TASK 01

```
interface called_command{
    void execute();
    void undo();
}
interface called_light{
    void on();
    void off();
    void dim();
}
class  kitchen_room_light  implements called_light{
   public void on(){
      System.out.println("kitchen light on");
   }
   public void off(){
      System.out.println("kitchen light off");
   }
   public void dim(){
      System.out.println("kitchen light dim");
   }
}
class living_room_light implements called_light{

   public void on(){
      System.out.println("living light on");
   }
   public void off(){
      System.out.println("living light off");
   }
   public void dim(){
      System.out.println("living light dim");
   }
}
class light_on_command implements called_command{

   public void execute(){
      System.out.println("light on command execute");
   }
   public void undo(){
      System.out.println("light on command undo");
   }

}
class light_off_command implements called_command{

   public void execute(){
      System.out.println("light off command execute");
   }
   public void undo(){
      System.out.println("light off command undo");
   }
}
class light_dim_command implements called_command{

  public void execute(){
    System.out.println("light dim command execute");
  }
  public void undo(){
    System.out.println("light dim command undo");
  }

}
public class demo{
  public static void main(String[] args){
    called_light ob;
    ob = new kitchen_room_light();
    ob.off();
    ob.on();
    ob.dim();

    ob = new living_room_light();
    ob.off();
    ob.on();
    ob.dim();

    called_command ob1;
    ob1 = new light_on_command();
    ob1.execute();
    ob1.undo();

    ob1 = new light_off_command();
    ob1.execute();
    ob1.undo();

    ob1 = new light_dim_command();
    ob1.execute();
    ob1.undo();

  }
}
```

Task 02

```
import java.util.Scanner;
abstract class Beverages{
   public void boil_water(){

   }
   public void brew(){

   }
   public void pourincup(){

   }
   public void addcondiments(){

   }
   public abstract void addExtras();
   public void finaltemplatemethod(){
       System.out.println("boiling water");
      // System.out.println("steeping the tea");
      // System.out.println("pouring into cup");
       //System.out.println("adding lemon");
   }
   public void setwantsextras(boolean wantsExtra){

   }

}
class tea extends Beverages{
   public void brew(){
     System.out.println("steeping the tea");
   }
   public void addcondiments(){
      System.out.println("pouring into cup");
   }
   public void addExtras(){
      System.out.println("adding lamon");
   }
}
class coffee extends Beverages{
   public void addExtras(){
      System.out.println("adding vanilla syrup");
   }
    public void brew(){
      System.out.println("dripping coffee through filter");
      System.out.println("pouring into cup");
   }
   public void addcondiments(){
      System.out.println("adding sugar and milk");
   }
}
```

```
public class BeverageTest{
  public static void main(String []args){
    Scanner ob = new Scanner(System.in);
    System.out.println("do you want extras with your tea(yes/no):");
    boolean extraTea = ob.nextLine().trim().equalsIgnoreCase("yes");

    System.out.println("do you want extras with your coffee(yes/no);");
    boolean extraCoffee = ob.nextLine().trim().equalsIgnoreCase("yes");

    Beverages Tea = new tea();
    Tea.setwantsextras(extraTea);


    Beverages Coffee = new coffee();
    Coffee.setwantsextras(extraCoffee);

    System.out.println("making tea...");
    Tea.finaltemplatemethod();

    Tea.brew();
    Tea.addcondiments();
    Tea.addExtras();

    System.out.println("\nmaking coffee...");
    Coffee.finaltemplatemethod();

    Coffee.brew();
    Coffee.addcondiments();
    Coffee.addExtras();

    ob.close();
  }
}
```

Task 03

```
public interface Shape {
  void draw();
}

// Shape Interface
public interface Shape {
  void draw();
}

// Square Class
public class Square implements Shape {
  @Override
  public void draw() {
    System.out.println("Drawing a Square");
  }
}

// Circle Class
public class Circle implements Shape {
  @Override
  public void draw() {
    System.out.println("Drawing a Circle");
  }
}

// Triangle Class
public class Triangle implements Shape {
  @Override
  public void draw() {
    System.out.println("Drawing a Triangle");
  }
}

// Rectangle Class
public class Rectangle implements Shape {
  @Override
  public void draw() {
    System.out.println("Drawing a Rectangle");
  }
}

// Shape Interface
public interface Shape {
  void draw();
}

// Square Class
public class Square implements Shape {
  @Override
  public void draw() {
    int size = 5; // You can change the size as needed
    for (int i = 0; i < size; i++) {
      for (int j = 0; j < size; j++) {
        System.out.print("* ");
      }
```

```
            System.out.println();
        }
    }
}

// Circle Class
public class Circle implements Shape {
    @Override
    public void draw() {
        int radius = 3; // You can change the radius as needed
        double dist;

        for (int i = 0; i <= 2 * radius; i++) {
            for (int j = 0; j <= 2 * radius; j++) {
                dist = Math.sqrt((i - radius) * (i - radius) + (j - radius) * (j - radius));

                if (dist > radius - 0.5 && dist < radius + 0.5) {
                    System.out.print("* ");
                } else {
                    System.out.print("  ");
                }
            }
            System.out.println();
        }
    }
}

// Triangle Class
public class Triangle implements Shape {
    @Override
    public void draw() {
        int height = 5; // You can change the height as needed
        for (int i = 0; i < height; i++) {
            for (int j = 0; j <= i; j++) {
                System.out.print("* ");
            }
            System.out.println();
        }
    }
}

// Rectangle Class
public class Rectangle implements Shape {
    @Override
    public void draw() {
        int width = 8; // You can change the width as needed
        int height = 4; // You can change the height as needed

        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                System.out.print("* ");
            }
            System.out.println();
        }
    }
}
```

```java
// ShapeFactory Class
public class ShapeFactory {

    // Method to get the desired Shape
    public Shape getShape(String shapeType) {
        if (shapeType == null) {
            return null;
        }
        if (shapeType.equalsIgnoreCase("SQUARE")) {
            return new Square();
        } else if (shapeType.equalsIgnoreCase("CIRCLE")) {
            return new Circle();
        } else if (shapeType.equalsIgnoreCase("TRIANGLE")) {
            return new Triangle();
        } else if (shapeType.equalsIgnoreCase("RECTANGLE")) {
            return new Rectangle();
        }
        return null;
    }
}
```

```java
// ShapeFactory Class
public class ShapeFactory {

    // Method to get the desired Shape
    public Shape getShape(String shapeType) {
        if (shapeType == null) {
            return null;
        }
        if (shapeType.equalsIgnoreCase("SQUARE")) {
            return new Square();
        } else if (shapeType.equalsIgnoreCase("CIRCLE")) {
            return new Circle();
        } else if (shapeType.equalsIgnoreCase("TRIANGLE")) {
            return new Triangle();
        } else if (shapeType.equalsIgnoreCase("RECTANGLE")) {
            return new Rectangle();
        }
        return null;
    }
}
```

```java
public class ShapeTest {
    public static void main(String[] args) {
        // Create a ShapeFactory object
        ShapeFactory shapeFactory = new ShapeFactory();

        // Test creating and drawing a Square
        Shape square = shapeFactory.getShape("SQUARE");
        System.out.println("Drawing a Square:");
        square.draw();
        System.out.println();

        // Test creating and drawing a Circle
        Shape circle = shapeFactory.getShape("CIRCLE");
        System.out.println("Drawing a Circle:");
        circle.draw();
        System.out.println();

        // Test creating and drawing a Triangle
        Shape triangle = shapeFactory.getShape("TRIANGLE");
        System.out.println("Drawing a Triangle:");
        triangle.draw();
        System.out.println();

        // Test creating and drawing a Rectangle
        Shape rectangle = shapeFactory.getShape("RECTANGLE");
        System.out.println("Drawing a Rectangle:");
        rectangle.draw();
        System.out.println();
    }
}
```