**JAVA QUESTIONS**

### What are the advantages of Packages in Java

There are various advantages of defining packages in Java.

* Packages avoid the name clashes.
* The Package provides easier access control.
* We can also have the hidden classes that are not visible outside and used by the package.
* It is easier to locate the related classes.

### What is object-oriented paradigm?

It is a programming paradigm based on objects having data and methods defined in the class to which it belongs. Object-oriented paradigm aims to incorporate the advantages of modularity and reusability. Objects are the instances of classes which interacts with one another to design applications and programs. There are the following features of the object-oriented paradigm.

* Follows the bottom-up approach in program design.
* Focus on data with methods to operate upon the object?s data
* Includes the concept like Encapsulation and abstraction which hides the complexities from the user and show only functionality.
* Implements the real-time approach like inheritance, abstraction, etc.
* The examples of the object-oriented paradigm are C++, Simula, Smalltalk, Python, C#, etc.

### What is the constructor?

The constructor can be defined as the special type of method that is used to initialize the state of an object. It is invoked when the class is instantiated, and the memory is allocated for the object. Every time, an object is created using the **new** keyword, the default constructor of the class is called. The name of the constructor must be similar to the class name. The constructor must not have an explicit return type.

### How many types of constructors are used in Java?

Based on the parameters passed in the constructors, there are two types of constructors in Java.

* **Default Constructor:** default constructor is the one which does not accept any value. The default constructor is mainly used to initialize the instance variable with the default values. It can also be used for performing some useful task on object creation. A default constructor is invoked implicitly by the compiler if there is no constructor defined in the class.
* **Parameterized Constructor:** The parameterized constructor is the one which can initialize the instance variables with the given values. In other words, we can say that the constructors which can accept the arguments are called parameterized constructors

### What do you understand by copy constructor in Java?

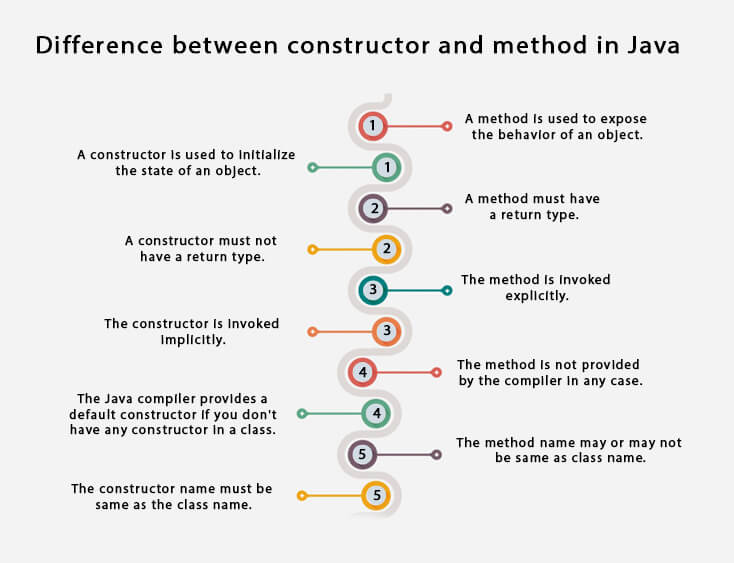
There is no copy constructor in java. However, we can copy the values from one object to another like copy constructor in C++.

There are many ways to copy the values of one object into another in java. They are:

* By constructor
* By assigning the values of one object into another
* By clone() method of Object class

In this example, we are going to copy the values of one object into another using java constructor

1. **class** Student6{
2. **int** id;
3. String name;
4. //constructor to initialize integer and string
5. Student6(**int** i,String n){
6. id = i;
7. name = n;
8. }
9. //constructor to initialize another object
10. Student6(Student6 s){
11. id = s.id;
12. name =s.name;
13. }
14. **void** display(){System.out.println(id+" "+name);}
16. **public** **static** **void** main(String args[]){
17. Student6 s1 = **new** Student6(111,"Karan");
18. Student6 s2 = **new** Student6(s1);
19. s1.display();
20. s2.display();
21. }
22. }



### What is the static variable?

The static variable is used to refer to the common property of all objects (that is not unique for each object), e.g., The company name of employees, college name of students, etc. Static variable gets memory only once in the class area at the time of class loading. Using a static variable makes your program more memory efficient (it saves memory). Static variable belongs to the class rather than the object.

 What is the static method?

* A static method belongs to the class rather than the object.
* There is no need to create the object to call the static methods.
* A static method can access and change the value of the static variable.

What are the restrictions that are applied to the Java static methods?

Two main restrictions are applied to the static methods.

* The static method can not use non-static data member or call the non-static method directly.
* this and super cannot be used in static context as they are non-static.

### Why is the main method static?

Because the object is not required to call the static method. If we make the main method non-static, JVM will have to create its object first and then call main() method which will lead to the extra memory allocation

### Can we override the static methods?

No, we can't override static methods.

### Can we make the abstract methods static in Java?

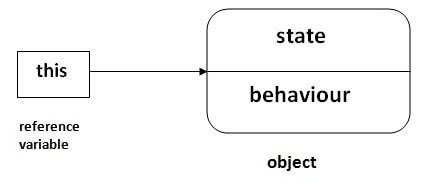
In Java, if we make the abstract methods static, It will become the part of the class, and we can directly call it which is unnecessary. Calling an undefined method is completely useless therefore it is not allowed.

### Can we declare the static variables and methods in an abstract class?

Yes, we can declare static variables and methods in an abstract method. As we know that there is no requirement to make the object to access the static context, therefore, we can access the static context declared inside the abstract class by using the name of the abstract class. Consider the following example.

### What is ****this**** keyword in java?

The **this** keyword is a reference variable that refers to the current object. There are the various uses of this keyword in Java. It can be used to refer to current class properties such as instance methods, variable, constructors, etc. It can also be passed as an argument into the methods or constructors. It can also be returned from the method as the current class instance.



### Why is multiple inheritance not supported in java?

To reduce the complexity and simplify the language, multiple inheritance is not supported in java. Consider a scenario where A, B, and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class.

Since the compile-time errors are better than runtime errors, Java renders compile-time error if you inherit 2 classes. So whether you have the same method or different, there will be a compile time error.

What is method overriding:

If a subclass provides a specific implementation of a method that is already provided by its parent class, it is known as Method Overriding. It is used for runtime polymorphism and to implement the interface methods.

**Rules for Method overriding**

* The method must have the same name as in the parent class.
* The method must have the same signature as in the parent class.
* Two classes must have an IS-A relationship between the

### Can we override the static method? Why?

No, you can't override the static method because they are the part of the class, not the object. It is because the static method is the part of the class, and it is bound with class whereas instance method is bound with the object, and static gets memory in class area, and instance gets memory in a heap.

### What is the final blank variable?

A final variable, not initialized at the time of declaration, is known as the final blank variable. We can't initialize the final blank variable directly. Instead, we have to initialize it by using the class constructor. It is useful in the case when the user has some data which must not be changed by others, for example, PAN Number

### What is the difference between the final method and abstract method?

The main difference between the final method and abstract method is that the abstract method cannot be final as we need to override them in the subclass to give its definition

### What is Runtime Polymorphism?

Runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time. In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.

### What is the difference between static binding and dynamic binding?

In case of the static binding, the type of the object is determined at compile-time whereas, in the dynamic binding, the type of the object is determined at runtime.

What is the abstraction?

Abstraction is a process of hiding the implementation details and showing only functionality to the user. It displays just the essential things to the user and hides the internal information, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery. Abstraction enables you to focus on what the object does instead of how it does it. Abstraction lets you focus on what the object does instead of how it does it. In Java, there are two ways to achieve the abstraction.

* Abstract Class
* Interface

### What is the difference between abstraction and encapsulation?

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

### 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

### 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. It is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java. In other words, you can say that interfaces can have abstract methods and variables. 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

### 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. |
| **Example:** public abstract class Shape{ public abstract void draw(); } | **Example:** public interface Drawable{ void draw(); } |

What are the advantages of Encapsulation in Java?

There are the following advantages of Encapsulation in Java?

* By providing only the 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.
* It provides you the control over the data. Suppose you want to set the value of id which should be greater than 100 only, you can write the logic inside the setter method. You can write the logic not to store the negative numbers in the setter methods.
* It is a way to achieve data hiding in Java because other class will not be able to access the data through the private data members.
* The encapsulate class is easy to test. So, it is better for unit testing.
* The standard IDE's are providing the facility to generate the getters and setters. So, it is easy and fast to create an encapsulated class in Java.

### What is the difference between Checked Exception and Unchecked Exception?

### 1) Checked Exception

The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions, e.g., IOException, SQLException, etc. Checked exceptions are checked at compile-time.

### 2) Unchecked Exception

The classes that extend RuntimeException are known as unchecked exceptions, e.g., ArithmeticException, NullPointerException, etc. Unchecked exceptions are not checked at compile-time.

### 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

### 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. There are eight wrapper classes present in **java.lang** package is given below

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