**CORE JAVA INTERVIEW QUESTIONS**

**1) What is the difference between class and object?**

**Class** **Object**

1. Template/Blue print of an object. 1. It is an instance of a class.

2. A logical construct. 2. A Physical reality.

3. Class contain variables and methods 3. Object have state and behaviours

4. Ex: 4. Ex: All real world objects like dog,phone,car,train etc.

class Test {

int a;

public void m1(){ }

}

**2) What are the major object oriented concepts in Java?**

**Class**

A class is a plane or a model or template

A class is a way of binding the data and it's associated methods into a single unit.

**Object**

The instance of a class is called as Object.

Instance is nothing but allocating the memory for the data members of the class

here are four different ways to create objects in java:

1. Using new keyword
2. Using Class.forName():
3. Using clone():
4. Using Object Deserialization:
5. Using newIntance() method

#### Using new keyword:

This is the most common way to create an object in java. Almost 99% of objects are created in this way.  
  
MyObject object=new Object();  
  
**Using Class.forName():**

* If we know the name of the class & if it has a public default constructor we can create an object in this way.
* Syntax:
* Myobject obj=(MyObject) class.forName("object").newInstance();

#### Using clone():

* The clone() can be used to create a copy of an existing object.
* Syntax:
* MyObject obj=new MyObject();
* MyObject object=(MyObject )obj.clone();

#### Using Object Deserialization:

* Object deserialization is nothing but creating an object from its serialized form.
* Syntax:
* objectInputStream istream=new objectInputStream(some data);
* MyObject object=(MyObject) instream.readObject();

**Using newInstance() method**  
  
Object obj = DemoClass.class.getClassLoader().loadClass("DemoClass").newInstance();

**Abstraction**

The process of hiding the internal implementation logic and just highlighting only the set of services we are offering is called as Abstraction

The main advantage of Abstraction is to provide the security.

For example when you consider the case of e-mail, complex details such as what happens soon you send an e-mail, the protocol your email server uses are hidden from the user, therefore to send an e-mail you just need to type the content, mention the address of the receiver and click send.

Likewise in Object oriented programming Abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user.

**In Java Abstraction is achieved using Abstract classes, and Interfaces.**

Abstract Class

A class which contains the abstract keyword in its declaration is known as abstract class.

Ex:

abstract class Bike{

Bike(){System.out.println("bike is created");}

abstract void run();

void changeGear(){System.out.println("gear changed");}

}

class Honda extends Bike{

void run(){System.out.println("running safely..");}

}

class TestAbstraction{

public static void main(String args[]){

Bike obj = new Honda();

obj.run();

obj.changeGear();

}

}

**Encapsulation**

Wrapping up the data and it's associated methods into a single unit is called as Encapsulation

The main advantage of encapsulation is to provide the security.

We can create a fully encapsulated class in java by making all the data members of the class private. Now we can use setter and getter methods to set and get the data in it.

The Java Bean class is the example of fully encapsulated class.

Let's see the simple example of encapsulation that has only one field with its setter and getter methods.

//save as Student.java

package com.javatpoint;

public class Student{

private String name;

public String getName(){

return name;

}

public void setName(String name){

this.name=name

}

}

//save as Test.java

package com.javatpoint;

class Test{

public static void main(String[] args){

Student s=new Student();

s.setname("Penchu");

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

}

}

**Inheritance:**

Getting the properties and behaviors from base class to derived class is called as inheritance.

The main advantage of inheritance is code re-usability.

**Polymorphism**

The ability to exist more than one form is called as polymorphism

The main advantage of polymorphism is to provide flexibility.

**3) What are the advantages of Inheritance in Java?**

Re-usability: Inheritance helps derived class to use methods of base class without rewriting them

Extensibility: Extending the base class logic as per business logic of the derived class

Data hiding: Allows base class to keep some private data which can't be altered by the derived class

**4) Explain polymorphism in Java ?**

The ability to exist more than one form is called as polymorphism

The main advantage of polymorphism is to provide flexibility.

There are two types of polymorphism in java

1.Compiletime Polymorphism: The polymorphism exhibited at compile time is called as Compile time polymorphism

Ex: Method overloading and Method hiding

Method overloading: We can define same method name and different parameters in a class is called as Method overloading.

Example:

1. **class** Calculation{
2. **void** sum(**int** a,**int** 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. Calculation obj=**new** Calculation();
7. obj.sum(10,10,10);
8. obj.sum(20,20);
10. }
11. }

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=Calculation)

Output:30

40

2.Runtime Polymorphism: The polymorphism exhibited at run time is called as Run time polymorphism

Ex: Method overriding

Method overriding: We can define same method name and same parameters including order is called as Method overriding.

Example:

1. **class** Bank{
2. **int** getRateOfInterest(){**return** 0;}
3. }
5. **class** SBI **extends** Bank{
6. **int** getRateOfInterest(){**return** 8;}
7. }
9. **class** ICICI **extends** Bank{
10. **int** getRateOfInterest(){**return** 7;}
11. }
12. **class** AXIS **extends** Bank{
13. **int** getRateOfInterest(){**return** 9;}
14. }
16. **class** Test{
17. **public** **static** **void** main(String args[]){
18. SBI s=**new** SBI();
19. ICICI i=**new** ICICI();
20. AXIS a=**new** AXIS();
21. System.out.println("SBI Rate of Interest: "+s.getRateOfInterest());
22. System.out.println("ICICI Rate of Interest: "+i.getRateOfInterest());
23. System.out.println("AXIS Rate of Interest: "+a.getRateOfInterest());
24. }
25. }

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=Test2)

Output:

SBI Rate of Interest: 8

ICICI Rate of Interest: 7

AXIS Rate of Interest: 9

Method hiding: Method hiding similar to method overriding except the difference in method hiding both the methods should be static where as in method overriding both the methods should be non-static.

Example:

class Parent{

public void property(){

System.out.println("Land,Gold,Money.....")

}

public static void marry(){

System.out.println("SubbaLaxmi")

}

}

class child extends Parent{

public static void marry()

{

System.out.println("3sha/9tara")

}

}

class OverridingDemo

public static void main(String args[]){

Parent p=new Child();

p.property();

p.marry();

}

}

**5) Explain the difference between compile time and run time polymorphism in Java ?**

Compile time Polymorphism Run time Polymorphism

1. Method are called at compile time Method are called at run time

2. Method resolution takes care by compiler 2.Method resolution takes care by JVM

3.Ex: Overloading 3.Ex: Overriding

**6) What is the difference between Overloading and Overriding ?**

Overloading Overriding

1. Methods are overloaded during compile 1.Method overriding takes place during runtime

2. All the overloaded methods should be placed in the same class 2.We can override methods in sub class

3. We can overload static methods 3.Static methods can't be overridden

4. Methods are bonded together using static binding. 4. Overridden method are bonded via dynamic bonding based

5. To overload a method, method signature needs to be changed 5.There is no need to change the signature

6. Private and final method can be overloaded. 6. Private and final method can't be overridden

7. Method is relatively fast. 7. Method is relatively slow.

**7). 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 cannot 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.

**8) Can you please explain about constructor?**

Constructor is a special member and it is used to provide initialization properties to an object.

Constructor name and class name should be same.

Constructors are executed exactly at the time of creating an object

Constrictors are classified into two types.

1. Userdefined Constructor

2. Default Constructor

Userdefined Constuctor: User defined Constructors are defined by programmers based on the application requirement.

Default Constructor: Default Constructors are provided by compiler when we are not providing a single constructor to the class

Note: Every java class contains at least one constructor either programmer provided or compiler generated constructor.

Ex:

class Test{

Test(int i){

sop(i);

sop("Test class Constructor");

}

psvm(String args[]){

Test t1=new Test(10);

}}

o/p: 10 Test class Constructor

**9) Can you explain about the default constructor in Java?**

When a constructor is not specified explicitly, java compiler automatically creates a "Default Constructor". When we creates an object instance, default constructor initialize variables with its default values.

**10) Why Java is not supporting multiple inheritances?**

Main features of java are very simple. If multiple inheritances is supported, it creates ambiguity and creates problem during casting, chaining etc.

**11) What is meant by final keyword in Java?**

If final variable is used in front of variable, we can't change the value.

If the variable is used in front of method, it can't be overridden.

If it is used in front of Class, class can't be extended by any other class.

**12) What is meant by static keyword in Java ?**

A static is a member of a class that's not associated with instance. So static members can be accessed without creating an instance of a class.

**13) What is meant by JVM ?**

JVM(Java Virtual Machine) is a run time environment for the compiled java class files. Main function of JVM is to convert byte code(.class file) to machine code and send appropriate commands to underlying machine for execution.

**16) What is the difference between interpreter and compiler in Java?**

Compiler Interpreter:

1. Compiler translates source code to JVM byte code. 1.Executes the byte code by running the program.

2. Compiling happens when programmers invoke compiler after the program is written. 2. Interpretation happens at run time

3. For compilation, use command "javac Employee.java". Compilation of java file will generate class file (ex: Employee. class). 3. Command java Employee, executes the class file (i.e. Employee. class)

**17) Can abstract class implement another interface?**

Yes. It's just a special case of implementation by which subclasses are forced to implement the methods.

**18) Can abstract class extend another abstract class?**

Abstract class: A class which contains the abstract keyword in its declaration is known as abstract class.

Abstract class allow zero or more no of abstract methods and zero or more no of concrete methods

Yes. It is perfectly valid for an abstract class to extend another abstract class.

Ex

abstract class Bike{

abstract void run();

}

class Honda extends Bike{

void run(){System.out.println("running safely..");}

public static void main(String args[]){

Bike obj = new Honda();

obj.run();

}

}

**19) What is an interface?**

Interface looks like class but it is not a class. An interface can have methods and variables just like the class but the methods declared in interface are by default abstract (only method signature, no body). Also, the variables declared in an interface are public, static & final by default.

Use of interfaces: Interfaces are used for abstraction.

Declaration: Interfaces are declared by specifying a keyword “interface”.

E.g.:

interface MyInterface{

/\* All the methods are public abstract by default

\* Note down that these methods are not having body

\*/

public void method1();

public void method2();

}

Note: Class implements interface but an interface extends another interface.

Example:

interface MyInterface

{

public void method1();

public void method2();

}

class Test implements MyInterface

{

public void method1()

{

System.out.println("implementation of method1");

}

public void method2()

{

System.out.println("implementation of method2");

}

public static void main(String arg[]){

MyInterface obj = new Test();

obj. method1();

obj. method2();

}}

Output:

implementation of method1

implementation of method2

**20) Can a interface extend another interface?**

Yes. An interface can extend another interface in Java.

interface Printable{

void print();

}

interface Showable extends Printable{

void show();

}

class Testinterface implements Showable{

public void print(){System.out.println("Hello");}

public void show(){System.out.println("Welcome");}

public static void main(String args[]){

Testinterface obj = new Testinterface();

obj.print();

obj.show();

}

}

**22) Can you explain about Implicit and Explicit type casting?**

Type casting:

The process of converting data from one data type to another data type is called as Type casting.

Implicit casting: The process of converting data from lower data type to upper data type is called as implicit typecasting.

This casting operation should be performed by JVM. For example

Ex: int i = 1; // 4 bytes

double d = i; // 8 bytes

Explicit casting: The process of converting data from higher data type to lower data type is called as explicit typecasting

This casting operation should be performed by the programmer. For example

Ex: double d = 1.0;

int i = (int) d;

**23) Can you explain about makeable interface in Java ?**

Interfaces with no methods are known as Marker interface. Some of the markable interfaces are

java.lang.Cloneable

java.io.Serializable

java.util.EventListener

**24) Explain about variables in java?**

Variable: variable is a named memory location

Variables can be classified into 3 types.

1. Local variables: We can declare any variable inside a method or inside a block or inside a constructor that variables are called as local variables.

Ex: public void m1(){

int x=20;

int y=20;

}

2. Static variables: We can declare the variables with in a class with the static modifier.

The value of the variable is not valid from object to object that variables are called as static variable.

Ex: class Test{

static int x=10;

public static void main(Sting args[])

{

system.out.println(x);

}

}

3. Instance variables: We can declare the variables with in a class directly

The value of the variable valid from object to object that variable are called as non-static variable.

Ex: class Test{

int x=10;

public static void main(Sting args[])

{

Test t1 = new Test();

t1.x=100;

system.out.println(x);

system.out.println(t1);

}

}

**27) What are the advantages of organizing classes and interfaces into a package ?**

1. Determination of a category of a file is simplified.

2. Name space collision is avoided.

3. Access restriction can be applied with the use of packages.

4. Packages provide reusability of code

**28) How to call a garbage collector in java?**

System.gc() or Runtime.getRuntime().gc().

**30) Can you explain about the access modifier in Java?**

Access modifiers specify the access levels of a variable or method. Java access modifiers are public, private, protected, default ,final,native,synchronized,strictfp.

**35) What is the difference between super() and this() in Java ?**

If you want to access methods of the base class from derived class "super" keyword is used. To access methods of the same class "this" keyword is used.

**36) Can constructor take parameters?**

Yes. Constructor can take arguments.

**39) Can you explain about literals in Java?**

Literals are used to represent a fixed value in source code. Literals don't require computation.

For Example: int Days = 7;

**40) Is it possible to override an overloaded method in Java ?**

Yes. We can override an overloaded method if that method is not a static or final.

**42) Can you explain about autoboxing and unboxing in Java?**

Autoboxing: Automatic conversion from primitive to wrapper object by the compiler is called as "Autoboxing"

Ex: class Test

{

psvm (String args[])

{

int x=10;

Integer I=x; //Autoboxing

sop(I);//10

}

}

It was introduced in the version of java 1.5

Autounboxing: Automatic conversion from wrapper object to primitive by the compiler is called as "Autounboxing"

Ex: class Test

{

psvm (String args[])

{

Integer I=new Integer(10);

int x=I;

sop(x);//10

int y=new Integer(20); //Autounboxing

sop(y);//20

}

}

**43) Is it possible to have multiple public classes in Java?**

As per java language specification, there can be only one public class in a file

**44) What is responsibility of Garbage Collector?**

For efficient memory utilization, Java uses Garbage collector. The main objective of the garbage collector is free up memory by detecting unused objects and deleting them. Garbage Collector gets invoked automatically and it won't assure required memory for a program to run.

**45) What is the use of finalize method?**

finalize() method is automatically called by the garbage collector to maintain clean the clean-up activities when an object is to destroy. So, we can override the finalize () method in the user de fined class to place the cleanup activities.

**46) Why String is immutable in Java?**

String objects are Immutable. Once we created a string object we can't perform any changes in the existing object, if we are trying to perform any changes in the existing object if those changes a new object will be created, this non changeable behavior is nothing but immutability.

Ex: String s= new String("Java");

s.concat("Technology");

sop(s);

o/p:java

**47) Why String Buffer is mutable in Java?**

String Buffer objects are mutable. Once we created a string buffer object we can perform any changes in the existing object, this changeable behavior is nothing but mutability.

Ex: StringBuffer sb= new StringBuffer("Java");

sb.append("Technology");

sop(sb);

o/p:java Technology

**48). What is the difference when String is gets created using literal or new() operator ?**

When we create string with new () it’s created in heap and not added into string pool while String created using literal are created in String pool itself.

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. It’s only when you create String object as String literal e.g. String s = "Test" Java automatically put that into String pool.

**49) Can abstract class have constructor in Java?**

Yes, abstract class can have constructor in Java. You can either explicitly provide constructor to abstract class or if you don't, compiler will add default constructor of no argument in abstract class.

**50) Which two methods is overridden by an Object, intended to be used as key in HashMap?**

Equals and Hash Code

**51) Difference between wait and sleep in Java?**

Main difference between wait and sleep is that wait () method release the acquired monitor when thread is waiting while Thread. Sleep () method keeps the lock or monitor even if thread is waiting

**52) What is the difference between String buffer and String builder?**

String Builder is exactly same as String Buffer (including methods and constructors) except the following difference.

**String Buffer String Builder**

1. All the methods are synchronized 1.All the methods are non-synchronized

2. At a time one thread is allowed to operate, hence string buffer object is thread safe. 2. At a time multiple threads are allowed to operate, hence string buffer object is not-thread safe

3. Relatively performance is very low 3. Relatively performance is very high

4. Introduced in java 1.0 version 4.Introduced in java 5.0 version

**53) What is the difference between String, String buffer and String builder?**

If the content is fixed won't change frequently then we should go for String.

If the content is not fixed and keep on changing but Thread safety is required, then we should go for String Buffer.

If the content is not fixed and keep on changing but Thread safety is not required, then we should go for String Builder.

**54) 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.

**55) Difference between final, finally and finalize ?**

There are many differences between final, finally and finalize. A list of differences between final, finally and finalize are given below:

No. final finally finalize

1) Final is used to apply restrictions on class, method and variable. Final class can't be inherited, final method can't be overridden and final variable value can't be changed. (1)Finally is used to place important code, it will be executed whether exception is handled or not. (1)Finalize is used to perform clean up processing just before object is garbage collected.

2) Final is a keyword. Finally is a block. Finalize is a method.

**56) Differences between checked exception and un checked exception?**

Checked exceptions: The exceptions which are checked by the complier for smooth termination of execution at run time is called as Checked exceptions. In this case compiler will check whether we are handling the exception, if the programmer is not handling the exception, then we will get compile time error.

Ex: FileNotFoundException, EOFException.

Unchecked exceptions: The exceptions which are not checked by the complier is called as Unchecked exceptions. In the case of unchecked exception compiler won't check whether the programmer is handling the exception or not.

Ex: AirthmeticException, NullPointerException.

Note: 1.Whether the exception is checked or unchecked compulsory it will occur at Runtime. There is no chance occurring any exception at compile time.

2. Runtime Exception and its child classes and error and it's child classes are called as unchecked, except this remaining all are called as checked exceptions

Ex:

import java.io.\*;

Class Test{

public static void main(String args[])

{

PrintWriter pw=new PrintWriter(abc.txt);

pw.println("Hello");

S.O.P(10/0);

}

}

**63) What is a thread? How many ways we can create thread?**

Thread is a independent flow of execution to complete a task. In java we can create thread in two ways

1. By extending thread class

2. By implementing runnable interface

By extending thread class:

class MyThread extends Thread{

public void run(){

for(int i=0;i<10;i++){

Syste.out.println("User Thread");

}

}

public static void main(String args[]){

MyThread mt=new Mythread();

mt.start();

for(int i=0;i<10;i++)

{

System.out.println("Main Thread");

}}}

By implementing runnable interface:

class MyRunnable implements Runnable{

public void run(){

for(int i=0;i<10;i++){

Syste.out.println("User Thread");

}

}

public static void main(String args[]){

MyRunnable mr=new MyRunnable();

Thread t=new Thread(mr);

t.start();

for(int i=0;i<10;i++)

{

System.out.println("Main Thread");

}

}

}

|  |
| --- |
| **63)Life cycle of a Thread (Thread States)**   * 1. [New](http://www.javatpoint.com/life-cycle-of-a-thread#threadstatenew)   2. [Runnable](http://www.javatpoint.com/life-cycle-of-a-thread#threadstaterunnable)   3. [Running](http://www.javatpoint.com/life-cycle-of-a-thread#threadstaterunning)   4. [Non-Runnable (Blocked)](http://www.javatpoint.com/life-cycle-of-a-thread#threadstateblocked)   5. [Terminated](http://www.javatpoint.com/life-cycle-of-a-thread#threadstateterminated)   A thread can be in one of the five states. According to sun, there is only 4 states in **thread life cycle in java** new, runnable, non-runnable and terminated. There is no running state.  But for better understanding the threads, we are explaining it in the 5 states.  The life cycle of the thread in java is controlled by JVM. The java thread states are as follows:   1. New 2. Runnable 3. Running 4. Non-Runnable (Blocked) 5. Terminated   1) New: The thread is in new state if you create an instance of Thread class but before the invocation of start () method. |

2) Runnable: The thread is in runnable state after invocation of start() method, but the thread scheduler has not selected it to be the running thread.

3) Running: The thread is in running state if the thread scheduler has selected it.

4) Non-Runnable (Blocked): This is the state when the thread is still alive, but is currently not eligible to run.

5) Terminated: A thread is in terminated or dead state when its run () method exits.

**66) Can we execute a program without main () method?**

Yes, by using static block we can execute the program.

Class Sample

{

Static

{

System.out.println (“Static block”);

System.exit(1);

}

}