**1) Explain run time polymorphism in Java ?**

Polymorphism can be explained as an object's ability to adapt to the program's context and take multiple forms. The method overriding is an example of run time polymorphism. You can have a method in a subclass, which overrides the method in its super classes with the same name and signature. At run time, Java virtual machine determines the appropriate method to be called.

**2) What are the rules (method access permission and exception) that needs to be followed, during method overloading and overriding ?**  
  
During method Overloading, method name should remain same. But method signature can vary. Components of signature that can vary are

* Number of arguments
* Datatype of arguments
* Order of arguments

During method Overriding, make sure that the method is not throwing checked exceptions that are new or higher than those declared by the overridden method.But we can't override Static and Final methods.

**3) What is the difference between an Interface and abstract class?**

|  |  |
| --- | --- |
| **Abstract** | **Interface** |
| Supports Single inheritance | Supports Multiple inheritance |
| Supports abstract and Non-abstract methods | Allows only abstract methods |
| Supports Non-static and non-final variables also. | Variables must be static and final(implicitly) |
| Supports non public member | Only public members are allowed |
| Using extends keyword | Using implements keyword |
| It can invoke if main exists | Pure abstract |
| Faster | Flexible |

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

|  |  |
| --- | --- |
| **Compile time Polymorphism** | **Run time Polymorphism** |
| Method are called at compile time | Method are called at run time |
| Ex: Overloading | Ex: Overriding |

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

|  |  |
| --- | --- |
| **Overloading** | **Overriding** |
| Methods are overloaded during compile time | Method overriding takes place during runtime |
| All the overloaded methods should be placed in the same class | We can override methods in sub class |
| We can overload static methods | Static methods can't be overridden |
| Methods are bonded together using static binding. | Overridden method are bonded via dynamic bonding based upon actual Object. |
| To overload a method, method signature needs to be changed | There is no need to change the signature |
| Private and final method can be overloaded. | Private and final method can't be overridden |
| Method is relatively fast. | Method is relatively slow. |

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

|  |  |
| --- | --- |
| **Class** | **Object** |
| Template/Blue print of an object. | It is an instance of a class. Object have states and behaviors. |
| A logical construct. | A Physical reality. |

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

**Abstraction**  
It denotes the critical properties of an object which differentiate from other object and thus provide crisply defined conceptual boundaries relative to the perspective viewer.  
  
**Encapsulation**  
Encapsulation can be explained as a mechanism which binds the code and the data it manipulates. It also keeps them safe from external intervention and misuse.  
  
**Inheritance**:  
One object inherits the properties and methods of another Object.  
  
**Polymorphism**  
It is an ability of an object to take on many forms. Ex: Compile time polymorphism – method over loading. Run time polymorphism – method overriding

**8) Why Java is not supporting multiple inheritance ?**  
  
Main features of java are very Simple. if multiple inheritance is supported, it creates ambiguity around Diamond problem and it does complicate the design and creates problem during casting, chaining etc. So Java will support multi-inheritance via single inheritance with interfaces to overcome above issues.

**9) 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.

**10) 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.

**20) Can you explain about markable 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

**21) Can you explain about reflection in Java ?**  
  
If a programmer wants to access entities or invoke methods in a program dynamically, i.e. if the programmer is unaware of the methods and variables that needs to be invoked at runtime but unaware of it while coding, we can use reflection. For example

Method method = ABC.getClass().getMethod("doSomething", null);  
method.invoke(ABC, null);

**22) Can you explain about java.lang.class ?**  
  
When JVM creates an instance of a class, it creates an object "java.lang.Class object" which describes the type of the object. This class object is shared by all the objects of a class. If you want to access the class object of an instance, use getClass() method of the object. This method is inherited from java.lang.Object

Ex: Created two instances class called Programmer  
Programmer A = new Programmer();  
Programmer B = new Programmer();  
// For check Instances  
if(A.getClass() == B.getClass())  
{  
 System.out.println("A and B are instances of same class");  
}else{  
 System.out.println("A and B are instances of different class");  
}

**23) Can you explain about Singleton class in Java ?**  
  
Singleton class is used to control no of object created for a class, limiting the number to one. But if the situation changes in future, it allows to create more objects without affecting existing clients.

**24) Can you explain about Static class in Java ?**  
  
A class can be made static provided that the class is a nested class. A nested class is class which is defined inside a class. But top class can't me made static. Example :

public class Test  
{   
 static class StaticInnerClass  
 {  
 public static void innerMethod()  
 { System.out.println("Static Inner Class!"); }  
 }   
 public static void main(String args[])  
 {  
 Test.StaticInnerClass.innerMethod();  
 }  
}

**25) Can you explain about volatile Keywords in Java ?**

* Volatile keyword is used to indicate the threads using a common variable that, the variable which is declared as Volatile can be updated by multiple threads. So threads should not cache the threads locally and in turn should get the value for the variable from main memory.
* If a variable is declared as volatile, it won't be serialized.

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

* Determination of a category of a file is simplified.
* Name space collision is avoided.
* Access restriction can be applied with the use of packages.
* Packages provide reusability of code

**27) Can you explain about Java naming convention ?**  
  
Common Naming conventions as below :

* package names always start with lowercase characters. Ex: java.util
* Class names always begin with a capital letter and followed next word start with a capital letter. Ex: GregorianCalendar
* Java Naming convention specifies that instances and other variables must start with lowercase followed next word should be capital letter. Ex : MyClass myClass = new MyClass();
* Constant variables are declared using “static final” modifiers. And such variables must contain only UpperCase charachters and multiple words must be seperated using ‘\_’. Ex: static final char END\_OF\_FILE = 'e';
* Methods in Java also follow the same like Objects and variables. For example

void myMethod(){  
String strVal = "ABCD";  
}

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

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

**29) What are the new features available in Java 1.7 ?**

* Strings in switch Statement
* Type Inference for Generic Instance Creation
* Multiple Exception Handling
* Support for Dynamic Languages
* Try with Resources
* Java nio Package
* Binary Literals, underscore in literals
* Diamond Syntax
* Automatic null Handling

**30) What are the advantage 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

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

String is a special type of immutable class. Immutable class is a class which once created, it’s contents can not be changed. Immutable objects are the objects whose state can't be changed once constructed.

**32) Can you explain about information hiding in Java ?**  
  
Information hiding helps objects to hide critical information from other other objects accessing it. It effectively decouples the method being invoked from the internal workings of a function. By doing so, object can change the hidden portions of the function without changing the calling code. Encapsulation is a common technique programmers use to implement information hiding.

**33) Can you explain about encapsulation in Java ?**  
  
Encapsulation helps java to bind code and data it manipulates, restrict outside interference and misuse of data. It also hides irrelevant details of an object.

**34) Can you explain about the access modifier in Java ?**  
  
Access modifiers specifies the access levels of a variable or method. Java access modifiers are public, private, protected, default modifier (Default access modifier).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Access Modifiers** | **Same Class** | **Same Package** | **Subclass** | **Other packages** |
| public | Yes | Yes | Yes | Yes |
| protected | Yes | Yes | Yes | No |
| default | Yes | Yes | No | No |
| private | Yes | No | No | No |

**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 you explain about the constructor in Java ?**

* Java constructors are special methods that are called when an object is instantiated.
* When objects are instantiated, arguments passed to the constructor will initialized the variables in an object.
* Name of the constructor should be same as the name of the Class. It can't have any return type.
* A class can have multiple constructors. Calling a constructor from another constructor in the same class is called Constructor chaining.
* All classes have a default empty constructor.

**37) Can constructor take parameters ?**  
  
Yes. Constructor can take arguments.

**38) 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 and object instance, default constructor initialize variables with it's default values.

**39) What are the common reasons to define a default constructor ?**

* To construct an object with default values.
* To initialize an object that doesn't need parameters in that initialization process.
* To redefine the scope of the constructor. By making the default constructor private, Java prevents everyone other than the class from constructing an object.

**40) Can you explain about native method in Java ?**

* Native is non access modifier. It can be applied only to method.
* It indicates the Platform-Dependent implementation of method or code.

**41) Can you explain about strictfp keyword in Java ?**

If we want floating point values to be consistent across platforms, then we need to use "strictfp" as per IEEE 754 standard. When a program runs on multiple platforms, precision of floating point differ which in turn given different results. strictfp helps to enforce the precision across all platforms. For example

Class Level - public strictfp class StrictFpModifierExample{}  
Method Level - public strictfp void example() {}

**42) Can you explain about String pool ?**  
  
String Pool is a pool of strings stored in Java heap memory. String objects can be created either by new operator or by specifying the values in double quotes.

**Case 1 :**When a new string is created using double quotes, JVM searches string pool for the string with the same value. if it finds a string which matches the values, it will return the reference of the string. Else it will create a new string in the pool and returns that reference.

String s1 = "Cat";   
String s2 = "Cat";   
if(s1 == s2) System.out.println("equal"); //Prints equal.

**Case 2 :**When new operator is used to create a string, String class will be forced to create a new String object. To put the newly created string into the pool or assign it to another string, use intern().

String n1 = new String("ABCD");   
String n2 = new String("ABCD");   
if(n1 == n2) System.out.println("equal"); //No output.

**43) Differences between String, StringBuffer and StringBuilder in Java ?**

|  |  |  |
| --- | --- | --- |
| **String** | **StringBuffer** | **StringBuilder** |
| Immutable | Mutable | mutable |
| String operations such as append would be less efficient | String operations such as append would be more efficient, | String operations such as append would be more efficient |
| - | synchronized | Not synchronized. |
| - | versions 1.4 or below you’ll have to use StringBuffer. | StringBuilder was introduced in Java 1.5 |

**44) What are the advantage of using unicode characters ?**

* Much larger number of characters or group of characters
* Contains some non western European characters
* Supported by all modern technologies
* Enhance integration opportunities
* Easy conversion from legacy code pages

**45) 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, we will have a look at using literals to assign a value to an int variable.

int Days = 7;

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

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

**47) What is the maximum size of an int ?**  
  
-(2 power 31) to (2 power 31-1) or -2,147,483,648 to 2,147,483,647

**48) Can you explain about autoboxing and unboxing in Java ?**  
  
When primitive data types are automatically converted into it's  wrapper type, it is called boxing. The opposite operation of converting wrapper class objects to it's primitive type is known as unboxing.

ArrayList<Integer> list = new ArrayList<Integer>();  
list.add(1); //autoboxing - primitive to object  
int number = list.get(0); // unboxing

**49) How to change the heap size of a JVM ?**   
  
The old generation's default heap size can be overridden by using the -Xms and -Xmx switches to specify the initial and maximum sizes respectively:

Format - java -Xms <initial size> -Xmx <maximum size> program  
Example - java -Xms64m -Xmx128m Myprogram