstrate String intern() method?

```
public class Intern{
public static void main(String args[]){
String s1=new String("Noteshacker");
String s2="Noteshacker";
String s3=s1.intern();

System.out.println(s2==s3);// return true as reference variable are
pointing to sameinstance
}
}
```

Output: true

Q46: What toString Method in Java used for?

Ans: Java.lang.Interger.toString method is used to return string objects representing this integer value.

Q47. Demonstrate the use of toString method using Java Program?

```
public class test{
public static void main(String args[]){
int s=5;
System.out.println(s.toString());
}
}
```

Output: 5

Core Java |Array Related Questions

Q49. What is Array in java?

Ans:An Array is a Structure that is collection of similar types of elements.An Array is a homogeneous data type where it can hold only elements of one data type.

Q50. Can we change the Size of Array at Runtime?

Ans: no we cannot change the size of array at runtime.

Q51. Can we declare an array without assigning the size of an array?

Ans: No we cannot declare array without assigning it size.if we try to do so it will throw compile time error

Q52. What is the default value of Arrays?

When new Array is initialized the default value is as follows:

- For byte,short,int,long-default value is 0.
- Float,double-default value is 0.0.
- Boolean-default value is false.
- Object -default value is null.

Q53. What is Jagged array?

Ans: Jagged Array are multidimensional Array,i.e Array having different length.

Q54. Where is the memory allocated for java?

Ans: In Java,Memory for arrays is always allocated on heap as array in java are considered as object.

Q55. differentiate between Array and Arraylist?

Array	**ArrayList**
Size of Array are fixed(Static) which means we cannot change the size of array.	Size of ArrayList is not fixed(dynamic),size of ArrayList changes when new elements are added to it.
We can store both primitive as well as objects in Array.	We can only store objects in ArrayList.

Q56. Demonstrate the use of ArrayList with Java Example?

```
/*java ArrayList Example */
import java.util.*;
public class ArrayExample{
public static void main(String args[]){
ArrayList<Integer> list= new ArrayList<>();
list.add(Integer.valueOf(100)).
list.add(200);
list.add(300);
System.out.println("List:");
for(Integer i:list){
System.out.println(i);
}
}
}
```

Collection Class related Questions

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

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.

58. Why Collection doesn't extend Cloneable and Serializable interfaces ?

The Collection 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 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.

59 What is an Iterator ?

The Iterator interface provides a number of methods that are able to iterate over any Collection. Each Java Collection contains the iterator method that returns an Iterator instance. Iterators are capable of removing elements from the underlying collection during the iteration.

60 What differences exist between Iterator and ListIterator ?

The differences of these elements are listed below:

• An Iterator can be used to traverse the Set and List collections, while the ListIterator can be used to iterate only over Lists.

• The Iterator can traverse a collection only in 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.

61 What is difference between fail-fast and fail-safe ?

The Iterator's 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, while fail-safe iterator never throws such an exception.

62 How HashMap works in Java ?

A HashMap in Java stores key-value pairs. The HashMap 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 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.

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

In Java, a HashMap uses the hashCode and equals methods to determine the index of the key-value pair and to detect duplicates.

More specifilically, 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 effificiency of the HashMap.

64 What differences exist between HashMap and Hashtable ?

Both the HashMap and Hashtable 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 doesn't 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 of its keys.

• The Hashtable class is considered to be a legacy class.

65 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 can contain primitive or objects, while an ArrayList can contain only objects.

• Arrays have fixed size, while an ArrayList is dynamic.

• An ArrayList 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.

66 What is difference between ArrayList and LinkedList ?

Both the ArrayList and LinkedList 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.

67 What is Comparable and Comparator interface ? List their differences.

Java provides the Comparable interface, which contains only one method, called compareTo. 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 interface, which contains two methods, called compare and equals.

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 fifirst 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.

68 What is Java Priority Queue ?

The PriorityQueue 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 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 is not thread-safe and it requires O(log(n)) time for its enqueing and dequeing operations.

69 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.

70 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 un ordered array takes constant time of O(1).

71 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 fifixed and know a priori, we shall use an Array, instead of an ArrayList.

• 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 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 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.

72 What's the difference between Enumeration and Iterator interfaces ?

Enumeration is twice as fast as compared to an Iterator and uses very less memory. However, the Iterator 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, Iteratorsallow the caller to remove elements from the underlying collection, something which is not possible with Enumerations.

73 What is the difference between HashSet and TreeSet ?

The HashSet 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 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).