OOP

Interface and Abstraction

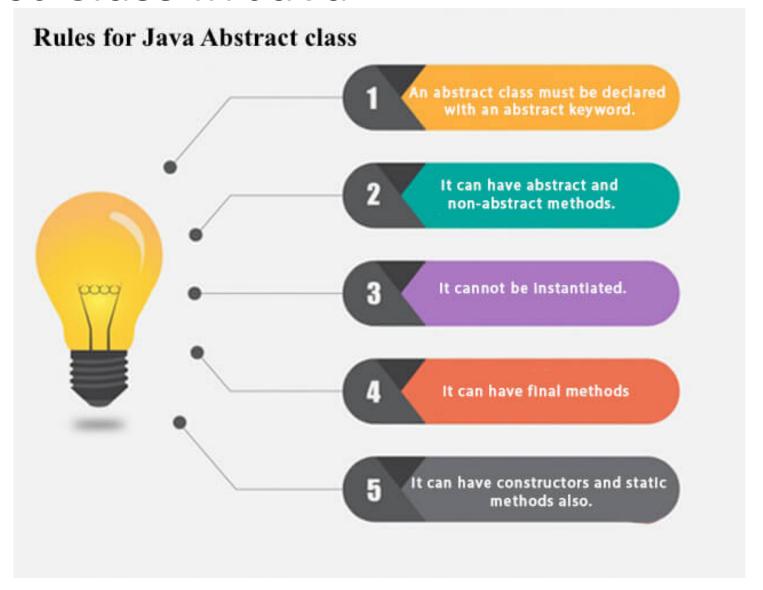
Abstraction

- is a process of hiding the implementation details and showing only functionality to the user
- lets you focus on what the object does instead of how it does it

Abstract Class in Java

• A class which is declared as abstract is known as an **abstract class**. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

Abstract Class in Java



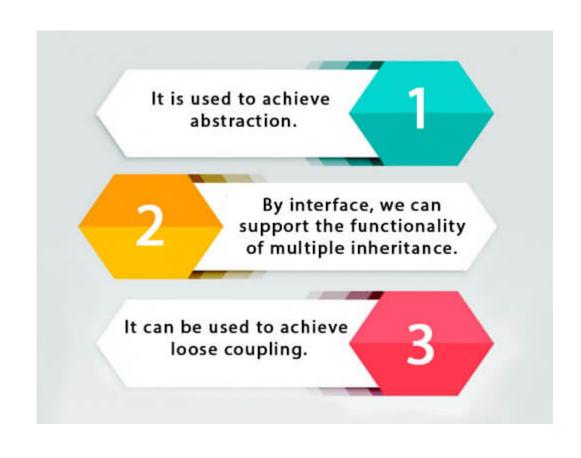
Example of Abstract Class

```
//Example of an abstract class that has abstract and non-abstract methods
abstract class Bike{
  Bike(){System.out.println("bike is created");}
  abstract void run();
 void changeGear(){System.out.println("gear changed");}
//Creating a Child class which inherits Abstract class
class Honda extends Bike{
void run(){System.out.println("running safely..");}
//Creating a Test class which calls abstract and non-abstract methods
class TestAbstraction2{
public static void main(String args[]){
 Bike obj = new Honda();
 obj.run();
 obj.changeGear();
```

Interface

- The interface in Java 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.
- It cannot be instantiated just like the abstract class.

Interface



Example of Interface

```
//Interface declaration: by first user
interface Drawable{
void draw();
//Implementation: by second user
class Rectangle implements Drawable{
public void draw(){System.out.println("drawing rectangle");}
class Circle implements Drawable{
public void draw(){System.out.println("drawing circle");}
//Using interface: by third user
class TestInterface1{
public static void main(String args[]){
Drawable d=new Circle();//In real scenario, object is provided by method e.g. getDrawable()
d.draw();
```