**1. Types of polymorphism**

Compile Time and Run Time

**2. Difference between override and overload**

|  |  |  |
| --- | --- | --- |
| **No.** | **Method Overloading** | **Method Overriding** |
| 1) | Method overloading is used *to increase the readability* of the program. | Method overriding is used *to provide the specific implementation* of the method that is already provided by its super class. |
| 2) | Method overloading is performed *within class*. | Method overriding occurs *in two classes* that have IS-A (inheritance) relationship. |
| 3) | In case of method overloading, *parameter must be different*. | In case of method overriding, *parameter must be same*. |
| 4) | Method overloading is the example of *compile time polymorphism*. | Method overriding is the example of *run time polymorphism*. |
| 5) | In java, method overloading can't be performed by changing return type of the method only. *Return type can be same or different* in method overloading. But you must have to change the parameter. | *Return type must be same or covariant* in method overriding. |
|  |  |  |

**3. Methods that cannot be overloaded**

**Static** methods cannot be overridden

**Private** methods can be overloaded

Static methods cannot be overloaded only if they are differ by static keyword. Otherwise, we can overload static methods.

Yes, you can overload main method in Java. you have to call the overloaded main method from the actual main method. Yes, main method can be overloaded. Overloaded main method has to be called from inside the "public static void main (String args [])" as this is the entry point when the class is launched by the JVM.

**4. Static block and instance block**

Features of **static *initialization* block in java >**

* **Static blocks** are **also** called **Static initialization blocks** in java.
* Static block executes **when class is loaded** in java.
* **static blocks** executes **before instance blocks** in java.
* Only static variables can be **accessed** inside static block in java
* **static blocks** can be used for **initializing** static **variables or** calling any static **method** in java.
* **this** [**keyword**](http://www.javamadesoeasy.com/2015/05/keywords-in-java-language.html) cannot be used in static block in java.

|  |  |
| --- | --- |
| ***Static block*** | ***instance block*** |
| Known only as [**static**](http://www.javamadesoeasy.com/2015/05/static-keyword-in-java-variable-method.html) **initialization block** in java. | Also known as **non-static initialization block** in java. |
| **static blocks** executes **before instance blocks** in java. | **instance blocks** executes **after static blocks** in java**.** |
| Only static variables can be **accessed** inside **static block** | Static and non-static variables (instance variables) can be **accessed** inside **instance block.** |
| **static blocks** can be used for **initializing** static **variables**  **or**  calling any static **method** in java. | **instance blocks** can be used for initializing instance **variables**  ***or***  calling any instance **method** in java. |
| **static blocks** executes **when class is loaded** in java. | **instance block** executes only **when instance of class is created**, not called when class is loaded in java. |
| **this** [**keyword**](http://www.javamadesoeasy.com/2015/05/keywords-in-java-language.html) cannot be used in **static blocks**. | **this keyword** can be used in **instance block**. |

**5. Difference in Static and non-static**

Static methods are methods that are associated with a class, whereas non static methods are methods that are associated with objects of a class. A class needs to be instantiated first to invoke a non static method, but static methods do not have this requirement. They can be simply invoked using the name of the class that holds the static method. Another important difference is that a non static method usually possesses a reference to the object that called the method, and it can be accessed using this keyword inside the method. But this keyword cannot be used in static methods since they are not associated with a particular object.

**6. Encapsulation and common use cases**

Encapsulation in Java is a process of wrapping code and data together into a single unit, for example, a capsule which is mixed of several medicines.

encapsulation in java

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 a fully encapsulated class.

Advantage of Encapsulation in Java

By providing only a setter or getter method, you can make the class read-only or write-only. In other words, you can skip the getter or setter methods.

**7. Abstract vs interfaces**

**Abstract class Interface**

1) Abstract class can have abstract and non-abstract methods. Interface can have only abstract methods. Since Java 8, it can have default and static methods also.

2) Abstract class doesn't support multiple inheritance.Interface supports multiple inheritance.

3) Abstract class can have final, non-final, static and non-static variables. Interface has only static and final variables.

4) Abstract class can provide the implementation of interface. Interface can't provide the implementation of abstract class.

5) The abstract keyword is used to declare abstract class. The interface keyword is used to declare interface.

6) An abstract class can extend another Java class and implement multiple Java interfaces. An interface can extend another Java interface only.

7) An abstract class can be extended using keyword "extends". An interface can be implemented using keyword "implements".

8) A Java abstract class can have class members like private, protected, etc. Members of a Java interface are public by default.

9)Example:

public abstract class Shape{

public abstract void draw();

} Example:

public interface Drawable{

void draw();

**8. Inheritance in java**: To achieve runtime dynamic binding and reusability among objects

**9. Methods that cannot be overloaded**

**10. Type casting in java**

Type casting is when you assign a value of one primitive data type to another type.

In Java, there are two types of casting:

Widening Casting (automatically) - converting a smaller type to a larger type size

byte -> short -> char -> int -> long -> float -> double

Narrowing Casting (manually) - converting a larger type to a smaller size type

double -> float -> long -> int -> char -> short -> byte

float c=(float)a/b;

int myInt = 9;

double myDouble = myInt; // Automatic casting: int to double

Double ab=45.7d;

int myint=(int) ab; // Manual casting: double to int

**11. String buffer and string builder**

Java provides three classes to represent a sequence of characters: String, StringBuffer, and StringBuilder. The String class is an immutable class whereas StringBuffer and StringBuilder classes are mutable. There are many differences between StringBuffer and StringBuilder. The StringBuilder class is introduced since JDK 1.5.

A list of differences between StringBuffer and StringBuilder are given below:

No. StringBuffer StringBuilder

1) StringBuffer is synchronized i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously. StringBuilder is non-synchronized i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously.

2) StringBuffer is less efficient than StringBuilder. StringBuilder is more efficient than StringBuffer.

**12. Why string is immutable in java**

The String pool cannot be possible if String is not immutable in Java. A lot of heap space is saved by JRE.

If we don't make the String immutable, it will pose a serious security threat to the application.

The String is safe for multithreading because of its immutableness. ...

Immutability gives the security of loading the correct class by Classloader.

**13. How to handle exceptions**

using try, catch block.

**14. Can I write try catch without the catch block?**

yes, using finally block with try.

**15. Difference between throws and throw**

1. Throws clause is used to declare an exception, which means it works similar to the try-catch block. On the other hand throw keyword is used to throw an exception explicitly.

2. If we see syntax wise than throw is followed by an instance of Exception class and throws is followed by exception class names.

For example:

throw new ArithmeticException("Arithmetic Exception");

and

throws ArithmeticException;

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

**16. Use of iterator in java,**

Inshort, iterator is used to iterate through collections(List,maps)

**17. Difference in Final, finally and finalize**

Final keyword is used with variable,method and class

Finally is used with try ,catch block.

Finalize is used in garbage collection

**18. Boxing and unboxing in java.**

Converting primitive datatype to object is called boxing. Whereas, converting an object into corresponding primitive datatype is known as unboxing.

The automatic conversion of primitive data types into its equivalent Wrapper type is known as boxing and opposite operation is known as unboxing. This is the new feature of Java5. So, java programmer doesn't need to write the conversion code.

int a=50;

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

Integer i=new Integer(50);

int a=i; //unboxing.

**19. Increment and decrement** operation

**20. Variable Args**

Var arguments should be the last argument in method

**21. This and super keyword in java**

‘this’ is a reference variable that refers to the current object.

* Using ‘this’ keyword to refer current class instance variables
* Using this () to invoke current class constructor
* Using ‘this’ keyword to return the current class instance

The **super** keyword in Java is a reference variable which is used to refer immediate parent class object.

Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

Usage of Java super Keyword

* super can be used to refer immediate parent class instance variable.
* super can be used to invoke immediate parent class method.
* super() can be used to invoke immediate parent class constructor.

**22. Issues during Switch case without break**

All the statements will get executed after the case matches if we don't use break statement.

**23. Upcasting and downcasting**

**24. Baseclass of all class in java**

Object class is the baseclass of all classes in java. Since everything is represented in form of objects in java

**25. Baseclass of error and exceptions**

Throwable class is the Baseclass of error and exceptions classes in java

**26. Access specifiers**

Public, private, protected, default

**27. Continue and break statement**

**28. Can main method return any value**

No.

**29. Can we overload main method. What happens when overloaded**

Yes, you can overload main method in Java. you have to call the overloaded main method from the actual main method. Yes, main method can be overloaded. Overloaded main method has to be called from inside the "public static void main (String args [])" as this is the entry point when the class is launched by the JVM.

**30.how to execute and statement before main method**

using static block.

**31. Difference between == and equals()**

Main difference between .equals () method and == operator is that one is method and other is operator. We can use == operators for reference comparison ( address comparison) and .equals () method for content comparison.

**32. Can user declare constructor as final?**

No.

**33. Can we cast any other type to Boolean data with type casting.?**

No

**34.Does java compile if user use 'static public void' instead of 'public static void'**

Yes.

**35.Can we use this() and super() in a constructor**

We can't use both together in a constructor. Reason being both should be the first statement inside a constructor.

**36. Can we create object of abstract class**

No.

**37. Can we create reference for an abstract class**

Yes

**38. Can we declare a class as static**

yes, but only with the nested class we can use static keyword

**39. What is instanceOf keyword?**

In Java, instanceof keyword is a binary operator. It is used to check whether an object is an instance of a particular class or not. The operator also checks whether an object is an instance of a class that implements an interface .It returns either true or false. It returns true if the left side of the expression is an instance of the class name on the right side.

**40. What's the load factor of HashMap.**

The initial capacity of hashmap is=16. The default load factor of hashmap=0.75.

The Load Factor is a threshold, if the ratio of the current element by initial capacity crosses this threshold then the capacity increases so that the operational complexity of the HashMap remains O(1).

**41. How to prevent a class from being sub classes.**

You can prevent a class from being subclassed by using the final keyword in the class's declaration. Similarly, you can prevent a method from being overridden by subclasses by declaring it as a final method.

### **Why can we not override static method?**

It is because the static method is bound with class whereas instance method is bound with an object. Static belongs to the class area, and an instance belongs to the heap area.

**42. Final variable, final method and final class.**

Final Variable=becomes constant

Final method= cannot be overridden

Final class= cannot be inherited

**43. Ways to create a string variable.**

**String str=”abc”;**

**String str=new String(“abc”);**

**44. What is gc() - garbage collector**

The gc() method is used to invoke the garbage collector to perform cleanup processing. The gc() is found in System and Runtime classes. Note: Garbage collection is performed by a daemon thread called Garbage Collector(GC). This thread calls the finalize() method before object is garbage collected. Simple Example of garbage collection in java.

**45. Subclass and innerclass**

**46. Infinite loop in java**

**47. How to make copy of an element**

**48. Checked and unchecked exceptions**

|  |  |
| --- | --- |
| Checked exceptions occur at compile time. | Unchecked exceptions occur at runtime. |
| The compiler checks a checked exception. | The compiler does not check these types of exceptions. |
| These types of exceptions can be handled at the time of compilation. | These types of exceptions cannot be a catch or handle at the time of compilation,  because they get generated by the mistakes in the program. |
| They are the sub-class of the exception class. | They are runtime exceptions and hence are not a part of the Exception class. |
| Here, the JVM needs the exception to catch and handle. | Here, the JVM does not require the exception to catch and handle. |
| Examples of Checked exceptions:   * File Not Found Exception * No Such Field Exception * Interrupted Exception * No Such Method Exception * Class Not Found Exception | Examples of Unchecked Exceptions:   * No Such Element Exception * Undeclared Throwable Exception * Empty Stack Exception * Arithmetic Exception * Null Pointer Exception * Array Index Out of Bounds Exception * Security Exception |



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Java Collection :** A Collection is a group of individual objects represented as a single unit. Java provides Collection Framework which defines several classes and interfaces to represent a group of objects as a single unit

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1. Classes inside List interface, Set interface, Map Interface**

List Interface: ArrayList, LinkedList

Set Inteface: HashSet, LinkedHashSet, TreeSet

Map Interface: HashMap,LinkedHashMap, TreeMap

**2. Arraylist vs Linkedlist**

**3. Arraylist vs array**

**4. Arraylist vs vector or stack**

**5. Which class of List Interface to be used if user have more insertions and deletions**

LinkedList is faster than ArrayList for insertion and deletion. Reason being we don’t need to shift elements in LinkedList

**6. Which class of List Interface to be used if user have more retrieval?**

ArrayList

ArrayList is faster than LinkedList if I randomly access its elements

**7. Set Interface: HashSet, TreeSet, SortedSet**

**8. Map - HashMap, HashTable, TreeMap, LinkedHashMap.**

**9. Stack and Queue**

**10. How to maintain insertion order in Set, List and Map**

Insertion order can be maintained in list, set or map by using linkedlist, linkedhashset, linkedhashmap

**11. How to sort elements in ascending order in Set and Map**

Using Collections.sort method

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GIT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**1. Use of common commands used:**

add, status, commit, push, pull, restore, checkout, clone

**2. How to handle merge conflicts?**

git merge --abort

The git merge --abort command helps in exiting the merge process and returning back to the state before the merging began

git reset

The git reset command is used at the time of merge conflict to reset the conflicted files to their original state.

The most direct way to resolve a merge conflict is to edit the conflicted file. Open the merge.txt file in your favorite editor. For our example lets simply remove all the conflict dividers. The modified merge.txt content should then look like:

**3. How merge issues occur.**

Conflicts while merging is a frequent part of the Git experience. If there are several developers working on the same file the odds of encountering a merge conflict increases. Most of the time, Git will automatically figure out how to integrate new changes. Conflicts generally arise when two people have changed the same lines in a file, or if one developer deleted a file while another developer was modifying it.