Module Design Patterns
Prof. Dr. Dominik Gruntz
Prof. Dr. Wolfgang Weck

Worksheet Spring

As part of the last worksheet you have defined an *abstract factory* that can be used to create the components of the various product families (GUI controls for AWT, Swing, SWT, or JavaFX), and you have seen how the factory can be registered in the context.

You should now develop a Spring variant, in which the configuration can be defined in a Spring configuation file. To create the GUI of the calculator, we have defined the interface `CalculatorFactory` and provide the body of class `CalculatorFactoryImpl`, which implements this interface.

```java
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;
    }

}
```

Method `newCalculatorFrame` creates the calculator GUI and requires an instance of a `ComponentFactory`.

The Spring configuration file `gui-context.xml` looks as follows:

```xml
<!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>
```

You have to replace the class name `patterns.factory.gui.FactoryFX` by the name of your factory class.

The main program can access the configuration with the statement

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

and request the bean `calculatorFactoryBean` defined in this configuration with the statement

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

On this object (that is of type `CalculatorFactory`) the calculator frame can be created using method `newCalculatorFrame`.

The main program then looks as follows (this class is also included in the project):

```
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);
    }
}
```

Your task is to implement class `CalculatorFactoryImpl` by extending the given fragment. This class shall be parameterizable with a concrete `ComponentFactory` instance. The factory to be used can be defined in the Spring configuration file. According to the configuration file on the front page, class `CalculatorFactoryImpl` must define a property with the name `componentFactory` (at least a setter, i.e. at least method `setComponetFactory` must be available).

In addition, class `CalculatorFactoryImpl` requires a method `newCalculatorFrame`(), which creates a calculator frame (using the `ComponentFactory` passed to this class). Copy and adjust the code from method `showCalculator` declared in class `GUI01FactoryMethods`.

The jar files required by Spring are declared as Gradle dependencies in the file `build.gradle`.

The title of the calculator window is set in method `newCalculatorFrame()`. Extend class `Calculator-FactoryImpl` such that this title can be defined using the Spring configuration file as well (and that this way a default title can be overwritten with the Spring configuration).