JAVA INTERVIEW QUESTIONS

1. What is **Static** Keyword in Java

Static keyword is mainly used for memory management. Memory allocation for static variable happens only once when the class is loaded in the memory.

- It can be used with variables, methods, blocks and nested classes.
- Static variables are initialized only once at the start of the execution. If not initialized then default value is assigned.
- A single copy is shared by all instances of the class.
- A static variable/method can be accessed directly by the class name and doesn't need any object. JVM executes static blocks before the main method.
- You cannot access a non static member from a static member
- This and Super cannot be used for static members
- cannot override static method.

2. How many types of variables are found in java

1) Local Variable

A variable declared inside the body of the method is called local variable. A local variable cannot be defined with "**static**" keyword. It is necessary to initialize local variables because they don't get any default value like instance variables.

2) Instance Variable

A variable declared inside the class but outside the body of the method, is called instance variable.

3) Static variable

A variable which is declared as static is called static variable. It cannot be local.

Example to understand the types of variables in java

```
1. class A{
2. int data=50;//instance variable
3. static int m=100;//static variable
4. void method(){
5. int n=90;//local variable
6. }
7. }//end of class
```

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.

4) 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.

example.

```
class Simple1{
public static void main(String args[]){
Simple1 s=new Simple1();
System.out.println(s instanceof Simple1);//true
```

Note : 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.

5) What is the abstraction?

Abstraction is a process of hiding the implementation details and showing only functionality to the user. Abstraction enables you to focus on what the object does instead of how it does it. There are two ways to achieve the abstraction.

i. Abstract Class

ii. Interface

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.

7) What is the difference between abstraction and encapsulation?

Abstraction hides the implementation details whereas encapsulation wraps code and data into a single unit.

8) 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.

9) 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.

10) Is it possible to instantiate the abstract class?

No, the abstract class can never be instantiated. If its instantiated then the purpose of abstract class is defeated.

11) 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. There can be only abstract methods in the Java interface, not method body. 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.

12) 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.
public interface Serializable{ }

13) 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.
example.

```
public class Student{
private String college="AKG";
public String getCollege(){
return college;
```

14) 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.
example.

```
public class Student{
private String college;
public void setCollege(String college){
this.college=college;
```

15) 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.

16) Can I import same package/class twice?

One can import the same package or the same class multiple times. However, the JVM will internally load the class only once no matter how many times you import the same class.

17) What is the static import?

Static import allows public **static** members (fields and methods) of another class, to be used in **Java** code without specifying the class in which the field has been defined.

Example

import static java.lang.Math.*;

```
import static java.lang.System.out;

public class HelloWorld {
    public static void main(String[] args) {
        out.println("Hello World!");
        out.println("Considering a circle with a diameter of 5 cm, it has");
        out.println("a circumference of " + (PI * 5) + " cm");
        out.println("and an area of " + (PI * pow(2.5, 2)) + " sq. cm");
```

you dont have to write Math.PI or Math.pow etc.

18) 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.

19) 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.

20) What are the differences between String and StringBuffer?

21) What are the differences between StringBuffer and StringBuilder?

22) 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.

example.

```
public final class Employee{
final String pancardNumber;
public Employee(String pancardNumber){
this.pancardNumber=pancardNumber; }
public String getPancardNumber(){
return pancardNumber;
```

23) 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. By overriding the toString() method of the Object class, we can return the values of the object as desired.

example.

```
class Student{
 int rollno;
String name;
String city;
Student(int rollno, String name, String city){
this.rollno=rollno; this.name=name; this.city=city; }
public String toString(){
//overriding the toString() method return rollno+" "+name+" "+city; }
public static void main(String args[]){
Student s1=new Student(101,"Raj","lucknow");
Student s2=new Student(102,"Vijay","ghaziabad");
System.out.println(s1);//compiler writes here s1.toString()
System.out.println(s2);//compiler writes here s2.toString()
} }
```

Output: 101 Raj lucknow

102 Vijay ghaziabad

24) 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.

25) 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.

example

```
public class TestGarbage1{
public void finalize(){
System.out.println("object is garbage collected");}
public static void main(String args[]){
TestGarbage1 s1=new TestGarbage1();
TestGarbage1 s2=new TestGarbage1();
s1=null; s2=null; System.gc();
```

26) 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.

27) How can an object be unreferenced?

1) By nulling a reference. example Employee e=new Employee(); e=null;
2) By assigning a reference to another. Employee e1=new Employee();
Employee e2=new Employee();

e1=e2;//now the first object referred by e1 is available for garbage collection

3) By anonymous object. new Employee();

28) 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. The cleanup processing is the process to free up all the resources, objects created without new keyword, network which was previously used and no longer needed. finalize method is present in the object class. Here, we must note that garbage collection is not guaranteed.

29) What are the FileInputStream and FileOutputStream?

Java FileOutputStream is used for writing data to a file. You can write byte-oriented as well as characteroriented data through the FileOutputStream class. However, for character-oriented data, it is preferred to use FileWriter than FileOutputStream.

example

```
public class FileOutputStreamExample {
public static void main(String args[]){
 try{
        FileOutputStream fout=new FileOutputStream("D:\\testout.txt");
```

```
fout.write(65);
fout.close();
System.out.println("success...");
}catch(Exception e){
System.out.println(e);
```

30) What is the purpose of using BufferedInputStream and BufferedOutputStream classes?

Java BufferedOutputStream adds more efficiency than to write data directly into a stream. It internally uses a buffer to store data. 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.

31) In Java, How many ways you can take input from the console?

1. Using BufferedReader class: It provides an efficient reading as the input gets buffered.
   example.
   ```
   public class Person {
   public static void main(String[] args) throws IOException {
   System.out.println("Enter the name of the person");
   BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
   String name = reader.readLine();
   System.out.println(name);
   ```
2. Using Scanner class: The Java Scanner class breaks the input into tokens using a delimiter that is whitespace by default. Java Scanner class extends Object class and implements Iterator and Closeable interfaces.
   example.
   ```
   public class ScannerClassExample2 {
   public static void main(String args[]){
   String str = "Hello/";
   Scanner scanner = new Scanner(str);
   System.out.println("Boolean Result: "+scanner.hasNextBoolean());
   scanner.useDelimiter("/");   //Change the delimiter of this scanner
   while(scanner.hasNext() {
   System.out.println(scanner.next()); }
   scanner.close();
   ```
3. 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.
   example.
   ```
   class ReadStringTest{
   public static void main(String args[]){
   Console c=System.console();
   System.out.println("Enter your name: ");
   String n=c.readLine();
   System.out.println("Welcome "+n);
   ```

32) 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.

33) How can you make a class serializable in Java?

A class can become serializable by implementing the Serializable interface.

34) 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

35) 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.

```
class Depersist{
public static void main(String args[]) throws Exception{
ObjectInputStream in=new ObjectInputStream(new FileInputStream("f.txt"));
Student s=(Student)in.readObject();
System.out.println(s.id+" "+s.name);
 in.close();
```

36) 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.

37) 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.

example.

```
import java.io.*;
import java.net.*;
public class InetDemo{
public static void main(String[] args){
try{ InetAddress ip=InetAddress.getByName("195.201.10.8");
System.out.println("Host Name: "+ip.getHostName());
}catch(Exception e){
System.out.println(e)
```

38) 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.

39) 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.

40) 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.

41) 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.

42) Write a Java program that prints all the values given at command-line.

```
Program class A{
public static void main(String args[]){
for(int i=0;i < args.length; i++){
System.out.println(args[i]);
```