Name: Abdul Hamid Piash
ID: IT22021

## 1. Classes and Objects

Code:

```
class Car{
   String color;
   int speed;

   void drive(){
      System.out.println("Car is driving....");
   }
}
public class Main{
   public static void main(String[] args){
      Car mycar = new Car();
      myCar.color = "Red";
      myCar.speed = 100;
      myCar.drive();
   }
}
```

## 2. Access Modifiers

```
class Person{
   private String name;
   public void setName(String name){
      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.getName());
    }
}
```

3. 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!");
    }
}
```

IT22021

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

4. Encapsulation

```
class BankAccount{
    private double balance;
    public void deposit(double amount){
        if (amount > 0) balance += amount;
    }
    public double getBalance(){
        return balance;
    }
}
public class Main{
    public static void main(String[] args){
        BankAccount acc = new BankAccount();
        acc.deposit(500);
        System.out.println(acc.getBalance());
    }
}
```

IT22021

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

IT22021

## 6. 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();
    }
}
```

## 7. Multiple Inheritance using Interface

```
interface Flyable {
    void fly();
}
interface Swimmable {
    void swim();
}
```

IT22021

```
class Duck implement Flyable, Swimmable{
    public void fly(){
        System.out.println("Duck is flying....");
    }
    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();
    }
}
```