Übung 04: Interfaces, Inner Classes

Abgabetermin: 14. 4. 2016, 8:15

Name: _____ **Matrikelnummer**: _____

Informatik: ❑ G1 (Grimmer) ❑ G2 (Prähofer) ❑ G3 (Prähofer)

WIN: ❑ G1 (Khalil) ❑ G2 (Khalil) ❑ G3 (Hummel)

Aufgabe	Punkte	abzugeben schriftlich	abzugeben elektronisch	korr.	Punkte
Übung 3	24	Java-Programm, Ausgabe der Testläufe	Java-Programm	❑	

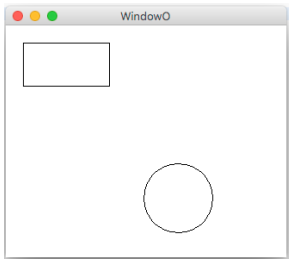
For this exercise, you should develop a framework for creating animated figures. There are five stages, in which features are added to the program. We provide you with a small test run for each of the stages that focuses on the newly developed capabilities (`Main1, …, Main5`). There are three predefined types (`Controller`, `Window`, `Animation, HasArea`) you may use - they are explained in the Appendix.

Please note: The `Controller` class has compilation errors in it when you first open the project. These errors disappear when you implement the `Figure` class in Stage 1.

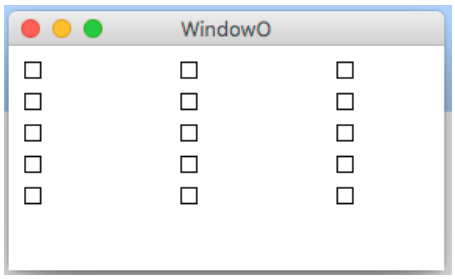
Stage 1: Figures (5 Points)

Create an abstract base class `Figure` that contains two fields for the x- and y-coordinates. Additionally, it should have an abstract method `void draw(int xOrigin, int yOrigin)` that draws the figure relative to the given coordinates. Write two subclasses of `Figure` (`Rectangle`, `Circle`).

Hint: Use the class `Controller` for registering the figures. The `Controller` uses the class `Window` to draw them. The position of a figure is determined by its `x`/`y`-position plus the `xOrigin`/`yOrigin`-position given as a parameter to the draw method.



Stage 2: Compound Figure (5 Points)



Create a class `CompoundFigure` as a subclass of `Figure`. Implement a method `void add(Figure f)` that adds a child figure. Internally, the `CompoundFigure` should use a `FigureList` to store the child figures.

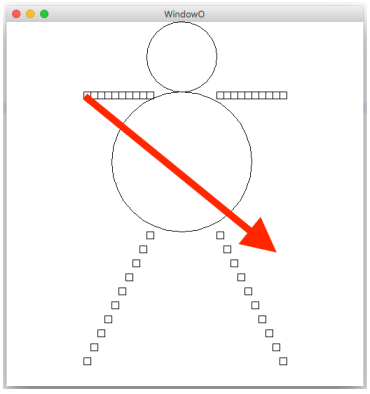
Implement `FigureList` as a *private static inner class*. The `FigureList` itself has nodes for storing the list of child figures. `FigureList` should implement the `Iterable<Figure>` interface, which allows iterating over all figures:

```
for (Figure f : figures) { ... }
```

Make sure all child figures are drawn relative to the position of the compound figure.

Hint: The child figures are no longer registered with the `Controller`, but only with their parent compound figure.

Stage 3: X- and Y-Animation (5 Points)



The interface `Animation` is provided to you, it contains the method `void animate(int frame)` and is called from the `Controller` at each frame. Create a method `Animation createXAnimation()` in the class `Figure`. Use an ***anonymous*** *inner class* for implementing the `Animation` interface. The animation should increase the x-coordinate of the figure at every frame. Additionally, create a method `Animation createYAnimation()`. For implementing this method, you should create a *named* ***static*** *inner class* called `YAnimation`. In a comment, explain the advantages and disadvantages of using an anonymous inner class compared to a named inner class.

Hint: Use the class `Controller` for registering the animations.

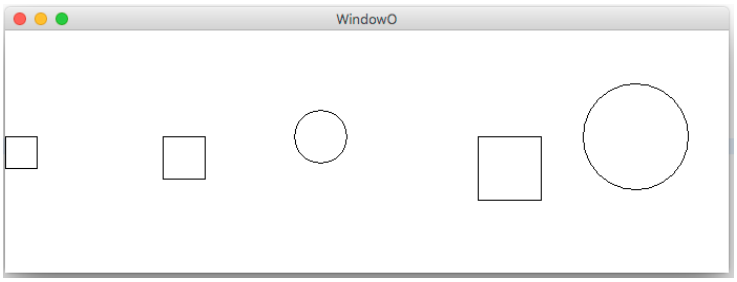
Stage 4: Circle-Animation (5 Points)

Create a named inner class called `CircleAnimation` that implements the `Animation` interface and can be used to let the figure move in a circle. Also, add a method `Animation createCircleAnimation(int radius, int framesPerRotation)` to the Figure class. In a comment, explain the advantages and disadvantages of modeling animations using an interface (`Animation`) compared to an abstract base class.

Stage 5: Sorting (4 Points)

The class `Main5` creates 5 figures, which the `Controller` can draw from left to right in a sorted manner using the function `void displaySorted(Comparator<Figure> comparator)`. Complete the implementation of `Main5` by adding an implementation of the `Comparator<Figure>` needed for `displaySorted`. The figures should be sorted by their area.

Hint: Add a method `int getArea()` to all your figure classes, which returns the area of a figure. Use this method to implement the `Comparator<Figure>`.



Appendix

Controller.java

The `Controller` class is used to register the figures (`addFigure`) and animations (`addAnimation`). Afterwards, the `display` method can be called to bring up the window and start the animation. This file will only compile after you have successfully completed the first stage of the exercise.

- `Controller.addFigure(Figure f)`: Registers the given figure `f` to be displayed on the screen. The controller will automatically call the `draw` method of all registered figures. Figures, that are already registered as a child of a compound figure must not be registered with the controller.

- `Controller.addAnimation(Animation a)`: Registers the given animation `a` to be performed at every frame. The controller will automatically call the `animate` method of all registered animations. The parameter `frame` of the animate method indicates the number of the current displayed frame. This parameter can be used e.g. to restart an animation every 25 frames.

- `Controller.display()`: Displays all registered figures and starts all registered animations. Needs to be called after all animations and figures have been registered.

- `Controller.displaySorted()`: Displays all registered figures in a sorted manner from left to right. Needs to be called after all animations and figures have been registered. To get a reasonable output all registered figures should have (0, 0) as their center. Figures will be sorted from left to right with 150px space between figures.

Example code:

```
Circle c = new Circle(50, 50, 10);
Controller.addAnimation(c.createXAnimation()); // Works at stage 3+ only
Controller.addFigure(c);
Controller.display();
```

Window.java

The `Window` class is used to draw onto the screen. It should only be used within the `draw` methods. Here are some example methods that might be useful (feel free to use any of the `Window` methods in your program):

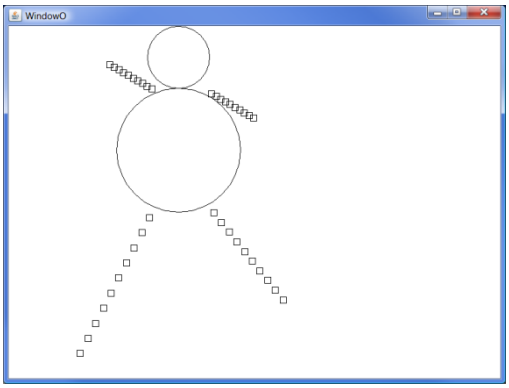
- `Window.drawCircle(int x, int y, int r)`: Draws a black circle with the center coordinates `x/y` and radius `r`.

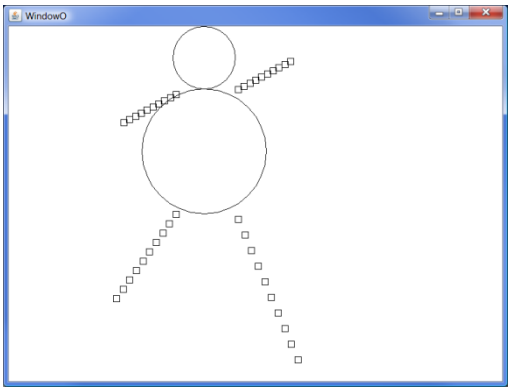
- `Window.drawPoint(int x, int y)`: Draws a black pixel at the given `x/y` location.

- `Window.drawRectangle(int x, int y, int w, int h)`: Draws a black rectangle with the top left coordinate `x/y` and the given width `w` and height `h`.

Animation.java

This file contains the interface `Animation` that should be used at stages 3 and 4 of the exercise. The interface has a single method `animate` with a parameter `frame`. This method is automatically called by the controller at every frame.


WindowO


WindowO