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

**DESIGN PATTERNS**

CLIENT

|  |
| --- |
| Shape Factory |
| getShape() |

**CALLSLS**

|  |
| --- |
| 🡨Interface🡪 |
| IMYshape |
| Drawgraphics() |

**implements**

**implements**

|  |
| --- |
| **MyOval** |
| Drawgraphics() |

|  |
| --- |
| **MyRectangle** |
| Drawgraphics() |

**Explanation:**

This is a simple design pattern which is implementing a solution to a problem.

The problem is that we have to create shapes by just creating one object for the shapes.   
For example, in a game if we want to create 100 cars, it is best to create a single interface then use its properties to create 100 cars shapes.

This solution implements the same concept.

**Step 1:**

Clients object gets created.

**Example code:**

Public Class Client

{

Public Client () {};

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

Private static final String shapes[] = { "R", "O" };

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

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

Private static final String font[] = { "Arial", "Courier" };

//Write the remaining code according to your understanding

}

**Step 2:**

Client calls the Shape Factory which selects a shape (rectangle or circle) and puts it in the IMYShape object.

**Example code:**

This statement is used to do so:

**IMyShape shape = ShapeFactory.getShape (getRandomShape ());**

Where the interface for IMYShape is defined as:

###################################################################

Public interface IMyShape {

Public void draw (Graphics g,

Int x, int y,

Int width, int height,

Color color, Boolean fill,

String font);

}

###################################################################

This interface is implemented by Rectangle and Oval through:

Public class MyRectangle implements IMyShape

{

//Write your draw function by using the template of IMYShape

}

Public class MyOval implements IMyShape

{

//Write your draw function by using the template of IMYShape

}

###################################################################

Shape Factory class has only one function “getShape ()”

This function checks whether the label is set to R or O and hence decides the shape

Public class ShapeFactory {

Private static final HashMap shapes = new HashMap ();

Public static IMyShape getShape (String label)

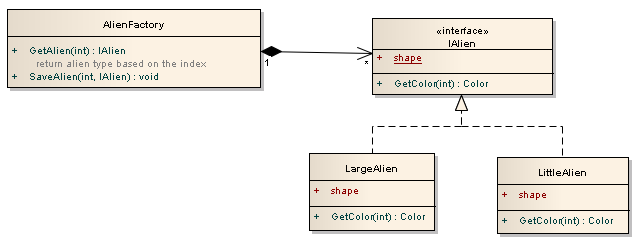
{

//Write your code here

}

}

**Example of Inheritance and Design Pattern in java:**



This example explains the flyweight design pattern as Alienfactory calls the IAlien which is a common interface for an object. Two classes Large Alien and LittleAlien implement this interface hence creating one object at a time of either a LargeAlien or a LittleAlien.

This is similar to our example as we have a shape factory which calls the IMyShape which is a common interace for an object. Two classes MyRectangle and MyOval implement this interface hence creating one shape at a time of either a rectangle or an oval.

**TASK:**

* You have to write a program where random shape is not printed instead you have to print a rectangle and an oval one after the other.
* You have to create a (rectangle, square, line, oval) of different sizes in the four corners of the screen.

BEST OF LUCK

|  |
| --- |
| **Code** |
| public interface Shape {      void draw();  }  public class Rectangle implements Shape {      @Override      public void draw() {          System.out.println("Inside Rectangle::draw() method.");      }  }  public class Square implements Shape {      @Override      public void draw() {          System.out.println("Inside Square::draw() method.");      }  }  public class Circle implements Shape {      @Override      public void draw() {          System.out.println("Inside Circle::draw() method.");      }  }  public class ShapeFactory {      // use getShape method to get object of type shape      public Shape getShape(String shapeType) {          if (shapeType == null) {              return null;          }          if (shapeType.equalsIgnoreCase("CIRCLE")) {              return new Circle();          } else if (shapeType.equalsIgnoreCase("RECTANGLE")) {              return new Rectangle();          } else if (shapeType.equalsIgnoreCase("SQUARE")) {              return new Square();          }          return null;      }  }  public class FactoryPatternDemo {      public static void main(String[] args) {          ShapeFactory shapeFactory = new ShapeFactory();          // get an object of Circle and call its draw method.          Shape shape1 = shapeFactory.getShape("CIRCLE");          // call draw method of Circle          shape1.draw();          // get an object of Rectangle and call its draw method.          Shape shape2 = shapeFactory.getShape("RECTANGLE");          // call draw method of Rectangle          shape2.draw();          // get an object of Square and call its draw method.          Shape shape3 = shapeFactory.getShape("SQUARE");          // call draw method of circle          shape3.draw();      }  } |