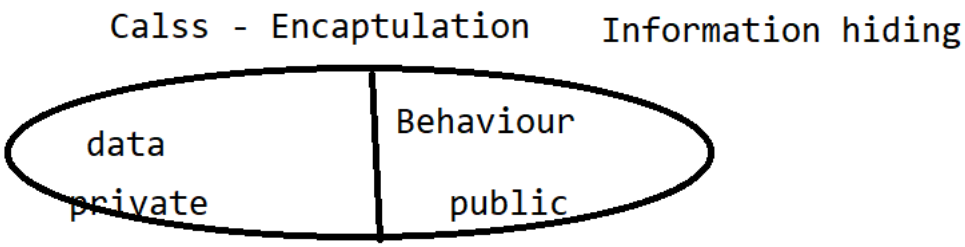
Daily Notes

Lesson-3

Mutable class

```
public class IntHolder {
private int value;
public IntHolder(int  value) {
this.value = value;
}
   public int getValue()  {
    return value;
    }
   public String SetValue(int value)   {
      this.value = value;
   }
}
// main method
IntHolder holder = new IntHolder(10);
int v = holder.getValue(); //  return 10
Holder.SetValue(20);
v = holder.getValue(); //  return 20
```



Creation of Class

1. How to create a class
2. How to add data members/attributes/instance fields/properties
3. How to create a constructor
4. Default constructor, Parameterized constructor and its rules
5. Access Modifiers
6. Constructor Overloading
7. How to create methods / behavior / instance methods
8. this keyword
9. How to create an objects using new keyword, assigning objects
10. Array of objects
11. Processing the collection of objects
12. Accessing methods
13. Getters and setters

Use Final Keyword

final class Sample{} – If you make your class as final – you cannot inherit

final int x = 12; → You cannot modify the value

final void test() -- > Method as final, you cannot override

Boxed Primitives

```
public class BoxedPrimitiveDemo {
      private int a ; // Instance variables

     public static void main(String[] args) {
           int x = 10; // Primitive , Local Variable
           // All BoxedPrimitives are Immutable
           Integer b = 20; // Boxed Primitive / Reference
Type
           int cr = b.compareTo(12);
           System.out.println(cr);
           System.out.println(Integer.max(13, 15));
           Integer c = x; // Auto boxing - Converting
primitive to Boxed is called Autoboxing
```

```
            String x1 = "12.45";
            // Convert the String input to double
            Double d = Double.parseDouble(x1);
            System.out.println(d);
            Float f=null;
            System.out.println(f.toString());

    }

}
```

A Program can have different blocks

1. Static block
2. Static methods, static fields
3. Instance block { } – Execute for all instances
4. Instance methods and fields
5. Constructors

    {

    }