Arbeitsblatt Spring

Sie haben bei der Bearbeitung des letzten Arbeitsblatts eine *Abstract Factory* definiert, mit der die Komponenten der verschiedenen Produktfamilien (GUI Controls für AWT, Swing, SWT oder JavaFX) erzeugt werden können, und Sie haben auch gesehen, wie die Factory im Kontext gesetzt werden kann.

Sie sollen nun eine Spring-Variante entwickeln, bei der die Konfiguration in einem Spring-Konfigurations-File definiert werden kann. Um das GUI des Taschenrechners zu erzeugen haben wir das Interface `CalculatorFactory` definiert und den Rumpf der Klasse `CalculatorFactoryImpl` bereitgestellt, welche dieses Interface implementiert.

```
public class CalculatorFactoryImpl implements CalculatorFactory {

    public void setComponentFactory(Object fact) {
        // TODO this method is invoked by Spring in order to set the property
        //      "componentFactory". Change the type of the argument of this method
        //      from Object to something more concrete.
    }

    public Frame newCalculatorFrame() {
        // TODO this method is invoked by the main program to get the frame
        //      to be shown.
        return null;
    }

}
```

Diese Klasse erzeugt in der Methode `newCalculatorFrame` das Taschenrechner-GUI und benötigt dazu eine Instanz einer `ComponentFactory`. Das Spring-Konfigurationsfile `gui-context.xml` sieht wie folgt aus:

```
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">

<beans>
   <bean id="calculatorFactoryBean"
                              class="patterns.factory.gui.CalculatorFactoryImpl">
      <property name="componentFactory">
         <ref local="componentFactoryBean"/>
      </property>
   </bean>

   <bean id="componentFactoryBean" class="patterns.factory.gui.FactoryFX"/>
</beans>
```

wobei anstelle der Klasse `patterns.factory.gui.FactoryFX` ihre Factory-Klasse stehen muss.

Das Hauptprogramm kann dann mit

```
ctx = new ClassPathXmlApplicationContext("gui-context.xml");
```

auf die Konfiguration zugreifen und das in dieser Konfiguration definierte Bean `calculatorFactoryBean`- mit

```
ctx.getBean("calculatorFactoryBean")
```

anfordern. Auf diesem Objekt wird dann die Methode `newCalculatorFrame` aufgerufen, die ein Fenster für den Taschenrechner zurückliefert.

Das Hauptprogramm sieht dann wie folgt aus (diese Klasse ist ebenfalls im Projekt enthalten):

```
public class Gui04FactorySpring {
    private static ClassPathXmlApplicationContext ctx;

    static {
        ctx = new ClassPathXmlApplicationContext("gui-context.xml");
    }

    public static final void main(String[] args) {
        CalculatorFactory calcFactory =
                    (CalculatorFactory)ctx.getBean("calculatorFactoryBean");
        Frame f = calcFactory.newCalculatorFrame();
        f.setVisible(true);
    }
}
```

Die Klasse `CalculatorFactoryImpl` müssen Sie nun schreiben (bzw. das gegebene Fragment erweitern). Diese Klasse soll mit einer konkreten `ComponentFactory` parametrisiert werden können. Welche Factory dies ist, kann im Spring Konfigurationsfile definiert werden. Dem Konfigurationsfile auf der Vorderseite ist somit zu entnehmen, dass es in der Klasse `CalculatorFactoryImpl` ein Property geben muss (mindestens ein Setter) mit dem Namen `componentFactory` (d.h. mindestens die Methode `setComponetFactory`).

Des Weiteren benötigt die Klasse `CalculatorFactoryImpl` eine Methode `newCalculatorFrame()`, die dann (mit Hilfe der `ComponentFactory`) ein Calculator-Frame erzeugt. Den Code dazu können Sie z.B. aus der `showCalculator`-Methode von `GUI01FactoryMethods` kopieren und anpassen.

Die für Spring nötigen Bibliotheken sind als Gradle-Abhängigkeiten in das Projekt eingebunden.

Der Titel des Fensters wird in der Methode `newCalculatorFrame()` festgelegt. Erweitern Sie die Klasse `CalculatorFactoryImpl` so, dass auch der Titel über das Spring-Konfigurationsfile definiert werden kann (und dass so ein Default-Titel per Konfiguration überschrieben werden kann).