Assignment 11: Spring Configuration

In this assignment we extend our graphics editor so that the figure menu and the tool palette can be defined in the configuration file `src/main/resources/jdraw-config.xml`. Class `StdContext` should no longer explicitly reference and instantiate all draw tool implementations which are provided by the editor. This task should be performed by the *Spring* framework.

In the JDraw framework, the following factory interface is given:

```
package jdraw.framework;
/**
 * A DrawTool factory creates an instance of a draw tool and returns a name and icon which can be
 * used in the menu and in the tool palette of the graphics editor.
 */
public interface DrawToolFactory {

    // Name which can be used in a menu
    String getName();
    void setName(String name);

    // Name of path to draw tool icon
    String getIconName();
    void setIconName(String name);


    // Returns a draw tool operating on the given controller
    DrawTool createTool(DrawContext context);
}
```

In the configuration file `src/main/resources/jdraw-context.xml` beans can be defined for your factory implementations:

```
<bean id="rectangle" class="jdraw.figures.RectangleToolFactory">
    <property name="name"><value>Rectangle</value></property>
    <property name="iconName"><value>rectangle.png</value></property>
</bean>
```

These beans can be passed as constructor argument at the definition of the bean `drawContext`. The entry `null` is interpreted as menu separator.

```
<constructor-arg>
    <list>
        <ref bean="line"/>
        <ref bean="rectangle"/>
        <ref bean="oval"/>
        <null/>                 <!-- separator -->
        <ref bean="star"/>
    </list>
</constructor-arg>
```

You have to adapt method `doRegisterDrawTools` in your context class (by default `jdraw.std.StdContext`). This method will have to iterate over the list of tool factories which was passed to the constructor. Each factory will have to generate a tool to be registered through method `addTool`.

Method `createTool`, which is implemented for each tool factory, creates a new draw tool instance. This method could use the name and the icon to construct the tool as they were defined in the configuration file and passed to methods `setName` and `setIconName`.

If you understood the mechanism of the *Spring* configuration you could also use *Spring* dependency injection to define the decorator menu entries or generally all JDraw menu entries.

Once you have solved this assignment, your figure implementations can be easily integrated in the editor of other students. To do this you have to package all necessary classes, i.e. the class of the figure implementation, any abstract base classes you implemented, the handle and tool classes and the tool factory class into a JAR file. This archive can then be added to the class path and you only have to extend the configuration file. Class names obviously have to be unique. You should use a separate package name for your own figures (e.g. `jdraw.figures.mueller`).

A sample figure is provided in the archive `jdraw-swisscross-figure.jar`. You only have to add this JAR File on the class path and then adjust your Spring context.

You can add this JAR to the classpath using your IDE, but it's easier if you adjust the dependencies in the gradle configuration `build.gradle`. If you save the JAR file in the directory `lib/` in your project directory, you must extend the gradle dependencies by the following line:

```
dependencies {
    …
    compile files('lib/jdraw-swisscross-figure.jar')
}
```

The name of the class which implements the `DrawToolFactory` for this figure is `figures.ernst.SwissCrossToolFactory`. With the following declaration you can define this bean in your spring configuration file:

```
<bean id="ernst-swiss" class="jdraw.figures.ernst.SwissCrossToolFactory">
    <property name="name"><value>Swisscross</value></property>
    <property name="iconName"><value>swisscross.png</value></property>
</bean>
```

Deadline: January 15, 2019