

20th Feb'23 OOPs Assignment - 18

1. What is Inheritance in Java?

Ans: The technique of creating a new class by using an existing class functionality is called inheritance in Java. In other words, inheritance is a process where a child class acquires all the properties and behaviors of the parent class.

2. What is superclass and subclass??

Ans: A class from where a subclass inherits features is called superclass. It is also called base class or parent class. A class that inherits all the members (fields, method, and nested classes) from another class is called a subclass. It is also called a derived class, child class, or extended class.

3. How is Inheritance implemented/achieved in Java?

Ans: Inheritance can be implemented or achieved by using two keywords: extends: extends is a keyword that is used for developing the inheritance between two classes and two interfaces.

implements: implements keyword is used for developing the inheritance between a class and interface.

4. What is polymorphism?

Ans: Polymorphism in OOP is the ability of an entity to take several forms. In other words, it refers to the ability of an object (or a reference to an object) to take different forms of objects. It allows a common data-gathering message to be sent to each class. Polymorphism encourages what is called 'extendibility' which means an object or a class can have its uses extended.

5. Differentiate between method overloading and method overriding.

Overriding:

- Implements "runtime polymorphism"
- The method call is determined at runtime based on the object type
- Occurs between superclass and subclass
- Have the same signature (name and method arguments)
- On error, the effect will be visible at runtime

20th Feb'23 OOPs Assignment - 18

Overloading:

- Implements "compile time polymorphism"
- The method call is determined at compile time
- Occurs between the methods in the same class
- Have the same name, but the parameters are different
- On error, it can be caught at compile time

6. What is an abstraction explained with an Example?

Ans: Abstraction is nothing but the quality of dealing with ideas rather than events. It basically deals with hiding the internal details and showing the essential things to the user.

```
Abstract class Sports { // abstract class sports
Abstract void jump(); // abstract method
}
```

7. What is the difference between an abstract method and final method in Java? Explain with an example

Ans: The abstract method is incomplete while the final method is regarded as complete. The only way to use an abstract method is by overriding it, but you cannot override a final method in Java.

8. What is the final class in Java?

Ans: A class declared with the final keyword is known as the final class. A final class can't be inherited by subclasses. By using the final class, we can restrict the inheritance of the class. We can create a class as a final class only if it is complete in nature, which means it must not be an abstract class. In java, all the wrapper classes are final classes like String, Integer, etc.

If we try to inherit a final class, then the compiler throws an error at compilation time. We can't create a class as immutable without the final class.

```
final class ParentClass
{
```

20th Feb'23 OOPs Assignment - 18

```
void showData()
{
    System.out.println("This is a method of final Parent class");
}
}
//It will throw compilation error
class ChildClass extends ParentClass
{
    void showData()
    {
        System.out.println("This is a method of Child class");
    }
}
class MainClass
{
    public static void main(String arg[])
    {
        ParentClass obj = new ChildClass();
        obj.showData();
    }
}
```

9. Differentiate between abstraction and encapsulation.

Abstraction:

- Abstraction is a feature of OOPs that hides the necessary details but shows the essential information.
- It solves an issue at the design level.
- If focused on the external lookout.
- It can be implemented using abstract classes and interfaces.
- It is the process of gaining information.
- In abstraction, we use abstract classes and interfaces to hide the code complexities.
- The objects are encapsulated that helps to perform abstraction.

Encapsulation:

20th Feb'23 OOPs Assignment - 18

- Encapsulation is also a feature of OOPs. It hides the code and data into a single entity or unit so that the data can be protected from the outside world.
- Encapsulation solves an issue at implementation level.
- It focuses on internal working.
- It can be implemented by using the access modifier (private, public, protected).
- It is the process of containing the information.
- We use the getters and setters methods to hide the data.
- The object need not to abstract that result in encapsulation

10. Difference between Runtime and compile time polymorphism explain with an example

Compile Time Polymorphism:

- Compile Time Polymorphism is less flexible as all things execute at compile time.
- In Compile Time Polymorphism, the call is resolved by the compiler.
- Inheritance is not involved.
- It is also known as Static binding, Early binding and overloading as well.
- It provides fast execution because the method that needs to be executed is known early at the compile time.
- Method overloading is the compile-time polymorphism where more than one method shares the same name with different parameters or signature and different return type.

Runtime Polymorphism:

- Run Time Polymorphism is more flexible as all things execute at run time.
- In Run Time Polymorphism, the call is not resolved by the compiler.
- Inheritance is involved.
- It is also known as Dynamic binding, Late binding and overriding as well.
- It provides slow execution as compared to early binding because the method that needs to be executed is known early at the runtime.

20th Feb'23 OOPs Assignment - 18

- Method overriding is the runtime polymorphism having the same method with same parameters or signature but associated with compared, different classes.