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 *Leave*s 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.

Aufgabe 2

***javax.xml.parsers*:**

SAXParserFactory fungiert als *AbstractFactory*. Eine *ConreteFactory* 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 Pattern*s ein *AbtractProduct* ist, *Transformer* dagegen ein *concreteProduct*.

***javax.xml.xpath*:**

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