Class and Object

IT22020

```
class Dog{
    String name;
    int age;
   void bank(){
        System.out.println(name + "is barking!");
        }
    }

~~Pa~~ public class classANDobject{
    public static void main(String[] args){

        Dog ~~*~~ myDog = new Dog();
        myDog.name ="Buddy";
        myDog.age =3;
        my Dog.bank();
    }
}
```

Access Modifier

IT22020

```
class BankAccount{
    ~~Pr~~private double balance;
  public void deposit (double amount){
        if (amount >0){
          bblance+ = amount;
        System.out.println("Deposited : $"+
                                   amount);
        }
      }

public void withdraw (double amount){
      if (amount > 0 && amount <= balance){
        balance -= amount;
     System.out.println("withdraw: $"+amount);
      } else
      system.out.println("Insufficient balance
          or ivalid amount!");
     }

 public ~~class~~ double getBalance(){
              return Balance;
           }
```

IT22020

~~Inter~~

Inheritance and Protected Access

```
class Vehicle {
    protected String brand = "Generic vehicle";

    void start() {
        System.out.println(brand + " is starting...");
    }
}

Class Bike extends vehicle{
    void ride () {
    System.out.println ("Riding the"+ brand);
    }
}

public class Inheritance_and_Protected_Access{
    public static void main (string [] args){
        Bike myBike = new Bike();
        my Bike.start();
        my Bike.ride();
    }
}
```

Encapsulation

```
class Student{
        private int age;
        private string name;
public void setName (String newName){
        name = newName;
  }
public String getName (){
          return name;
  }
public void setAge (int newAge){
          if (newAge >0){
             age = newAge;
          }
      }
public int getAge(){ return ~~A~~ age;}}
Public class Encaptulation {
       public static void main(String[] args){
         Student  s=new Student();
         s.setName ("John");
         s. set Age (28);
System.out.println("Name :"+ s.getName());
System.out.println ("Age :"+ s.getAge());
    }
}
```

Abstract Class

```
abstract class Shape{

        abstract void draw();

    void info(){
         System.out.println("This is a shape.");
         }
    }}
class Circle extends Shape{
         void draw(){
         System.out.println("Drawing a circle.");
    }}
public class abstruct class{
             public static void main(String[] args){
         Circle c = new circle();
              c.draw();   c.info(); }}
```

Interface   IT22020

```
interface vehicle1{
            void start();}

class  Car implements vehicle1{
        publice void start(){
          System.out. println("car is starting...");
         }
    }

public class interface1{
    public static void main(String[] args){
     Car myCar = new Car();
       myCar.start();
     }
  }
```