Assignment 8: Save & Load of graphics

In this assignment you will implement the commands File->Save and File->Open such that your graphics can be saved on and loaded from disk.

The figures should be written to an `ObjectOutputStream` and read from an `ObjectInputStream` (you have met these classes in the context of the topic *Cloning using Serialilzation)*. All objects which shall be serialized must implement the marker interface `java.io.Serializable`. Your figures already fulfil this condition as interface `Figure` is derived from interface `Serializable`.

An instance of class `ObjectInputStream` and `ObjectOutputStream` can be created as follows:

```
ObjectInputStream ois = new ObjectInputStream(new FileInputStream(...));
ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(...));
```

Writing

First the user must be asked under which name the data should be stored. A suitable dialog for this is implemented in class `JFileChooser`. Method `getFile` of that class provides access to the selected file. Use the constructor `FileOutputStream(File)` to create a file output stream writing to that file and decorate it with an `ObjectOutputStream`.

In order to store a graphic, all figures contained in its model must be written one by one to the `ObjectOutputStream`. To avoid storing registered listeners (and consequently also the model referenced by these listeners), a copy of the figure can be created using method `clone`.

When all the figures are written, the output stream must be closed using method `close`.

Reading

In order to load a stored graphic a `JFileChooser` can serve to query the user for the respective file. Similarly to the description above for writing the figures a `FileInputStream` and an `ObjectInputStream` can connect to this file to read the figures.

A loop can read the figures and add each figure to the model. Before reading a graphic the model may be cleared. Alternatively, figures may be *added* to the current graphics (comparable to an “Import graphics” command).

If figures are read using the `ObjectInputStream` then it is unknown when all figures stored in the file have been read. If the stream just ends after reading the last figure, on an attempt to read further an `EOFException` is thrown. Alternatively, saving a `null` reference as last figure may mark the end explicitly, or the number of the figures contained in a file can be stored prior to the data of the first figure.

Probably you learned in the module OOP2 other methods how to serialize an object structure (e.g. using XML or using JSON). You can also apply those techniques instead of Java serialization in this assignment.

Deadline: November 20, 2018