IT22018

Classes and Objects:

```
Class car {
    String color;
    int speed;
    void drive() {
        System.out.println("Car is driving....");
    }
}
public class Main {
    Public static void main(String[] arg) {
        Car myCar = new Car();
        my Car.color = "Red";
        my car.speed = 100;
        my car.drive();
    }
}
```

Acces Modifiers:

```
class person {
    private String name;
    public void setName(String newName) {
        name = newName;
    }
    public String getName() {
        return name;
    }
}
Public class Main {
    public static void main(String[] args) {
        person p = new person();
        p.setName("Alice");
        System.out.println(p.setName());
    }
}
```

I 7/2/2018

Inheritance and protected Access:

```
Class Animal {
    protected String type = "Animal";
    Void display() {
        System.out.println("This is an animal.");
    }
}

Class Dog extends Animal {
    Void bark() {
        System.out.println(type + "says woof!");
    }
}

public Class Main {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.display();
        d.bark();
    }
}
```

Encapsulation:

```
Class BankAccount {
    private double balance;
    Public Void deposit(double amount) {
        if(amount > 0) balance += amount;
    }
    public double getBalance() {
        return balance;
    }
}
```

IT22018

```
Public class Main {
    public static void main(String[] args) {
        BankAccount acc = new BankAccount();
        acc.deposit(500);
        System.out.println(acc.getBalance());
    }
}
```

Abstract classes:

```
abstract class Animal {
    abstract void makeSound();
    void sleep() {
        System.out.println("Sleeping....");
    }
}

class Dog extends Animal {
    void makeSound() {
        System.out.println("Bark Bark");
    }
}

public static void main(String[] args) {
    Dog d = new Dog();
    d.makeSound();
    d.sleep();
}
```

IT22018

Interface:

```
interface Animal{
    void sound();
}
Class Cat implements Animal{
    public void sound(){
        System.out.println("Meow");
    }
}
public class Main{
    public static void main(String[] args){
        Cat c = new Cat();
        c.sound();
    }
}
```

Multiple Inheritance using Interface:

```
interface Flyable{
    void fly();
}
interface Swimmable{
    void swim();
}
Class Duck implements Flyable, Swimmable{
    public void fly(){
        System.out.println("Duck is flying...");
    }
```

IOT 22028

```
Public void swim(){
        System.out.println("Duck is swimming...");
    }
}
public class Main{
    public static void main(String[] args){
        Duck d = new Duck();
        d.fly();
        d.swim();
    }
}
```