**Interview Questions**

1. What is a static keyword in Java?

Ans: - Static is a Non-Access Modifier. Static can be applied to variable, method, nested class, and initialization blocks (static block).

2. Why main () method is declared as static?

Ans: - If our main () method is not declared as static then the JVM has to create an object first and call which causes the problem of having extra memory allocation.

3. Can constructors be static in Java?

Ans: - In general, a static method means that “The Method belongs to the class and not to any particular object” but a constructor always invoked with respect to an object, so it makes no sense for a constructor to be static.

4. Can we use this to refer static members?

Ans: - Yes, it's possible to access the static variables of a class using this but it's discouraged and as per best practices this should be used on non-static reference.

5. What are all the differences between this and the super keyword?

Ans: - This refers to the current class object whereas super refers to the superclass object

Using this we can access all non-static methods and variables. Using super we can access superclass variables and methods from sub-class.

Using this(); call we can call other constructors in the same class. Using super we can call superclass constructor from sub-class constructor.

6. What is a final method?

Ans: - When a method is declared as final, then it is called a final method, The subclass can call the final method of the parent class but cannot override it.

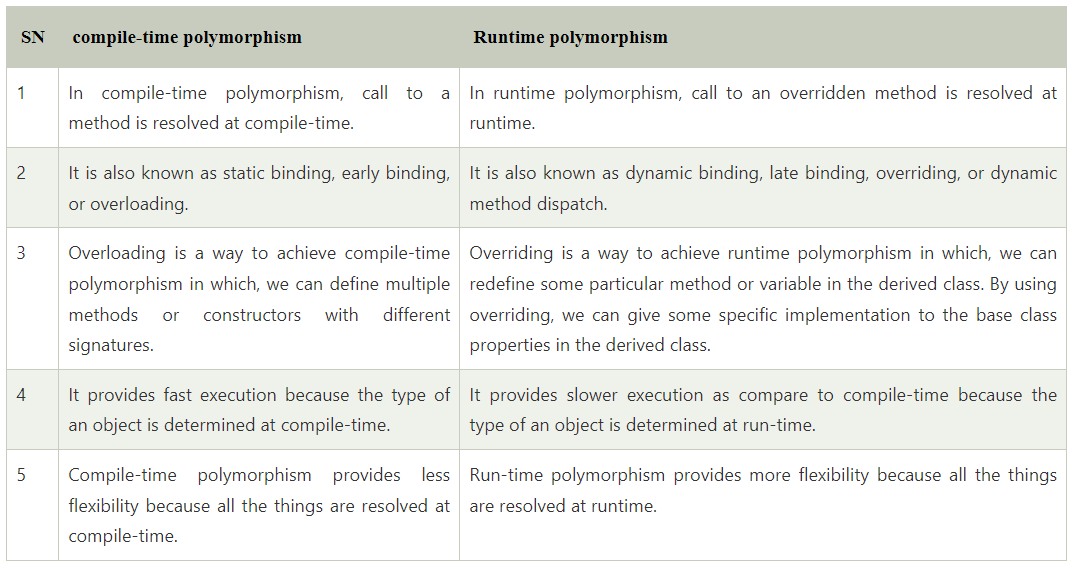
7. Can a main() method be declared final?

Ans: - Yes, the main() method can be declared as final and cannot be overridden.

8. What is a Static Final variable in Java?

Ans: - When have declared a variable as static final then the variable becomes a CONSTANT.  Only one copy of the variable exists which cannot be changed by any instance.

9.What is the difference between compile-time polymorphism and runtime polymorphism?



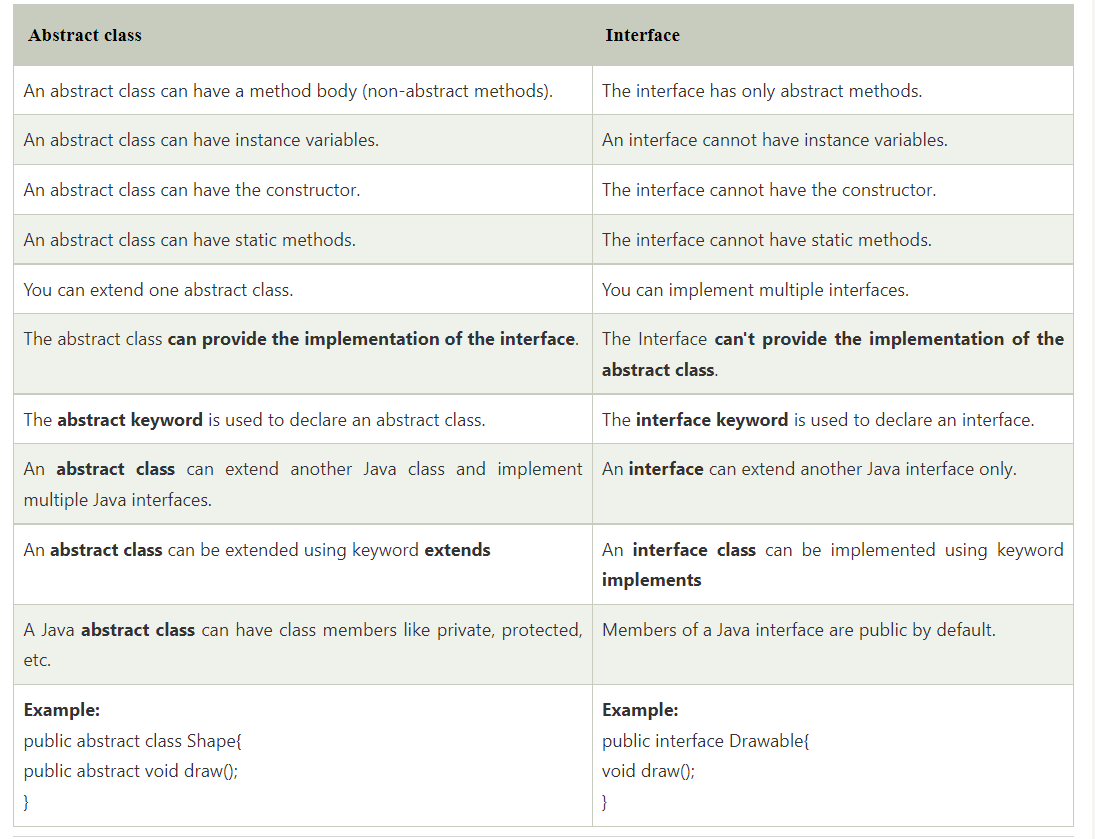
2. What is the Java instanceOf operator?

Ans: - 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.

3. What is the difference between abstraction and encapsulation?

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

4. What are the differences between abstract class and interface?



Q5. What is a superclass?

Ans: - A superclass—also called a base class—is a class that is a parent for more classes rather than objects. It usually contains the most basic code and data that will be used by every class and object under it. Using the above example, ‘beverage’ and ‘machine’ could be super classes for ‘soda’ and ‘computer.’

Q6. What is a subclass?

Ans: - A subclass is a class that falls under a superclass. It inherits from the superclass and is considered to have an “is-a” relationship with the superclass.

Q7. Are there any limitations of Inheritance?

Ans: - Yes, with more powers comes more complications. Inheritance is a very powerful feature in OOPs, but it has some limitations too. Inheritance needs more time to process, as it needs to navigate through multiple classes for its implementation. Also, the classes involved in Inheritance - the base class and the child class, are very tightly coupled together. So if one needs to make some changes, they might need to do nested changes in both classes. Inheritance might be complex for implementation, as well. So if not correctly implemented, this might lead to unexpected errors or incorrect outputs.

Q8. What are the various types of inheritance?

Ans: - Single inheritance

Multiple inheritances

Multi-level inheritance

Hierarchical inheritance

Hybrid inheritance

Q9. What is meant by static polymorphism?

Ans: - Static Polymorphism is commonly known as the Compile time polymorphism. Static polymorphism is the feature by which an object is linked with the respective function or operator based on the values during the compile time. Static or Compile time Polymorphism can be achieved through Method overloading or operator overloading.

1. Is java a fully object-oriented programming language?

Ans: - Java is not a fully object-oriented programming language because it supports primitive data types like - int, byte, short, long, etc., which are not object-oriented and, of course, are the opposite of oops.

2. What are the advantages of packages in java?

Ans: - There are various advantages of defining packages in Java.

Packages avoid 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.

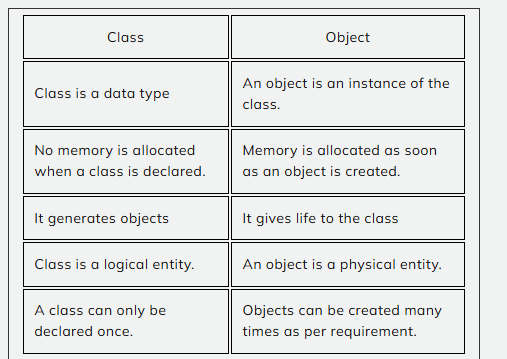
3. What happens if you don’t define a constructor in your class. Can we still create the object of that class?

Ans: - Yes, we can create that class’s object because the compiler automatically defines an empty, default constructor inside the class, which remains hidden to the programmer/user/outside world.

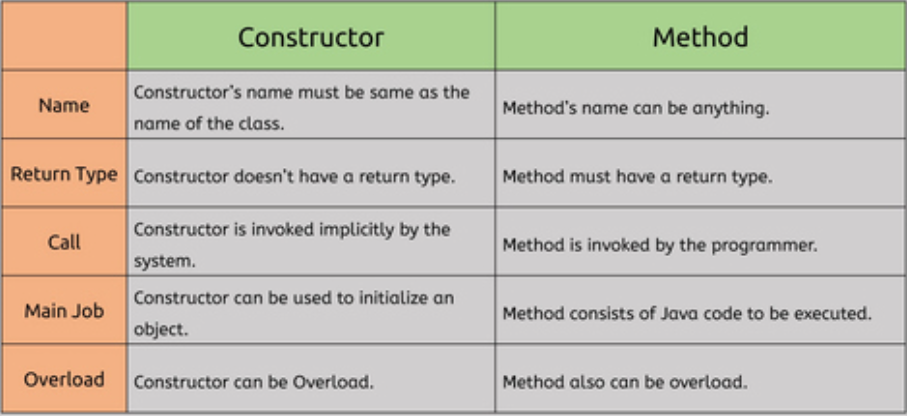
4. Why is OOPs so popular?

Ans: - Oops is so popular because it helps in writing a complex piece of code easily, and it also allows users to handle and maintain them easily. With OOPs, the code’s readability, understandability, and maintainability increase multifold.

 5. What are the differences between the class and the object?



6. What are the differences between the constructor and the method?



Q1. What is a Constructor in Java?

Constructor is just like a method in Java that is used to initialize the state of an object and will be invoked during the time of object creation.

Q2. What are the Rules for defining a constructor?

The constructor name should be the same as the class name

It cannot contain any return type

It can have all Access Modifiers are allowed (private, public, protected, default)

It Cannot have any Non Access Modifiers (final, static, abstract, synchronized)

No return statement is allowed

It can take any number of parameters

Q3. What is a No-arg constructor?

A constructor without arguments is called a no-arg constructor. In Java Default constructor is a no-arg constructor.

class Demo

{

    public Demo()

    {

        //no-arg constructor

    }

}

 Q4. Can we have both Default Constructor and Parameterized Constructor in the same class?

Yes, we have both Default Constructor and Parameterized Constructor in the same class.

Q5. What happens if you don’t define a constructor in your class. Can we still create the object of that class?

Yes, we can create the object of that class because the compiler defines an empty, default constructor inside the class automatically which remains hidden to the programmer/user/outside world.

Q6. Will the compiler create the Default Constructor when we already have a Constructor defined in the class?

No, the compiler will not create the Default Constructor when we already have a Constructor defined.

Q7. What is the use of Private Constructors in Java?

When we use the private keyword for a constructor then an object for the class can only be created internally within the class, no outside class can create an object for this class. Using this we can restrict the caller from creating objects.

class ExampleOfPrivateConstructor

{

    /\*\*

     \* Private Constructor for preventing object creation

    from the  outside class

    \*\*/

    private ExampleOfPrivateConstructor (){ }

    public void display()

    {

        System.out.println("disp() method called");

    }

}

public class Sample

{

    public static void main(String args[])

    {

        //Creating the object for the Private Constructor class

        ExampleOfPrivateConstructor pc = new ExampleOfPrivateConstructor ();

        pc.display();

    }

}

When we will try to run the above code we will be getting the below exception.

Exception in thread "main" java.lang.Error: Unresolved compilation problem:

    The constructor ExampleOfPrivateConstructor () is not visible

    at Sample.main(Sample.java:21)