Übung 11: Konfigurationsdatei

In dieser Übung soll unser Grafikeditor so erweitert werden, dass das Figuren-Menu und die Tool-Palette in der Konfigurationsdatei `src/main/resources/jdraw-context.xml` definiert werden können. In der Klasse `StdContext` sollen nicht mehr alle `DrawTool`-Implementierungen aufgeführt sein, welche im Editor zur Verfügung gestellt werden sollen, sondern diese Aufgabe soll von Spring übernommen werden.

Im Framework ist folgendes Fabrik-Interface vorgegeben:

```
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);
}
```

Im Konfigurationsfile `src/main/resources/jdraw-context.xml` können nun beans für Ihre Figuren definiert werden

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

und bei der Definition des Beans `drawContext` als Konstruktorargument übergeben werden. Der Eintrag `null` soll dabei als Menu-Separator interpretiert werden.

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

In Ihrer Kontextklasse (per Default `jdraw.std.StdContext`) muss nun die Methode `doRegisterDrawTools` angepasst werden. In dieser Methode muss über die im Konstruktor gesetzte Liste von ToolFactories iteriert werden und für jede Factory muss das damit erzeugte Tool mit der Methode `addTool` registriert werden. Die im Konfigurationsfile definierten Namen für das Tool und das Icon können in der Methode `createTool` dem erzeugten Tool übergeben werden damit es diese Namen verwendet.

Wenn Sie den Mechanismus der Spring-Konfiguration verstanden haben können Sie sich überlegen, ob Sie auch die Decorator-Menueinträge oder generell alle JDraw Menueinträge via Spring konfigurierbar machen wollen.

Sobald Sie diese Übung gelöst haben können Ihre Figurenimplementationen leicht in den Editoren anderer Studierender integriert werden. Dazu müssen die class-Dateien der Figurenimplementation, allfälliger eigener abstrakter Basisklassen, Handles, Tool-Klasse sowie der Tool-Factory in ein JAR-File gepackt werden. Dieses kann in den Klassenpfad aufgenommen werden und in der Konfigurationsdatei muss nur ein weiterer Eintrag vorgenommen werden. Damit die Klassennamen eindeutig sind sollten Sie für Ihre Figuren ein eigenes Paket verwenden (z.B. `jdraw.figures.mueller`).

Eine Beispielfigur finden Sie in der Datei `jdraw-swisscoross-figure.jar`. Sie müssen dieses JAR-File nur auf den Klassenpfad aufnehmen und ihren Spring-Kontext entsprechend anpassen.

Sie können das JAR-File mit Ihrer IDE dem Klassenpfad hinzufügen, aber einfacher ist es wenn Sie dazu das `build.gradle`-File anpassen. Wenn Sie das JAR-File im Verzeichnis `lib/` in Ihrem Projektverzeichnis speichern, müssen sie die Gradle-Dependencies um folgende Zeile erweitern:

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

Die Klasse `jdraw.figures.ernst.SwissCrossToolFactory`, die in diesem JAR-File enthalten ist, implementiert das Interface `DrawToolFactory`. Dieses Bean können Sie folglich mit folgender Deklaration im Konfigurationsfile definieren.

```
<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>
```

Abgabe: 15. Januar 2019