Name: Hossain Mohammad
IT-22056

① Classes And Object

```
class Bike{
    String brand;
    int speed;
void ride(){
    System.out.println("Bike is riding...");
  }
}
```

; create a class called bike
; string variable called brand which store the brand name (Yamaha....)

```
public class Main{
    public static void main(String[] args){
    Bike mybike = new Bike();
    myBike.brand = "Yamaha";
    myBike.speed = 80;
    myBike.ride();
    }
}
```

Create a object for bike class)
Set the brand name Yamaha in the bike. object

2. Access Modifiers

```
class student {
    private int rollnumber;
    public void setrollnumber(int newroll){
    rollNumber = newRoll; }
public int getRollNumber(){
    return rollNumber;
}}
    public class Main {
        public static void main(String[] args){
        student s = new student();
        s.setRollnumber(101);
System.out.println(s.getRollNumber());
}
}
```

create student class
private int means it can only accessed inside the class

setrollnumber can be set from outside because it's public int.

3. Inheritance

```
class Vehicle {
    protected String brand = "Vehicle";
    void start() {
        System.out.println("Vehicle is starting");
    }
}
class Bike extends Vehicle {
    void ride() {
        System.out.println(brand + " is being ridden");
    }
}
public class Main {
    public static void main(String[] args) {
        Bike myBike = new Bike();
        mybike.start();
        mybike.ride();
    }
}
```

4. Encapsulation

```
class wallet {
    private double money;
    public void addMoney(double amount) {
    if (amount > 0) {
    money += amount; } }
public double checkBalance() {
 return money ; } }
public class main {
        public static void main(String[] args) {
    wallet mywallet = new wallet();
    mywallet.addmoney(300);
    System.out.println("Current Balance: " + mywal
                                    .checkBalance()
}
}
```

5. Abstract Class:-

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

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

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

6. Interface

```
interface Animal {
    void sound();
}
class Dog implements Animal {
    public void sound() {
        System.out.println("woof");
    }
}
public class Main {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.sound();
    }
}
```

7. Multiple inheritance using interface

```
interface Flyable {
```

→ this create an interface named flyable

```
    void fly();
}
```

```
interface Swimable {
    void swim();
}
```

→ Another interface Swimable is defined

→ Any class implementing it must provide a swim() method.

```
class Seagull implements Flyable, Swimable {
```

→ A class seagull is created

→ Seagull implements inherit from both flyable and Swimable.

```
    public void fly() {
        System.out.println("Seagull is flying over ..");
    }
```

→ Here we implement fly() method

→ when fly() is called, it prints

"Seagull is flying over"

```
public class swim (){
    System.out.println("Seagull is swimming");
}
```

→ Here we implement swim() method

→ when swim() method is called it prints ....

```
class Penguin implements Swimable {
```

→ A new class Penguin is created

→ Penguin only implements Swimable because it cant fly

→ So, it needs to implement only swim() method.

```
public void swim(){
    System.out.println("Penguin is only swimmin
    in cold water");
}
```

```
public class main {

    public static void main(String[] args) {

        Seagull seagull = new Seagull();
```

→ Create a new object of type Seagull

```
        Penguin penguin = new Penguin();
        seagull.fly();
```

→ call the fly() method on the seagull object

→ It prints "Seagull is flying ore"

```
        seagull.swim();
```

→ call the swim method on seagull object.

→ It prints("Seagull is swimming");

```
        Seag penguin.swim();
```