Inheritance, Polymorphism, Interfaces and String Handling

Inheritance, Polymorphism, Interfaces and String Handling

Inheritance

Deriving new classes from existing classes such that new classes acquire all the features of existing classes is called inheritance.

In Inheritance, a programmer reuses the super class code without rewriting it, in creation of sub classes. So, developing the classes becomes very easy. Hence the programmer's productivity is increased.

Using 'super' keyword - If we create an object to super class, we can access only the super class members, but not the sub class members. But if we create sub class object, all the members of both super and sub classes are available to it. This is the reason; we always create an object to sub class in inheritance. Sometimes, the super class members and sub class members may have same names. In that case, by default only sub class members are accessible.

The protected specifier - The private members of the super class are not available to sub classes directly. But sometimes, there may be a need to access the data of super class in the sub class. For this purpose, protected specifier is used. protected is commonly used in super class to make the members of the super class available directly in its sub classes.

Types of Inheritance:

1. **Single Inheritance -** Producing sub classes from a single super class is single called inheritance. In this, a single super class will be there. There can be one or more sub classes.
2. **Multiple Inheritance -** Producing sub classes from multiple super classes is called multiple inheritance. In this case, there will be more than one super class and there can be one or more sub classes.

Inheritance, Polymorphism, Interfaces and String Handling

Polymorphism

Polymorphism in Java is the ability of an object to take many forms. To simply put, polymorphism in java allows us to perform the same action in many different ways. Any Java object that can pass more than one IS-A test is considered to be polymorphic and in java, all the java objects are polymorphic as it has passed the IS-A test for their own type and for the class Object. There are two types of polymorphism in java: compile-time polymorphism and runtime polymorphism.

This article also talks about two types of polymorphism in Java: compile-time polymorphism and runtime polymorphism, java polymorphism examples, method overloading, method overriding, why to use polymorphism in java and many more.

Polymorphism is a feature of the object-oriented programming language, Java, which allows a single task to be performed in different ways. In the technical world, polymorphism in java allows one to do multiple implementations by defining one interface.

Inheritance, Polymorphism, Interfaces and String Handling

Interfaces

Like a class, an interface can have methods and variables, but the methods declared in an interface are by default abstract (only method signature, no body).

- Interfaces specify what a class must do and not how. It is the blueprint of the class.
- An Interface is about capabilities like a Player may be an interface and any class implementing Player must be able to (or must implement) move (). So, it specifies a set of methods that the class has to implement.
- If a class implements an interface and does not provide method bodies for all functions specified in the interface, then the class must be declared abstract.

To declare an interface, use interface keyword. It is used to provide total abstraction. That means all the methods in an interface are declared with an empty body and are public and all fields are public, static and final by default. A class that implements an interface must implement all the methods declared in the interface. To implement interface use implements keyword.

- It is used to achieve total abstraction.
- Since java does not support multiple inheritance in case of class, but by using interface it can achieve multiple inheritance.
- It is also used to achieve loose coupling.
- Interfaces are used to implement abstraction. So, the question arises why use interfaces when we have abstract classes?
  The reason is, abstract classes may contain non-final variables, whereas variables in interface are final, public and static.

Inheritance, Polymorphism, Interfaces and String Handling

String Handling

String is an object that represents sequence of characters. In Java, String is represented by String class which is located into java.lang package

It is probably the most commonly used class in java library. In java, every string that we create is actually an object of type String. One important thing to notice about string object is that string objects are immutable that means once a string object is created it cannot be changed.

In Java, CharSequence Interface is used for representing a sequence of characters. CharSequence interface is implemented by String, StringBuffer and StringBuilder classes. These three classes can be used for creating strings in java.

Inheritance, Polymorphism, Interfaces and String Handling