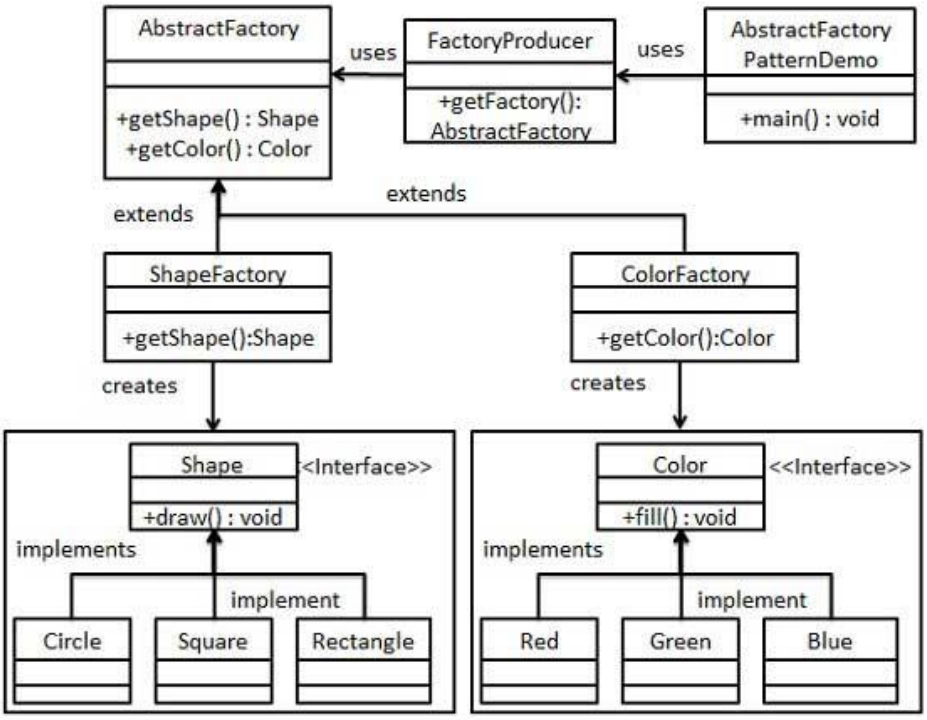
Name: Aimable M and Romerico D_

Abstract Factory Design Pattern

Problem:
The objective of this exercise is to implement the Abstract Factory design pattern.



Steps:

1. Create an interface called **Shape**. It defines a public method **draw**(). Create three subclasses (**Circle**, **Square**, and **Rectangle**) as indicated in the class diagram above. Each subclass implements draw() method by printing out a string as "Inside xxx.draw() method". For example, in **Circle**, it should print out "inside Circle.draw() method".

exercise_11 > Shape.java > ...

```
1  package exercise_11;
2
3  public interface Shape{
4      public void draw();
5  }
6
```

Name: Aimable M and Romerico D_

exercise_11 > Circle.java > ...

```java
package exercise_11;

public class Circle implements Shape {
    public void draw() {
        System.out.println(x:"Circle: draw()");
    }
}

```

exercise_11 > Square.java > Square

```java
package exercise_11;

public class Square implements Shape {

    public void draw() {
        System.out.println(x:"Square: draw()");
    }
}

```

exercise_11 > Rectangle.java > {} exercise_11

```java
package exercise_11;

public class Rectangle implements Shape {
    public void draw() {
        System.out.println(x:"Rectangle: draw()");
    }
}

```

2. Create an interface called **Color**. It defines a public method **fill**(). Create three subclasses (**Red**, **Green**, and **Blue**) as indicated in the class diagram above. Each subclass implements fill() method by printing out a string as “Inside xxx.fill() method”. For example, in **Red**, it should print out “inside Red.fill() method”.

exercise_11 > Color.java > {} exercise_11

```java
package exercise_11;

public interface Color {
    public void fill();
}

```

Name: Aimable M and Romerico D_

```
exercise_11 > Red.java > Red
1  package exercise_11;
2
3
4  public class Red implements Color {
5      public void fill() {
6          System.out.println(x:"Red: fill()");
7      }
8  }
9
```

```
exercise_11 > Green.java > Green
1  package exercise_11;
2
3  public class Green implements Color {
4
5      public void fill() {
6          System.out.println(x:"Green: fill()");
7      }
8  }
```

```
exercise_11 > Blue.java > Blue > fill()
1  package exercise_11;
2
3  public class Blue implements Color {
4      public void fill() {
5          System.out.println(x:"Blue: fill()");
6      }
7  }
8
```

Create an Abstract class called **AbstractFactory** to get factories for Color and Shape Objects. Define two abstract methods:

abstract Color getColor(String color);
abstract Shape getShape(String shape);

Name: Aimable M and Romerico D_

```
exercise_11 > AbstractFactory.java > AbstractFactory
1   package exercise_11;
2
3   public abstract class AbstractFactory {
4       abstract Color getColor(String color);
5       abstract Shape getShape(String shape);
6
7   }
8
9
```

3. Create Factory classes **ShapeFactory** and **ColorFactory** extending **AbstractFactory** to generate object of concrete class based on given information.

4. Implement **getShape** function in **ShapeFactory**, so that it checks the argument shapeType and create corresponding Shape objects. For example, if shapeType is “CIRCLE”, it should return new Circle().

```
public class ShapeFactory extends AbstractFactory {

    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;
    }

    @Override
    public Color getColor(String color) {
        return null;
    }

}
```

Shape getShape(String shapeType){ … … }

5. Implement **getColor** function in **ColorFactory**, so that it checks the argument colorType and create corresponding Color objects. For example, if colorType is “RED”, it should return new Red().

Name: Aimable M and Romerico D_

Color getColor(String colorType){ ... ... }

exercise_11 > ColorFactory.java > ...

```java
package exercise_11;

public class ColorFactory extends AbstractFactory {

    public Color getColor(String color) {
        if (color == null) {
            return null;
        }
        if (color.equalsIgnoreCase(anotherString:"RED")) {
            return new Red();
        } else if (color.equalsIgnoreCase(anotherString:"GREEN")) {
            return new Green();
        } else if (color.equalsIgnoreCase(anotherString:"BLUE")) {
            return new Blue();
        }
        return null;
    }

    @Override
    public Shape getShape(String shape) {
        return null;
    }

}

```

6. Create a Factory generator/producer class called **FactoryProducer** to get factories by passing an information such as Shape or Color.

```
public class FactoryProducer {
  public static AbstractFactory getFactory(String choice){
    if(choice.equalsIgnoreCase("SHAPE")){
      return new ShapeFactory();
    }else if(choice.equalsIgnoreCase("COLOR")){
      return new ColorFactory();
    }
    return null;
  }
}
```

Name: Aimable M and Romerico D_

exercise_11 > J FactoryProducer.java > ...

```java
package exercise_11;

public class FactoryProducer {

    public static AbstractFactory getFactory(String choice) {

        if (choice.equalsIgnoreCase(anotherString:"SHAPE")) {
            return new ShapeFactory();
        } else if (choice.equalsIgnoreCase(anotherString:"COLOR")) {
            return new ColorFactory();
        }
        return null;
    }

}
```

7. Use the client below to check the results.

```
public class AbstractFactoryPatternDemo {
  public static void main(String[] args) {
    //get shape factory
    AbstractFactory shapeFactory = FactoryProducer.getFactory("SHAPE");
    //get an object of Shape Circle
    Shape shape1 = shapeFactory.getShape("CIRCLE");
    //call draw method of Shape Circle
    shape1.draw();
    //get an object of Shape Rectangle
    Shape shape2 = shapeFactory.getShape("RECTANGLE");
    //call draw method of Shape Rectangle
    shape2.draw();
    //get an object of Shape Square
    Shape shape3 = shapeFactory.getShape("SQUARE");
    //call draw method of Shape Square
    shape3.draw();
    //get color factory
    AbstractFactory colorFactory = FactoryProducer.getFactory("COLOR");
    //get an object of Color Red
    Color color1 = colorFactory.getColor("RED");
    //call fill method of Red
    color1.fill();
    //get an object of Color Green
    Color color2 = colorFactory.getColor("Green");
    //call fill method of Green
    color2.fill();
    //get an object of Color Blue
    Color color3 = colorFactory.getColor("BLUE");
    //call fill method of Color Blue
    color3.fill();
```

Name: Aimable M and Romerico D_

```
  }
}
```

```
Circle: draw()
Rectangle: draw()
Square: draw()
Red: fill()
Green: fill()
Blue: fill()
```

Upload your code to the Blackboard when you are done.