

Ex. 12 - Team 100

11. Februar 2016

Aufgabe 1

java.awt:

Component ist *java.awt.Component*, denn es ist *abstract*. Es enthält einige Methoden, welche die *Leaves* auch enthalten.

Composite ist *java.awt.Container*. Die Klasse enthält eine *java.util.List<Component>* und implementiert die Methoden *add(Component)*, *remove(Component)*, *remove(int)* und *getComponent(int)*, welches als das *getChild(int)* im Skript fungiert. *Leaves* sind Subklassen von *Component*, zum Beispiel *java.awt.Button* und *java.awt.Label*.

Warum wurde hier das Composite Pattern gewählt?

Eine graphische *java.awt* Benutzeroberfläche kann viele Elemente enthalten. Diese sollen auch als Gruppen behandelt werden können.

javax.swing:

Component ist die abstrakte Methode *JComponent*.

Leaves sind die Subklassen von *JComponent* wie z.B. *JTree* oder *Abstract Button*.

Composite ist *JPanel*. Diese Klasse erbt von *JComponent* und dient als Behälter für die *JComponents*. Sie besitzt die Methoden *add(Component)*, *remove(Component)* und *getComponent(int)*.

Aufgabe 2

javax.xml.parsers:

SAXParserFactory fungiert als *AbstractFactory*. Eine *ConcreteFactory* kann durch die statische Methode *newInstance(...)* erzeugt werden. Die *FactoryMethod* *newSAXParser()* erzeugt einen *SAXParser*. *SAXParser* ist das *AbstractProduct*-Interface, *concreteProducts* müssen implementiert werden.

javax.xml.transform:

Eine *AbstractFactory* ist *TransformerFactory*, mit *FactoryMethod* *newTransformer()*. *Templates* und *SAXTransformerFactory* sind *concreteFactories*. *Templates* ist das Interface, dass im Sinne des *Abstract Factory Patterns* ein *Abstract-Product* ist, *Transformer* dagegen ein *concreteProduct*.

javax.xml.xpath:

Die *XPathFactory* ist eine *AbstractFactory*. Mit deren statischer Methode *newInstance(...)* können *concreteFactories* erzeugt werden. *FactoryMethod* ist *newXPath()*, *XPath* ist das *ConcreteProduct*; *XPathExpression* etwa, ein *abstract-Product*-Interface.