Dr. Andreas Abel
Dr. Ulrich Schöpp
Dr. Steffen Jost

Ludwig-Maximilians-Universität München
Institut für Informatik
13. Dezember 2010

9. Musterlösung zur Vorlesung

Einführung in die Informatik: Programmierung und Software-Entwicklung

Aufgabe 9-1. (4 Punkte) (Geben Sie alle `.java`-Dateien Ihrer Lösung ab)

Implementieren Sie das folgende Spiel: In einem Grafikfenster werden dem Spieler nacheinander Kreise gezeigt, welche der Spieler in der gleichen Reihenfolge anklicken muss, wie die Kreise erschienen sind. Mit jedem Klick, den der Spieler macht, erscheinen kurz nacheinander zwei neue Kreise. Klickt der Spieler einen falschen Kreis an, so hat er verloren. Die Punktzahl des Spiels ist die Anzahl der richtig angeklickten Kreise.

Auf der Homepage können Sie zwei Beispiel-Lösungen zu dieser Aufgabe als ausführbare `jar`-Dateien herunterladen (mit Doppelclick oder mit `java -jar datei.jar` ausführen). Beide Beispiel-Lösungen würden die volle Punktzahl erreichen, auch wenn Sie unterschiedlich aufwändig implementiert wurden.

Es bleibt Ihnen dieses Mal völlig freigestellt, wie Sie diese Aufgabe lösen. Sie dürfen Klassen oder Code aus allen Ihren bisherigen Abgaben oder den Musterlösungen wiederverwenden.

Mögliches Vorgehen bei der Lösung. Ein mögliches Vorgehen bei dieser Aufgabe ist, die noch anzuklickenden Kreise in einer verketteten Liste zu speichern, in der Reihenfolge wie sie angeklickt werden müssen. Die Liste repräsentiert also eine Warteschlange ("Queue"). Bei jedem Klick wird geprüft, ob der Klick innerhalb des Kreises an der ersten Stelle in der Warteschlange lag; falls ja, wird dieser erste Kreis aus der Warteschlange entfernt und zwei neue Kreise werden hinten in die Warteschlange eingereiht. Ansonsten wird das Spiel beendet.

Es bietet sich an, für die Repräsentierung von Kreisen eine Klasse `Circle` zu schreiben. Für verkettete Listen von solchen Kreisen können Sie dann die Klasse `LinkedList<Circle>` aus der Java-Standardbibliothek verwenden. Diese Klasse stellt Methoden wie `addFirst`, `addLast`, `removeFirst` sowie Iteratoren, wie aus der Vorlesung bekannt, bereit. Um diese Klasse zu benutzen fügen Sie "`import java.util.LinkedList;`" am Anfang ihrer Datei ein. Alternativ können Sie natürlich auch die Listen-Klasse aus Aufgabe 8-3 anpassen und erweitern.

Für technische Dinge, z.B. wie man ein Grafikfenster öffnet oder die Koordinaten eines Klicks abfragt, sei an Aufgabe 3-2 und die Beispielprogramme `Beispiel1.java` und `Beispiel2.java` der Klasse `CanvasFrame` erinnert.

In dem Spiel sollen zwei neue Kreise stets kurz nacheinander, d.h. mit leichter Zeitverzögerung erscheinen. Eine Zeitverzögerung können Sie mit der neu zur Klasse `CanvasFrame` hinzugefügten Methode `sleep` erreichen. Ein Aufruf `frame.sleep(100)` hält die Programmausführung für 100ms an. Eine beispielhafte Verwendung finden Sie in `ListDemo.java` aus Aufgabe 8-3.

Neue Kreise sollen immer an einer zufälligen Position erscheinen. Zufallszahlen erhalten Sie in Java durch die Klasse `Random`. Im Kopf ihrer Klasse fügen Sie folgende Zeile ein: "`import java.util.Random;`" Dann können Sie mit "`Random rdm = new Random();`" ein Objekt dieser Klasse erzeugen. Mit dem Methodenaufruf "`int i = rdm.nextInt(23)`" belegen Sie dann die

Variable `i` mit einer Zufallszahl zwischen 0 und 22. Als Beispiel können Sie sich hierfür auch wieder die Klasse `ListDemo` aus Aufgabe 8-3 anschauen, z.B. in Zeile 34 oder Zeile 56.

LÖSUNG
Auf der Homepage der Vorlesung wird der Quelltext beider Beispiele zur Verfügung gestellt. Fragen zu den Lösungen sollten am Besten in den Übungen oder notfalls in einer Sprechstunde behandelt werden.

Aufgabe 9-2. (4 Punkte) (Abgabeformat: `.txt` oder `.pdf`)
Wir betrachten folgenden Ausschnitt der Realität: Es gibt Schiffe und als Spezialfälle davon Segelschiffe, Motorschiffe und RoRo-Fähren. Jedes Schiff hat eine Tonnage (in Bruttoregistertonnen, gegeben als Wert des Typs `long`). Jedes Segelschiff hat zusätzlich zur Tonnage eine Segelfläche (in Quadratmetern, gegeben als Wert des Typs `int`). Jedes Motorschiff hat zusätzlich zur Tonnage eine Motorleistung (in Kilowatt, gegeben als Wert des Typs `double`). Jede RoRo-Fähre (RoRo = Roll On Roll Off) ist ein spezielles Motorschiff, welches eine bestimmte Anzahl an PKW und LKW transportieren kann (zwei Werte des Typs `int`).

Zeichnen Sie ein UML-Diagramm mit den Klassen Schiff, Segelschiff, Motorschiff und RoRo-Fähre zur Repräsentation dieses Ausschnitts der Realität. Tonnage, Segelfläche, Motorleistung und Transportkapazitäten für Kraftwagen sollen durch private Instanzvariablen der entsprechenden Klassen repräsentiert sein. Beachten Sie, dass beispielsweise Segelschiffe keine Motorleistung, und Motorschiffe keine Segelfläche haben, etc. und andere Schiffe haben weder das eine noch das andere. Da Konstruktoren und Methoden nicht spezifiziert wurden, fallen diese in dem UML-Diagramm weg.

Sie sollten sich jedoch Gedanken machen, in welcher Beziehung die Klassen am Besten stehen sollten (Stichworte: Vererbung, Komposition, Assoziation). Begründen Sie kurz Ihre Wahl!

LÖSUNG

Jeden Schiffstyp repräsentieren wir mit einer eigenen Klasse. Die Klasse Schiff bekommt lediglich ein Attribut `tonnage` des Typs `long`.

Da jedes Segelschiff auch ein Schiff ist, lassen wir die Klasse Segelschiff von der Klasse Schiff erben. Damit erben Segelschiffe automatisch das Attribut `tonnage` und wir müssen nur noch ein Attribut für die Segelfläche hinzufügen.

Motorschiffe sind sicherliche keine Segelschiffe, aber Schiffe. (Jedes Motorschiff kann sich als Schiff ausgeben, aber es kann eben nicht ein Segelschiff spielen.) Damit erbt die Klasse Motorschiff nur von der Klasse Schiff. Wir müssen lediglich ein Attribut für die Motorleistung hinzufügen.

RoRo-Fähren sind spezielle Motorschiffe, also lassen wir die Klasse für RoRo-Fähren einfach von der Klasse Motorschiff erben. Damit ist die Klasse der RoRo-Fähren auch automatisch ein Erbe der Klasse Schiff, da Vererbung transitiv ist.

Wir erhalten folgendes UML-Diagramm, welches zusätzlich noch Getter- und Setter-Methoden enthält, welche explizit nicht gefragt waren. In der erwarteten Lösung dürfen die unteren Kästchen für die Methoden also leer bleiben. Die Namen der Attribute dürfen natürlich auch variieren, so lange der Name noch sinnvoll gewählt wurde. Der Typ und die Deklarierung als private musste für diese Aufgabe jedoch zwingend angegeben werden.