# Department of Software Engineering

**SE-210: Software Design and Architecture**

**Class: BESE 9AB**

**Lab 09: Structural Design Patterns – II**

**CLO2:**

# Instructor: Dr. Hasan Ali Khattak

# 

**Ahmed Hassan Ismail – BESE-9B – 237897**

**Lab 09: Structural Design Patterns – II**

**Introduction**

Students will have hands-on experience of implementing a structural design pattern to a given problem.

**Objectives**

This objective of this lab is to get a practical understanding and knowledge of the Flyweight Design Pattern. After the completion of this lab, students will be able to apply Flyweight design pattern to a given scenario.

**Tools/Software Requirement**

* Papyrus/Rational Rose

**Description**

In software engineering, structural design patterns are design patterns that ease the design by identifying a simple way to realize relationships between entities. Examples of Structural Patterns include: Adapter pattern: 'adapts' one interface for a class into one that a client expects.

**Lab Task**

**Task 1**

In this lab you have to extend this work as given below:

1. There will be one interface namely IShapeInterface with a single draw function having the following signature:

public void draw(String graphicType,

int x, int y,

int width, int height,

String color, boolean fill

);

1. There will be eight different types of concrete shapes implementing IShapeInterface: rectangle, triangle, square, circle, oval, and parallelogram, hexagon, sphere.
2. There will be a ShapeFactory Class which will have a getShape(String label) function. Given a label, the function will first check if the label exists within a HashMap. If it exists, the object will be returned. Otherwise, it will create a relevant object, put it in the HashMap and will return it.

Important: You have to make sure that factory only return on object and it does not create multiple objects of the same class.

1. The draw() function within each concrete shape will simply output the value of each variable using cout() or system.out.println() as shown below:

Shape drawn with following parameters: X: 1512 Y: 1552 Width: 51 Height: 210 Color: Red, Fill: false Type: O

1. The client class will have following logic
   1. Get a random shape between eight different shapes.
   2. Get random x and y and random width and height
   3. Get random color and fill
   4. Using ShapeFactory, get object you have obtained at (a)
   5. Call the draw method to print the output
   6. Repeat this 100 times
   7. Make sure only eight objects are created by outputting shape HashTable size.

The client code is as follows:

public class ClientLight {

private static final int WIDTH = 1800, HEIGHT = 1600;

//rc=rectangle, tr=triangle, sq=square, ci=circle, ov=oval, pa=parallelogram, he=hexagon, sp=sphere

private static final String shapes[] = { "rc", "tr", "sq", "ci", "ov", "pa", "he", "sp"};

private static final Color colors[] = { Color.red, Color.green, Color.blue, Color.darkGray };

private static final boolean fill[] = { true, false };

public ClientLight() {

int i;

for(i = 0; i < 1000; ++i) {

**//Your code for ShapeFactory and generating random values for different variables**

}

}

private String getRandomShape() {return shapes[(int) (Math.random() \* shapes.length)];}

private int getRandomX() {return (int) (Math.random() \* WIDTH\*0.9);}

private int getRandomY() {return (int) (Math.random() \* HEIGHT\*0.8);}

private int getRandomWidth(){return (int) (Math.random() \* (WIDTH / 7));}

private int getRandomHeight() {return (int) (Math.random() \* (HEIGHT / 7));}

private Color getRandomColor() {return colors[(int) (Math.random() \* colors.length)];}

private boolean getRandomFill() {return fill[(int) (Math.random() \* fill.length)];}

public static void main(String[] args) {

ClientLight client = new ClientLight();

}

}

Based on the scenario given above, your tasks are as follows:

1. Draw UML Class diagram of the solution – after applying Flyweight design pattern. Fully functional code of the solution. Use the client code given above.
2. You must be able to demonstrate your code in next lab.

**Answer:**

|  |
| --- |
| Solution |
| UML Class Diagram    Screenshots of the output:      Source Code: Zip your source code and upload it as well. |

**Deliverables**

Compile a single word document by filling in the solution part and submit this Word file on LMS. This lab grading policy is as follows: The lab is graded between 0 to 10 marks. The submitted solution can get a maximum of 5 marks. At the end of each lab or in the next lab, there will be a viva related to the tasks. The viva has a weightage of 5 marks. Insert the solution/answer in this document. You must show the implementation of the tasks in the designing tool, along with your completed Word document to get your work graded. You must also submit this Word document on the LMS. In case of any problems with submissions on LMS, submit your Lab assignments by emailing it to **Sundas Dawood** <sundas.dawood@seecs.edu.pk>