Department: BCA
Faculty Name: Er. Ashish Kumar
Subject: JAVA (5th sem.)
Topic: Abstraction in java **(part 2)**

Encapsulation in Java

Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. Others way to think about encapsulation is, it is a protective shield that prevents the data from being accessed by the code outside this shield.

- Technically in encapsulation, the variables or data of a class is hidden from any other class and can be accessed only through any member function of own class in which they are declared.
- As in encapsulation, the data in a class is hidden from other classes, so it is also known as **data-hiding**.
- Encapsulation can be achieved by: Declaring all the variables in the class as private and writing public methods in the class to set and get the values of variables.

Advantages of Encapsulation:

- **Data Hiding:** The user will have no idea about the inner implementation of the class. It will not be visible to the user that how the class is storing values in the variables. He only knows that we are passing the values to a setter method and variables are getting initialized with that value.
- **Increased Flexibility:** We can make the variables of the class as read-only or write-only depending on our requirement. If we wish to make the variables as read-only then we have to omit the setter methods like setName(), setAge() etc. from the above program or if we wish to make the variables as write-only then we have to omit the get methods like getName(), getAge() etc. from the above program
- **Reusability:** Encapsulation also improves the re-usability and easy to change with new requirements.
- **Testing code is easy:** Encapsulated code is easy to test for unit testing.

Example:

*C:\Users\ashishjha\Desktop\java\Student.java - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

setbackgroundimage.html | Student.java

```
1  //Java program to demonstrate encapsulation
2  public class Student{
3      private String name;
4      private int roll;
5      private int age;
6      public int getAge()
7      {
8        return age;
9      }
10     public String getName()
11     {
12       return name;
13     }
14     public int getRoll()
15     {
16       return roll;
17     }
18     public void setAge( int age)
19     {
20       this.age = age;
21     }
22     public void setName(String name)
23     {
24       this.name = name;
25     }
26     public void setRoll( int roll)
27     {
28       this.roll=roll;
29     }
30 }
31 class TestStudent{
32 public static void main (String[] args) {
33         Student obj = new Student();
34         obj.setName("Rahul");
35         obj.setAge(19);
36         obj.setRoll(51);
37         System.out.println("Student Name: " + obj.getName());
38         System.out.println("Student Age: " + obj.getAge());
39         System.out.println("Student Rollno: " + obj.getRoll());
40     }
41 }
```

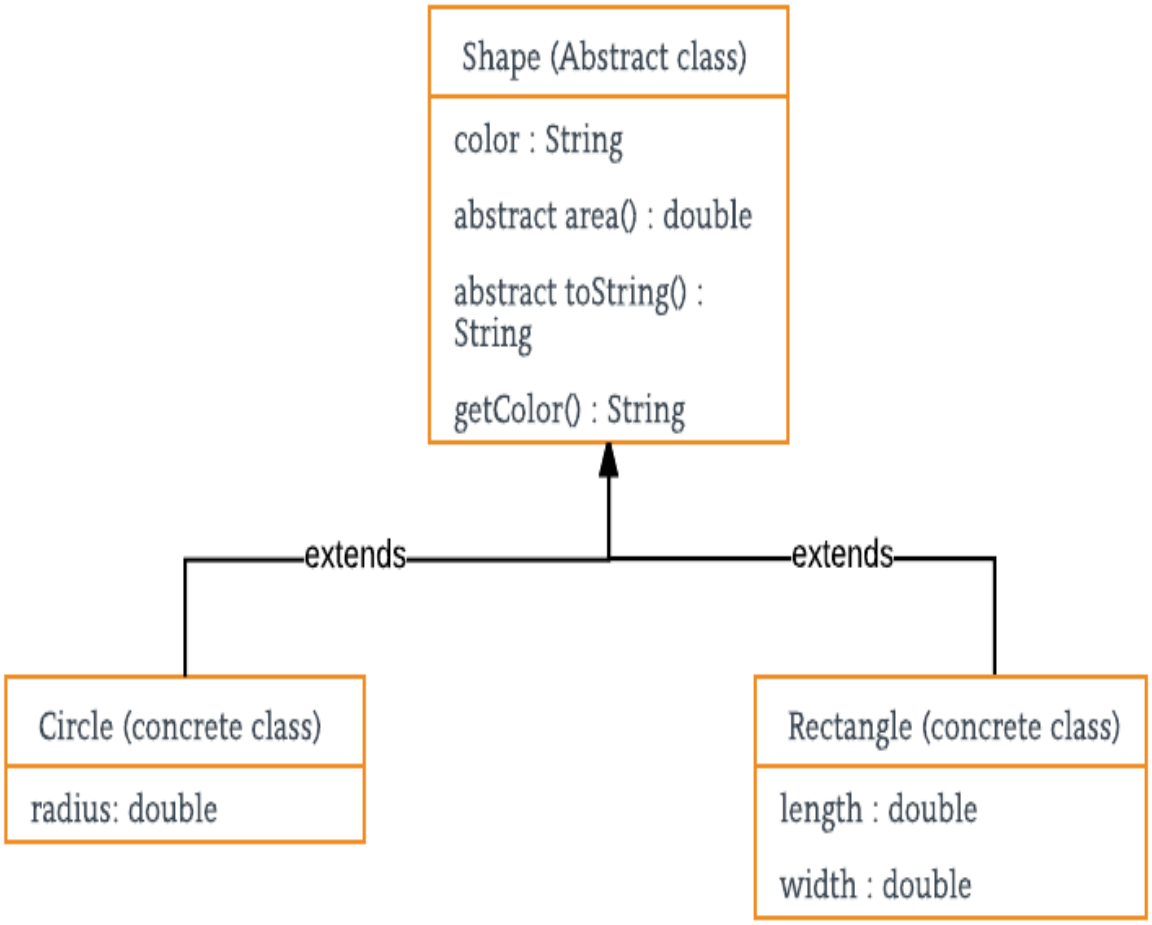
Command Prompt

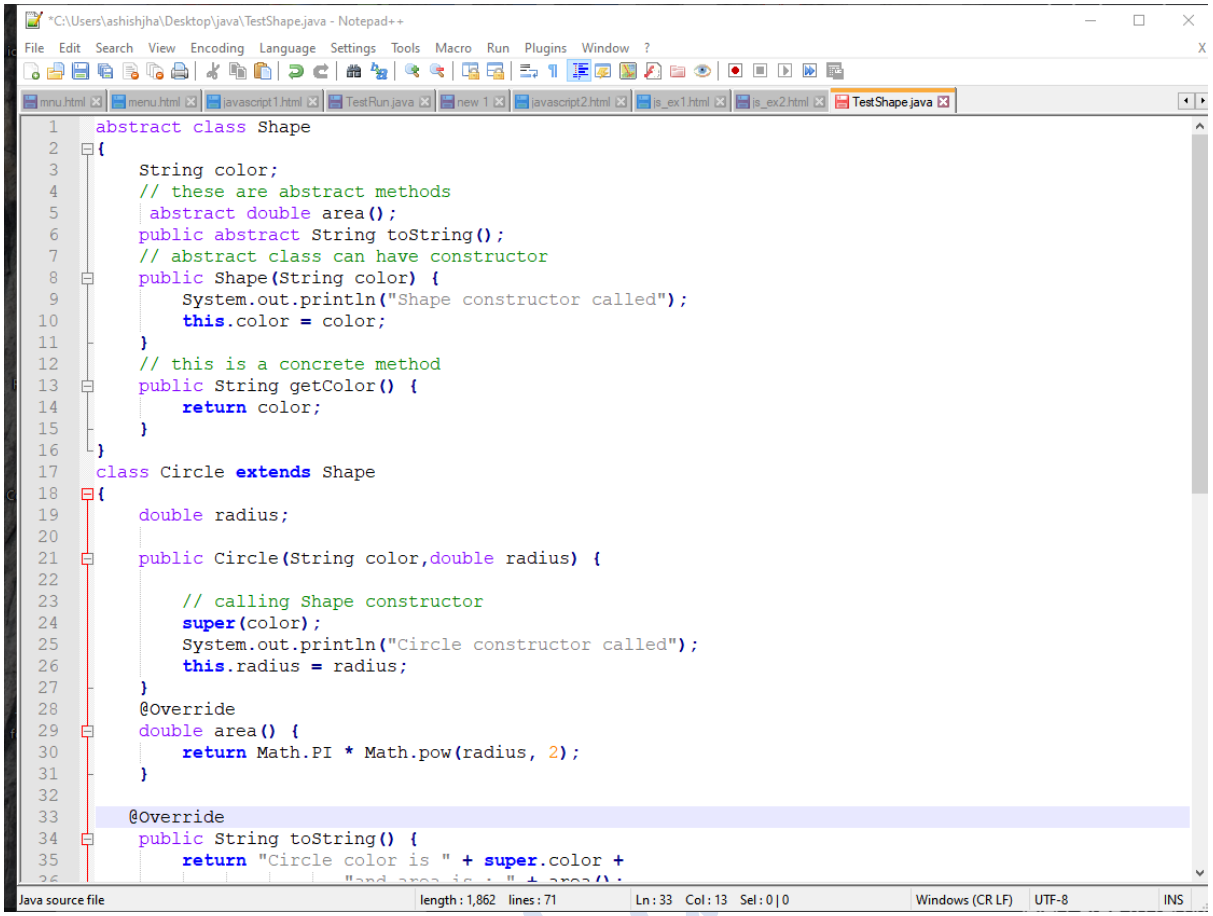
```
C:\Users\ashishjha\Desktop\java>javac Student.java

C:\Users\ashishjha\Desktop\java>java TestStudent
Student Name: Rahul
Student Age: 19
Student Rollno: 51

C:\Users\ashishjha\Desktop\java>
```

Encapsulation vs Data Abstraction:

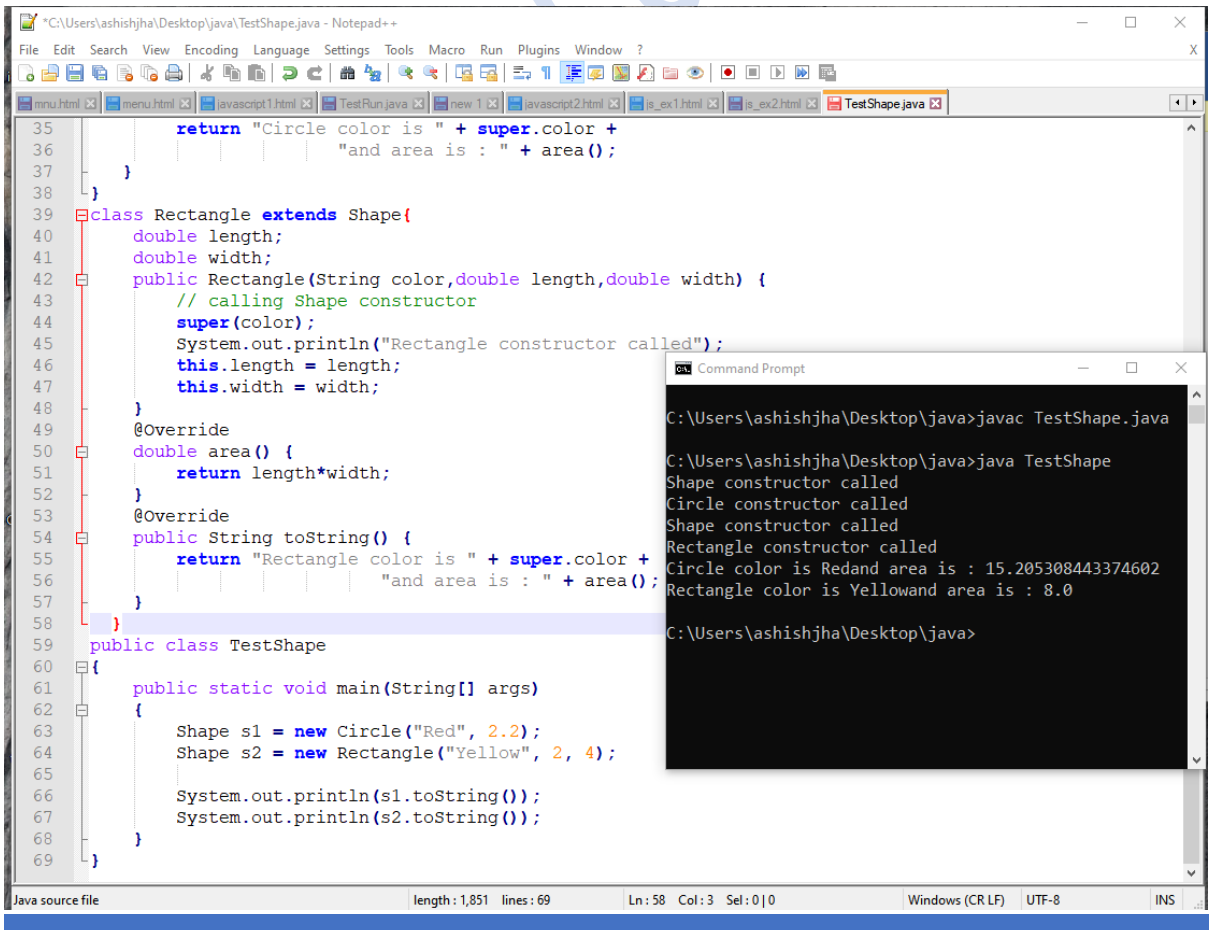
1. Encapsulation is data hiding (information hiding) while Abstraction is detail hiding (implementation hiding).
2. While encapsulation groups together data and methods that act upon the data, data abstraction deals with exposing the interface to the user and hiding the details of implementation.

Example of Abstract class And Abstract Method:





```abstract class Shape
{
    String color;
    // these are abstract methods
     abstract double area();
    public abstract String toString();
    // abstract class can have constructor
    public Shape(String color) {
        System.out.println("Shape constructor called");
        this.color = color;
    }
    // this is a concrete method
    public String getColor() {
        return color;
    }
}
class Circle extends Shape
{
    double radius;

    public Circle(String color,double radius) {

        // calling Shape constructor
        super(color);
        System.out.println("Circle constructor called");
        this.radius = radius;
    }
    @Override
    double area() {
        return Math.PI * Math.pow(radius, 2);
    }

   @Override
    public String toString() {
        return "Circle color is " + super.color +
```



```        return "Circle color is " + super.color +
                "and area is : " + area();
    }
}
class Rectangle extends Shape{
    double length;
    double width;
    public Rectangle(String color,double length,double width) {
        // calling Shape constructor
        super(color);
        System.out.println("Rectangle constructor called");
        this.length = length;
        this.width = width;
    }
    @Override
    double area() {
        return length*width;
    }
    @Override
    public String toString() {
        return "Rectangle color is " + super.color +
                "and area is : " + area();
    }
  }
public class TestShape
{
    public static void main(String[] args)
    {
        Shape s1 = new Circle("Red", 2.2);
        Shape s2 = new Rectangle("Yellow", 2, 4);

        System.out.println(s1.toString());
        System.out.println(s2.toString());
    }
}
```

```
C:\Users\ashishjha\Desktop\java>javac TestShape.java

C:\Users\ashishjha\Desktop\java>java TestShape
Shape constructor called
Circle constructor called
Shape constructor called
Rectangle constructor called
Circle color is Redand area is : 15.205308443374602
Rectangle color is Yellowand area is : 8.0

C:\Users\ashishjha\Desktop\java>
```

Advantages of Abstraction:

- It reduces the complexity of viewing the things.
- Avoids code duplication and increases reusability.
- Helps to increase security of an application or program as only important details are provided to the user.