Fachhochschule Nordwestschweiz

Modul Design Patterns
Prof. Dr. Dominik Gruntz
Prof. Dr. Wolfgang Weck
Prof. Dr. Christoph Denzler

First Steps with JDraw: Hints on Assignment 2

This is an introduction on how to get started with the first assignment on JDraw:

1. Implement `StdDrawModel.getFigures`:
   - This function is already being called from `StdDrawView`, but does not do anything yet
   - It requires an internal data structure to store all figures
   - This data structure needs to be filled dynamically. Therefore, you must also implement `StdDrawModel.addFigure`
   - In order to choose an appropriate data structure, consider that it must return overlapping figures correctly ordered
   - The data structures of the collection framework provide a method `stream()` returning a Stream of all of the data structure's current elements.
2. Implement `StdDrawModel.addModelChangeListener`:
   - This function is already being called from `StdDrawView`, but does not do anything yet
   - It requires an internal data structure to store all registered Observers
   - These registered Observers need to be notified upon changes in the model (check the `DrawModelListener` interface on how to do that)

   Checkpoint: By now you should be able to add a rectangle. However, it might randomly get stuck at some size or may not be visible at all. Upon triggering the redrawing of the window (e.g. by resizing it) the figure will pop up.

   The reason for this behavior is that the rectangle is initially inserted with a size of (0, 0) and only resized afterwards. However, resizing does not trigger a window redrawing yet. Triggering a redraw manually, however, causes all figures to be drawn at their current place and with their current size.

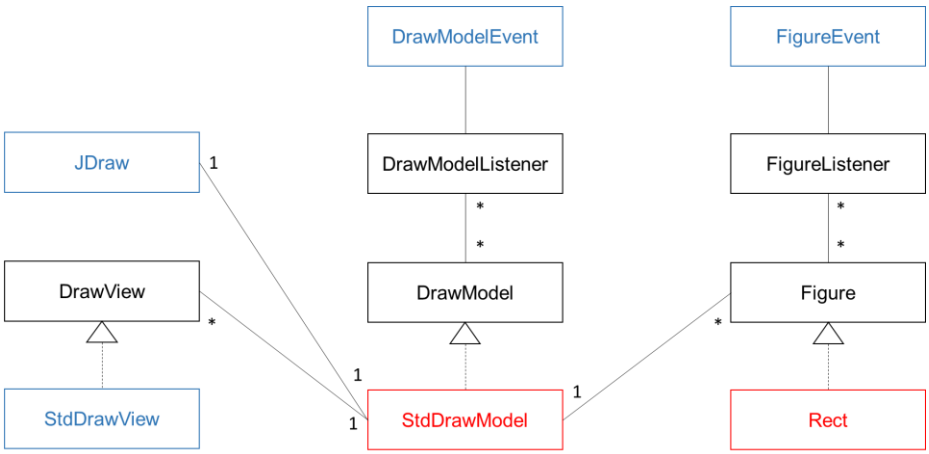
   Typically, the figures appear delayed, i.e. when a new figure is inserted, the figure inserted before appears. However, if the figures only appear after the redrawing of the window is explicitly triggered, `addFigure` is likely not notifying the view correctly.

   If the figures do not appear at all, method `getFigures` does not return the newly inserted figures.
3. Report resizing events of Rect to all views
   - `Rect` needs to become an Observable, in which Observers can be registered
   - The appropriate function `addFigureListener` is defined, but neither implemented nor called anywhere in the code
   - Upon resizing or moving a figure (functions `Figure.setBounds` and `Figure.move`) the registered Observers have to be notified
   - The corresponding Observers need to be coded and registered. There are multiple possibilities on how to do this.

   Checkpoint: The insertion and movement of rectangles should now work correctly. Furthermore, the same model should be viewable from multiple windows (test this via *Window → New Window*).

   However, rectangles cannot be removed.
4. Completion
   - Figures should not only be able to be added but also be removed. The functions `StdDrawModel.removeFigure` and `StdDrawModel.removeAllFigures` have to be implemented in order to achieve this. *Hint:* What do you need to take care of regarding the Observer-mechanisms which are part of the figures?
   - Views correspond to any open windows displaying the model. These can be created using the menu: *Window → New Window*. Upon closing an open window the function `StdDrawModel.removeModelChangeListener` is being called, which still needs to be implemented.
   - Reordering figures: `StdDrawView` draws all figures in the same order as they appear in the stream returned by the function `getFigures()`. Using the menu *Edit → Order…* we want to be able to move a selected figure to the beginning or end of the drawing list. This is achieved by calling the function `StdDrawModel.setFigureIndex` which is also not yet implemented.
   - Use the existing JUnit tests to test your program. You should achieve the passing of all tests. However, you might find that some additional edge cases need to be handled.

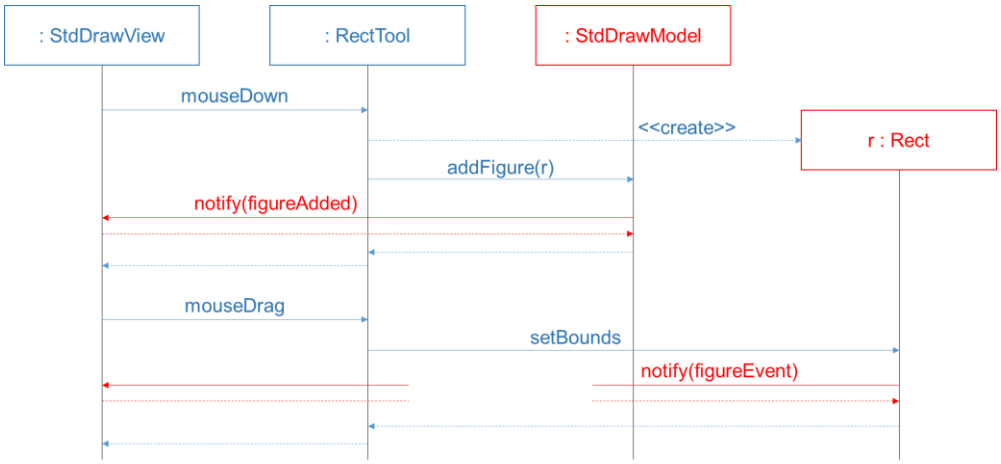
An overview of the most important classes/interfaces in JDraw:



schwarz: Interfaces
blau: bereits fertige Klassen
rot: Klassen, in denen für Übung 2 etwas programmiert werden muss (s, TODO-Tags)

black: interfaces
blue: readily implemented classes
red: classes to be modified in this assignment: TODO

Sequence diagram for the insertion of a new rectangle:



blau: Diese *Aufrufe* sind bereits implementiert
rot: Aufrufe die noch implementiert werden müssen als Teil der Übung 2

blue: readily implemented functions
red: functions to be implemented in this assignment