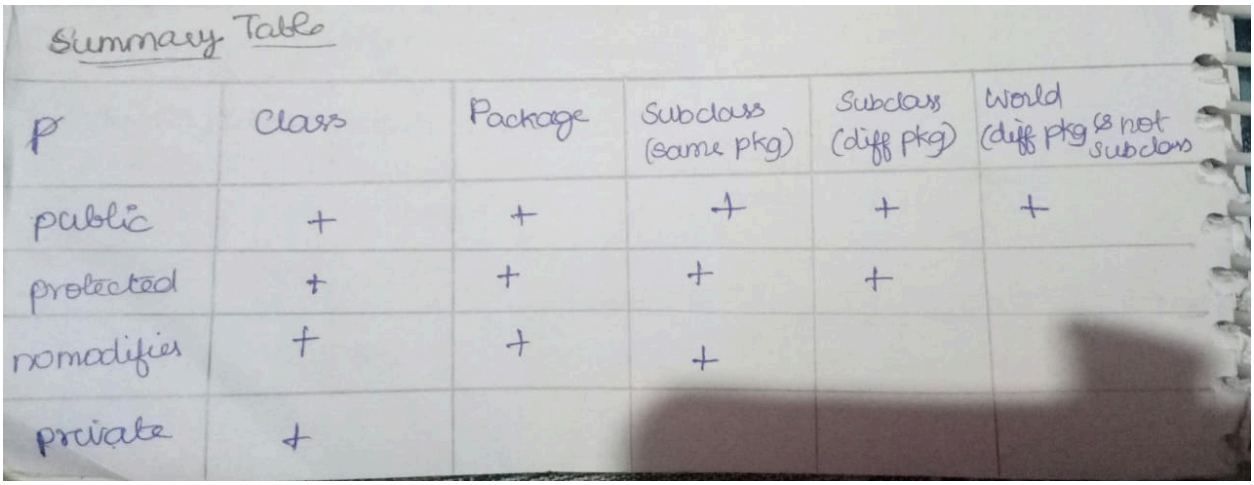
3️⃣

Encapsulation(Detailed notes)

Encapsulation is a fundamental principle of object-oriented programming (OOP) that refers to the **binding of data (variables) and methods (functions)** that manipulate the data into a single unit (class), while **restricting direct access** to some of the object's components using **access modifiers** to enforce controlled access.

🔐 Access Specifiers in Java

Access Specifiers (Modifiers) define the **scope and visibility** of classes, methods, and variables in Java. They help enforce **Encapsulation** by controlling access levels.



1.Public access Modifier : Accessible Everywhere

```
public class A { //Example of public access modifier
    int num;
```

```
    String name;
    int[] arr;

    // Constructor (must be outside any method)
    A(int num, String name, int[] arr) {
        this.num = num;
        this.name = name;
        this.arr = arr;  // or new int[0] if you want to initialize as empty
    }

    public static void main(String[] args) {
        A obj = new A(24, "Ayshu", new int[]{34});
        System.out.println("Num: " + obj.num);
        System.out.println("Name: " + obj.name);
        System.out.println("Array value: " + obj.arr[0]);
    }
}
```

2. Private access modifier： Accessible only within its own class

(Still accessing it from other classes using public getters and setter methods)

```
//Private access Modifier
public class b{
    private int num;
    String name;
    int[] arr;
    public int getnum(){
        return num;
    }
    public void setnum(int num){
        this.num=num;
    }
```

```
    public b(int num,String name){
        this.num=num;
        this.name=name;
    }
}
```

3. Protected: Accessible everywhere similar to public but the only difference is not accessible from Non-subclass of other packages

Person.java (Class containing the protected)

```
public class Person{
    protected String name;
    protected void displayName(){
        System.out.println("Name: "+name);
    }
}
```

Student. java (Class accessing the protected method)

```
public class Student extends Person{
    public Student(String name){
        this.name=name;
    }
    public void showname(){
        displayName();
    }
}
```

Main.java

```
public class Main{
     public static void main(String[] args) {
        b obj=new b(24,"Ayshu");
        //Accessing the private member using a Getter Setter
```

```
        System.out.println(obj.getnum());
        //Method accessing the protected method
        Student s1=new Student("Abi Ayshwariya S");
        s1.showname();

    }
}
```