Blatt 5

Punktetabelle

1	2	3	$\sum$
7	5	3	15

@Aufgabe *. . . für die Zukunft*

Kommentieren Sie ab diesem Übungszettel alle Klassen und Methoden ab der Sichtbarkeit `private` und alle Attribute ab der Sichtbarkeit `default` sinnvoll, ordentlich und vollständig mit JavaDoc. Vor allen Dingen sollte der `@author`-Tag verwendet werden, der alle Autoren, die an dem Projekt beteiligt waren, beinhalten sollte. (Fehlende) JavaDoc wird in Zukunft in die Punkte mit einfließen.

Hinweis: Falls Sie Eclipse verwenden, können Sie unter den Einstellungen (Window > Preferences) Eclipse so konfigurieren, dass es Sie bei der Erstellung der JavaDoc (Java > Code Style > Code Templates > Comments) bzw. dem Finden (Java > Compiler > JavaDoc) der fehlenden JavaDoc unterstützt. Sie sollten in Zukunft auch weiterhin all ihren Code kommentieren!

1 UML, JavaDoc und Packages

Zeichnen Sie ein UML Diagramm des Spiels *Die Siedler von Catan*[1]. Von der Basisversion soll zumindest jedes Spielelement (siehe Boxinhalt) abgebildet werden. Attribute und Methoden sollen sinnvoll angelegt werden. Implementieren Sie die Klassen, Attribute und Methoden. Die Funktionalität muss nicht implementiert werden, es reicht wenn die Methodenrümpfe implementiert sind. Kommentieren Sie ihr Programm wie oben beschrieben und erzeugen Sie eine HTML-Version der JavaDoc. Stellen Sie sich vor, sie würden nicht nur das Datenmodell des Spiels implementieren, sondern auch eine Oberfläche und eine Spiellogik. Legen Sie die entsprechenden Pakete hierfür an.

2 Java Generics in der Praxis . . .

Implementieren Sie eine `SortedList` die `Compareable` Objekte beinhaltet. Die Liste soll automatisch beim Hinzufügen das Objekt an die richtige Stelle einfügen. Es soll möglich sein, Elemente zu der Liste hinzuzufügen, sie zu entfernen, mit einer binären Suche den Index eines Elements herauszufinden, zu Überprüfen ob ein Element in der Liste enthalten ist und eine Teilliste zu erstellen, die alle Elemente enthalten, die größer als ein bestimmtes Element sind. Diese Liste soll auf unten aufgeführter Liste basieren. Testen Sie die Liste mit den in der Vorlesung verwendeten Klassen `Cartesian.java` und `Polar.java` (das Interface `Compareable` muss hier noch separat implementiert werden). Finden Sie eine Möglichkeit, das in *eine* Liste sowohl Objekte vom Typ `Polar` und Objekte vom Typ `Complex` in die Liste hinzuzufügen und zu vergleichen (hierfür muss evtl. die Klassendefinition der `SortedList` abgeändert werden). Zudem sollte es möglich sein, die generischen Typen `SortedList<Polar>` und `SortedList<Complex>` zu bilden.

[1] http://de.wikipedia.org/wiki/Die_Siedler_von_Catan

Listing 1: SortedList.java

```
/**
 * Project: PM-Ex-8
 * Date: 07.05.2013
 * This class represents a sorted list. Elements added to the list will be sorted
 *
 * @author Christian Haisjackl
 */
public class SortedList<T extends Comparable<T>> {
	/** The list containing the sorted list. Capacity has to be set with const
	private T list[];



	/**
	 * This method inserts an element to the list. The elements of the list wi
	 *
	 * @param e The element to add
	 * @return TRUE if adding was successful
	 */
	public boolean insert( T e ) {
		return false;
	}



	/**
	 * This method checks, if an element is contained in the list
	 *
	 * @param e The element to search for
	 * @return TRUE if the element is contained
	 */
	public boolean contains( T e ) {
		return false;
	}



	/**
	 * This method searches for a specific element and returns the index of th
	 *
	 * @param e The element to search for
	 * @return The index of the element
	 */
	public int binarySearch( T e ) {
		return 0;
	}
}
```

3 ...und Java Generics in Theorie

Welche Probleme können bei den unten stehenden Beispielen auftauchen? Versuchen Sie, die Aufgaben nicht mit dem Compiler und mit Testen zu lösen, sondern überlegen Sie erst, wo und welche Probleme auftreten könnten (beim Compilieren oder zur Laufzeit, Warnung oder Fehler, in welcher Zeile genau)

```
// Bsp 1
JComboBox<String>[] comboBox = new JComboBox<String>[3];

// Bsp 2
private T[] list = new T[5];

// Bsp 3
List<String> l1 = new Vector<String>();
l1.add( "test" );
List l2 = l1;
List<Integer> l3 = l2;
Integer i = l3.get( 0 );
```