SAARLAND UNIVERSITY
COMPUTER SCIENCE

Arbeitsgruppe Programmierung
Jun.-Prof. Dr. Sebastian Hack
Dipl.-Inform. Christoph Mallon

Programmierung 2 — Übungsblatt 9

Abgabe: **Dienstag, 30. Juni 2009, 11.45 Uhr, Geb. E1 3, Briefkasten im EG**
Lösung mit **Name**, **Matrikelnummer** und **Name des Tutors** beschriften!

Aufgabe 9.1: Entwurfsmuster

1. Sie möchten eine Klasse `Stack` (Keller) implementieren, die über drei Operationen verfügt:
   - `void push(Object)` legt ein Objekt auf den Keller.
   - `Object pop()` nimmt das oberste Objekt vom Keller und liefert es zurück.
   - `Object peek()` liefert das oberste Objekt vom Keller, lässt es aber auf dem Keller liegen.

   Selbstverständlich möchten Sie bereits existierenden Code (wie zum Beispiel die Klasse `LinkedList`) verwenden und nicht alles neu schreiben. Implementieren Sie die Klasse Stack unter Zuhilfenahme der Klasse `LinkedList`. Welches Entwurfsmuster kommt hierbei zum Einsatz?

2. In einem Projekt, an dem Sie arbeiten, existiert eine Klasse `Counter`, die einen Zähler implementiert.

```
1 class Counter {
2   private int count = 0;
3   public void increase() {
4     count = count + 1;
5   }
6 }
```

   Verschiedene andere Objekte zählen einen solchen Zähler bei bestimmten Ereignissen hoch. Das Objekt soll nun so erweitert werden, dass der Zähler andere Objekte benachrichtigen kann, sobald der Zählerstand ein beliebiges Vielfaches einer Zahl n erreicht. Hierbei können sich die zu benachrichtigenden Objekte bei einem Zähler-Objekt anmelden. Erweitern Sie die Klasse Counter um die vorgesehen Eigenschaften. Welches Entwurfsmuster kommt hierbei zum Einsatz?

Aufgabe 9.2: Innere Klassen

Betrachten Sie folgenden Auszug aus der Implementierung der Klasse `LinkedList`.

```
1  class LinkedList<T> implements Iterable<T> {
2      class Entry {
3          Entry(T obj) {
4              this.obj = obj;
5              this.prev = this.next = this;
6          }
7          T obj;
8          Entry prev, next;
9      }
10     class Iter implements Iterator<T> {
11         private Entry next = head.next;
12         public T next() { return next.obj; }
13         public boolean hasNext() { return next != head; }
14         public void remove() { }
```

```
15          }
16          private Entry head = new Entry(null);
17          public Iterator<T> iterator() {
18                return new Iter();
19          }
20          /* ... andere Methoden */
21  }
```

In dieser Aufgabe interessieren wir uns nur für die inneren Klassen. Daher ist die Implementierung der eigentlichen Funktionalität ausgespart. Beantworten Sie folgende Fragen:

1. Kann man die Vereinbarung ...
   - `class Entry { ... }` durch `static class Entry { ... }` ersetzen?
   - `class Iter { ... }` durch `static class Iter { ... }` ersetzen?

   Begründen Sie Ihre Antwort!
2. Modifizieren Sie die inneren Klassen, die nicht ohne Weiteres mit `static` vereinbar sind, so, dass man sie auch mit `static` vereinbaren kann.
3. Erstellen Sie eine Version des obigen Auszugs in der, wo immer möglich, anonyme Klassen verwendet werden.

Aufgabe 9.3: Graphische Benutzerschnittstelle (GUI)

Implementieren Sie eine Swing-Komponente, die Polynome (wie in Aufgabe 5.1 entwickelt) graphisch darstellen kann. Die Klasse soll einen Konstruktor mit folgenden Argumenten besitzen:

- Ein Objekt der Klasse Polynom (aus Aufgabe 5.1) welches darzustellen ist.
- Die Größe der Komponente (siehe `java.awt.Dimension`)
- Die Skalierung. Sie gibt an, wie viele Pixel auf den Achsen zwischen zwei ganzzahligen Werten des Polynoms liegen.

Beachten Sie beim Zeichnen des Polynoms, dass das Swing-Koordinatensystem, im Gegensatz zum mathematischen Koordinatensystem, oben links beginnt und die y-Koordinaten nach unten aufsteigend sind. Schreiben Sie des Weiteren ein Hauptprogramm, um Ihre Komponente zu testen.

Beispiel: Folgende Punkte werden gezeichnet, wenn das Polynom x^2 dargestellt werden soll, die Höhe und Breite der Komponente 300 ist sowie eine Skalierung von 100 gewählt wird:

$$(0, 300), (1, 299), (2, 299), \ldots, (100, 200), \ldots$$

Beachten Sie hierbei, dass Koordinaten außerhalb der Komponente liegen können und somit nicht mehr gezeichnet werden müssen (z.B. $(200, -100)$).

Hinweis: Falls Sie die Aufgabe 5.1 nicht bearbeitet haben, können Sie zum Testen Ihrer Komponente folgende Klasse verwenden, die das Polynom x2 implementiert:

```
1  class Polynom {
2    public double eval(double x) {
3      return x * x;
4    }
5  }
```