Abstract Class

- Restricts the creation of an object of an abstract class.
- Keyword *abstract* is used to create an abstract class.
- Abstract Class can have both Normal as well as Abstract Method.
- Abstract Method's body is not defined in abstract class.
- If any of method in abstract class is abstract method, it needs to be defined in sub class.
- Normal method does not needs to be re-defined in subclass.

Abstract Method

- If we declare a method as abstact method, if cannot provied the method definition, that is we can only declare the method and that is it. We cannot proved what it does.
- The abstract needs to be defined in inherited sub class, without *abstract* keyword.
- If a method is abstract, the class must be an abstract class.

Why Abstract Class:

- Restricts ceration of obejcts on abstract class.
- Allows access to subclass members using reference to abstract class.

Final

1. Final Varaible
2. Final Method
3. Final Class

Final Variable

- Creats a constant
- Once a value is assigned, it cannot be changed

Final Class

- A class can also be made final
- If a class is final, it cannot be inherited.

Final Method

- Methods can also be made final.
- It restircts method overriding.

Interface

- Interface is a type of abstract class, where all the methods are abstract
- $interface$ keyword is used to create an interface.
- We do not need to add abstract keyoword to the methods
- Methods are by default public abstract
- Variable inside an interface become final
- Advantages:
  - Allows multiple inheritance, class can implement multiple interfaces
  - All the methods are by default public abstract
- Difference form abstract class is that, in abstract class we can define a method but in inheritance we cannot define a method but only declare it.
- From java 1.8 and forward, Intefrace method definition is allowed using keyword $default$.
- Interface can be implemented by class using $implements$ keyword.
- Anonymous inner class can also be used for interface.

Types of Interfaces

- Noramal Interface - More than 1 methods
- Single Abstract Method Interface(Functional Interface) - Only 1 method
- Marker Interface - No method at all

Functional Interface (SAM Interface)

- Only one $abstrct$ method is allowed.
- Allows creating Lambda Expressions

Multiple Inheritance using Interface

- To remove ambiguity, we define the method in the implementing class.
- We can also call the methods of interface using $interFaceName.super.methodName()$

```
// A simple Java program to demonstrate multiple
// inheritance through default methods.
interface PI1
{
        // default method
        default void show()
        {
                System.out.println("Default PI1");
        }
}

interface PI2
{
        // Default method
        default void show()
        {
                System.out.println("Default PI2");
        }
}
// Implementation class code
class TestClass implements PI1, PI2
{
        // Overriding default show method
        public void show()
        {
                // use super keyword to call the show
                // method of PI1 interface
                PI1.super.show();

                // use super keyword to call the show
                // method of PI2 interface
                PI2.super.show();
        }

        public static void main(String args[])
        {
                TestClass d = new TestClass();
                d.show(); } }
                //OUTPUT:Default PI1
                //      Default PI2
```

- In the case of above example, We cannot remove the overriding method from implementing class.
- If we do remove than complie-time error will be generated, this is because the methods are already defined in both Interface and create

ambiguity problem.