P

Polymorphism

Project ID: 5592

Forked from an inaccessible project.

Slight cleanup

Mads Würtz Pedersen authored 1 month ago

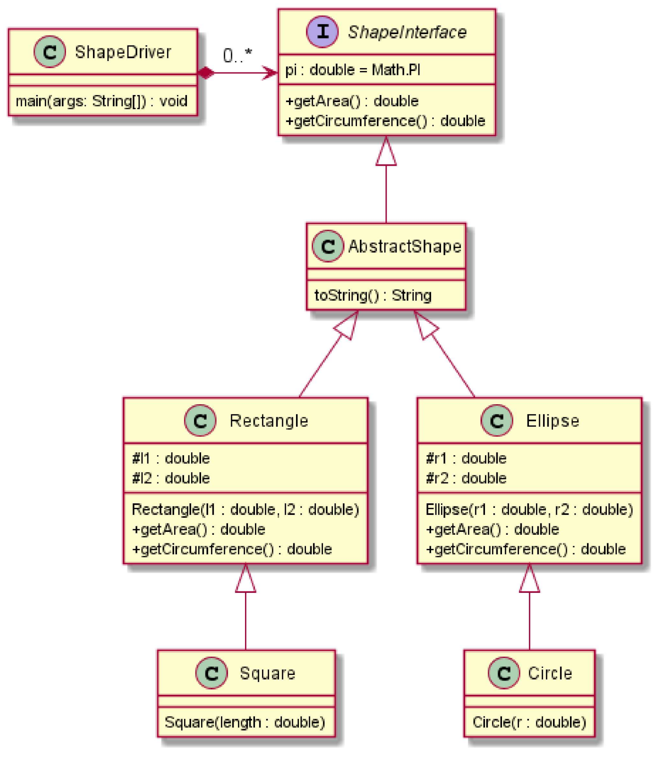
Name	Last commit	Last update
assets	Added exercise	1 month ago
solution/DO_NOT_LOOK_IN_HER...	Slight cleanup	1 month ago
src/main	Added exercise	1 month ago
README.md	Initial commit	1 month ago
pom.xml	Added exercise	1 month ago

README.md

Polymorphism

Tasks - Inheritance, Interface og Polymorphism

Your assignment is to implement the following class diagram:



Supplied code:

- ShapeInterface.java
- AbstractShape.java
- ShapeDriver.java

Task a - Ellipse and Rectangle

1. Create a subclass `Ellipse.java` as follows: `public class Ellipse extends AbstractShape`

2. Declare 2 instance variables of type `double`
3. Create a constructor to initialise these variables
4. Implement methods `getArea()` and `getCircumference()` using the formula below.

Similarly do it for Rectangle as well.

5. Create a subclass `Rectangle.java` as follows: `public class Rectangle extends AbstractShape`
6. Declare 2 instance variables of type `double`
7. Create a constructor to initialise these variables
8. Implement methods `getArea()` and `getCircumference()` using the formula below.

The following formulas can be used to calculate `getArea()` and `getCircumference()` :

	Ellipse	**Rectangle**
Area	$\pi * r1 * r2$	$l1 * l2$
Circumference	$2 * \pi * \sqrt{(½ * (r1^2 + r2^2))}$	$2 * (l1 + l2)$

Task b - Circle and Square

Create a subclass Circle.java as follows: `public class Circle extends Ellipse` Create a subclass Square.java as follows: `public class Square extends Rectangle`

Remember that circles and squares are just ellipses and rectangles with r1=r2 and l1=l2 respectively

When `ShapeDriver` is executed the output must look like the following:

```
Shapes: [
Jeg er en Circle med Areal 36,317 og Omkreds 21,363,
Jeg er en Rectangle med Areal 14,960 og Omkreds 15,600,
Jeg er en Ellipse med Areal 28,840 og Omkreds 19,289,
Jeg er en Square med Areal 11,560 og Omkreds 13,600]
```

Task c (ekstra) - GUI and Singleton Facade

In this task, you'll work with the following class:

- ShapeFacade.java

This is the only class we will access from the GUI. The Singleton pattern is a design pattern, ensuring only 1 instance of the class exists. This instance can be accessed through the `getInstance()` -method.

To distinguish between the 4 concrete instances of Shapes, the class contains the following enum:

```
public enum SHAPES {
    CIRCLE, ELLIPSE, RECTANGLE, SQUARE
};
```

`SHAPES` can be accessed statically through other classes (example: `ShapeFacade.SHAPES.CIRCLE`).

Task d - getShapeInfo() method

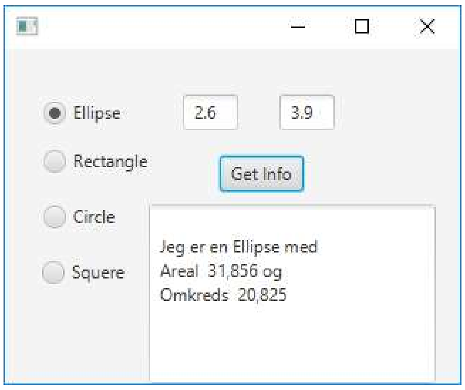
Implement the following method depending on the shape it recieves, it should take either 1 or 2 parameters of the double type, as follows

```
public String getShapeInfo(SHAPES shape, double... parametre) {
    throw new UnsupportedOperationException("Not supported yet.");
}
```

- Create a switch depending on the type of shape: `Circle` , `Square` , `Ellipse` , `Rectangle` .
- These shapes should then be created with the parameters the method receives.
- If the case is either `Circle` or `Square` , it should take the 1st double param passed to the method to construct the shape.
- If the case is `Ellipse` or `Rectangle` , it should take the 1st and 2nd double param passed to the method to construct the shape.

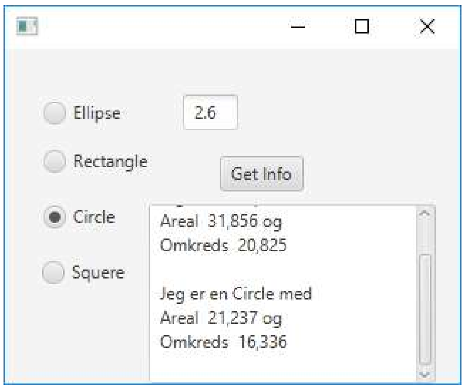
Task e - Implement the GUI

In your resources folder locate the FXML file. The final GUI is supposed to look like the following screenshots:



2.6
3.9

Jeg er en Ellipse med
Areal 31,856 og
Omkreds 20,825



2.6

Areal 31,856 og
Omkreds 20,825

Jeg er en Circle med
Areal 21,237 og
Omkreds 16,336

It consists of:

- **4** `RadioButtons` for choosing the Shape.
- **2** `TextFields` TextFields for params. Only one is visible if Circle or Square has been selected.
- **1** `TextArea` for results.
- **1** `Button` "Get Info" for calling the facade.

Implement an `ActionHandler` for the 4 `RadioButtons` , to ensure one or both `TextFields` are visible.

Hint: in the `initialize()` -method of the Controller the radiobuttons have been given userData with the enum-values from the Facade. i.e. `ellipseRadio.setUserData(ShapeFacade.SHAPES.ELLIPSE);`
These can be accessed with: `ShapeFacade.SHAPES shape = (ShapeFacade.SHAPES)ShapeToggle.getSelectedToggle().getUserData();`
Remember to uncomment the code and to name the radiobuttons accordingly.

If the Shape is Circle or Square, then 1 TextField should be visible and if the shape is Ellipse or Rectangle, 2 TextFields should be visible.

Implement a `ActionHandler` on the Get Info-button, so that the facade is called by:

- `ShapeFacade.getInstance().getShapeInfo(shape, new double[]{p1});` (Circle/Square) or
- `ShapeFacade.getInstance().getShapeInfo(shape, new double[]{p1, p2});` (Ellipse/Rectangle)

The result is then printed out in the TextArea.

```java
package vop;


import javafx.application.Platform;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;

import java.io.File;
import java.net.URISyntaxException;
import java.net.URL;
import java.util.ResourceBundle;

public class PrimaryController implements
Initializable, CallBackInterface {

    @FXML
    private TextArea textArea;
    @FXML
    private Button startButton;
    @FXML
    private Button stopButton;
    @FXML
    private ImageView die1view;
    @FXML
    private ImageView die2view;

    private FacadeWithCallback facade;


    @Override
    public void initialize(URL url, ResourceBundle rb
) {
        startButton.setDisable(false);
        stopButton.setDisable(true);
    }
    @FXML
    private void buttonAction(ActionEvent event) {
        //Ved if-statement løses ved Opgave 1
        if (event.getSource() == startButton) {
            try {
                facade = new FacadeWithCallback(this
);
                stopButton.setDisable(false);
                startButton.setDisable(true);
            } catch (URISyntaxException e) {
                throw new RuntimeException(e);
            }
            //Ved else-statement løses Opgave 2.
        } else {
            facade.interrupt();
            stopButton.setDisable(true);
            startButton.setDisable(false);
            // Stop the facade
        }
    }

    //I denne metode løser vi Opgave 3.
    @Override
    public void updateMessage(String message) {
        Platform.runLater(new Runnable() {
            @Override
            public void run() {
                textArea.appendText(message);
                if(!facade.isAlive()){
                    stopButton.fire();
                }
            }
        });
    }

    //Vi har fjernet kommentaret her for at løse
Opgave 4.
    @Override
    public void updateImages(File i1, File i2) {
        // Changes the pictures on the imageViews
        Platform.runLater(new Runnable() {
            @Override
            public void run() {
                die1view.setImage(new Image(i1.toURI().toString()));
                die2view.setImage(new Image(i2.toURI().toString()));
            }
        });



    }


}
```

```java
package circularbuffer;

import java.util.Arrays;

public class CircularBuffer {
    private Integer[] buffer;
    private int size;
    private int putIndex = 0;
    private int getIndex = 0;

    public CircularBuffer(int size) {
        buffer = new Integer[size];
        this.size = size;
    }

    synchronized int get() {
        //Vi løser den anden del af Opgaven her.
        int ligemeget = 0;
        if (buffer[getIndex] == null) {
            try {
                wait(1800);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }

        } else {
            System.out.println(Thread.currentThread().getName()+"\tGot: " + getIndex + ": " + buffer[getIndex]);
            ligemeget = buffer[getIndex];
            buffer[getIndex] = null;
            getIndex = (getIndex+1) % (buffer.length-1);
            notifyAll();
        }
        return ligemeget;
    }

    synchronized void put(int n) {
        //Vi løser den første del af Opgaven her.
        if (buffer[putIndex] != null) {
            try {
                wait(1800);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        } else {
            buffer[putIndex] = n;
            System.out.println("Produceren har puttet værdien" + n + " i position" + putIndex);
            putIndex = (putIndex+1) % (buffer.length-1);
            notifyAll();
        }
    }

    public String toString() {
        return "Buff: " + Arrays.toString(buffer);
    }
}


```