

Programmierkurs Java

Dr. Dietrich Boles

Aufgaben zu UE29-JDK (Stand 09.06.2010)

Aufgabe 1:

Bei dieser Aufgabe sollen Sie einen kleinen Ausschnitt aus einem Kartenspiel implementieren. Karten in diesem Spiel haben den Typ „kreuz“, „pik“, „herz“ und „karo“ und von jedem Typ gibt es die Kartenwerte „ass“, „2“, „3“, „4“, „5“, „6“, „7“, „8“, „9“, „10“, „bauer“, „dame“, „koenig“.

Aufgabe (a): Implementieren Sie zunächst eine Klasse `Karte`, die die Karten des Kartenspiels repräsentieren kann.

Für das Kartenspiel benötigen Sie eine Mischmaschine, die die Karten mischen kann. Objekte einer entsprechenden Klasse `MischMaschine` müssen in der Lage sein:

1. Karten aufzunehmen,
2. die enthaltenen Karten zu mischen,
3. alle enthaltenen Karten zu durchlaufen.

Sie kennen die Klasse `java.util.ArrayList` und wissen, dass diese bereits die Anforderungen (1) und (3) über die beiden Methoden `add` und `iterator` erfüllt. Also können Sie zur Realisierung der Klasse `MischMaschine` die Klasse `java.util.ArrayList` und das Konzept der Vererbung nutzen und müssen lediglich eine Methode `mischen`, die das Mischen realisiert, hinzufügen.

Aufgabe (b): Implementieren Sie eine Klasse `MischMaschine` durch Ableitung von der Klasse `java.util.ArrayList`, die Objekte der Klasse `Karte` (ausschließlich) aufnehmen, die enthaltenen Karten mischen und alle enthaltenen Karten durchlaufen kann.

Aufgabe (c): Schreiben Sie ein Programm, in dem alle Karten des Kartenspiels erzeugt und einer Mischmaschine übergeben werden. Die Mischmaschine soll die Karten mischen und anschließend sollen die Karten in der gemischten Reihenfolge auf den Bildschirm ausgegeben werden.

Aufgabe 2:

Ein Wörterbuch im Sinne der folgenden Aufgabe ist dadurch charakterisiert, dass es mehrere Begriffe enthält, denen jeweils ein oder mehrere Übersetzungen zugeordnet sind.

Aufgabe (a): Implementieren Sie eine Klasse `Woerterbuch` mit folgenden Methoden:

1. Eine Methode `ein fuegen` mit zwei `String`-Parametern, nämlich einem Begriff und einer (u. U. weiteren) Übersetzung.
2. Eine Methode `getUebersetzungen`, die einen `String` als Parameter übergeben bekommt und als `String-Array` die im Wörterbuch gespeicherten Übersetzungen dieses Begriffs liefert.

Nutzen Sie zur Realisierung der internen Datenstruktur die Klasse `java.util.HashMap`.

Aufgabe (b): Schreiben Sie mit Hilfe der Klasse `Woerterbuch` ein kleines Programm, in dem ein Benutzer zunächst eine Menge an Begriffen und Übersetzungen eingeben kann und er sich anschließend zu einzelnen einzugebenden Begriffen die gespeicherten Übersetzungen ausgeben lassen kann.

Aufgabe 3:

Schreiben Sie folgende Programme, indem Sie existierende Klassen des JDK nutzen:

- (1) Schreiben Sie ein Programm, das berechnet, wie viele Tage Sie aktuell alt sind (siehe Klassen `java.util.Calendar` und `java.util.Date`).
- (2) Schreiben Sie ein Programm, das die Ziehung der Lottozahlen (6 Zahlen zwischen 1 und 49) realisiert (siehe Klasse `java.util.Random`).
- (3) Schreiben Sie ein Programm, das vier `int`-Werte einliest, sie auf einen Stack legt und anschließend in der umgekehrten Reihenfolge wieder auf den Bildschirm ausgibt (siehe Klasse `java.util.Stack`).
- (4) Schreiben Sie ein Programm, das einen `double`-Wert vom Benutzer einliest und den Sinus, Cosinus und Tangens sowie die Quadratwurzel auf dem Bildschirm ausgibt (siehe Klasse `java.lang.Math`)

Aufgabe 4:

Implementieren Sie mit Hilfe der Klasse `java.util.ArrayList` die folgende Klasse `Stack`.

```
/**
 * A Last-In-First-Out (LIFO) stack of objects.
 */
class Stack extends ArrayList {
    /**
     * Pushes an item onto the stack.
     * @param item the item to be pushed on.
     */
    int push(int item);

    /**
     * Pops an item off the stack.
     */
    int pop();
}
```

```

/**
 * Peeks at the top of the stack.
 */
int peek();

/**
 * Returns true if the stack is empty.
 */
boolean empty();

/**
 * Sees if an object is on the stack.
 * @param o the desired object
 * @return the distance from the top, or -1 if it is not found.
 */
int search(int o);
}

```

Aufgabe 5:

Bei dieser Aufgabe geht es um die Definition einer Klasse *Kalender* zur Terminverwaltung. Die Klasse soll eine Methode *neuerTermin* definieren, über die zu einem bestimmten Datum ein bestimmtes Ereignis in einem Kalender gespeichert werden kann. Außerdem soll sie eine Methode *liefereEreignisse* definieren, die zu einem gewünschten Datum alle im Kalender gespeicherten Ereignisse liefert. Die Repräsentation von Datum und Ereignis erfolgt als Strings.

Ein Testprogramm für die Klasse *Kalender* könnte so aussehen:

```

public static void main(String[] args) {
    Kalender kalender = new Kalender();
    kalender.neuerTermin("03.09.2007", "Geburtstag Oma");
    kalender.neuerTermin("01.01.2007", "Ausschlafen");
    kalender.neuerTermin("03.09.2007", "Schulfest");
    // ...
    String[] ereignisse =
        kalender.liefereEreignisse("03.09.2007");
    for (int i = 0; i < ereignisse.length; i++) {
        System.out.println(ereignisse[i]);
    }
}

```

Das Programm liefert die Ausgabe:

```

Geburtstag Oma
Schulfest

```

Aufgabe: Definieren Sie eine Klasse *Kalender* entsprechend der oben beschriebenen Anforderungen.

Hinweis: Sie können (müssen aber nicht) zum Speichern die Klasse *java.util.ArrayList* verwenden:

```

public class ArrayList<E>

```

```

// Constructs an empty list.
public ArrayList() { ... }

// Appends the specified element to the end of this list.
public boolean add(E o) { ... }

// Returns the number of elements in this list.
public int size() {

// Returns the element at the specified position in this list.
public E get(int index) { ... }

// Returns an array containing all of the elements in this list in the
// correct order; the runtime type of the returned array is that of the
// specified array. If the list fits in the specified array, it is
// returned therein. Otherwise, a new array is allocated with the
// runtime type of the specified array and the size of this list.
public <T> T[] toArray(T[] a) { ... }
}

```

Aufgabe 6:

Schreiben Sie ein Java-Programm, das den DOS-Befehl „dir“ simuliert. Mit diesem Befehl werden bestimmte Informationen aller Elemente eines Verzeichnisses (Dateien und Verzeichnisse) auf dem Bildschirm angezeigt. Der Verzeichnisname soll dabei dem Java-Interpreter beim Programmaufruf übergeben werden.

Die Ausgabe soll für jede Datei bzw. jedes Verzeichnis in einer separaten Zeile erfolgen und folgendes Format haben:

- Datum der letzten Änderung
- „<DIR>“ falls es sich um ein Verzeichnis handelt, ansonsten 5 Leerzeichen
- „r“, falls das Element lesbar ist, ansonsten „-“
- „w“, falls das Element schreibbar ist, ansonsten „-“
- Anzahl an Bytes
- Elementname

Nutzen Sie die JDK-Klasse `java.io.File`

Beispielausgabe:

```

Wed Apr 04 11:32:02 CEST 2012 <DIR> rw 0 C:\Users\dibo\.android
Wed Feb 16 09:57:56 CET 2011 <DIR> rw 0 C:\Users\dibo\.appinventor
Fri Apr 01 11:54:28 CEST 2011      rw 155 C:\Users\dibo\.appletviewer
Tue Feb 28 15:37:54 CET 2012 <DIR> rw 0 C:\Users\dibo\.argouml
Wed Mar 16 14:39:40 CET 2011      rw 96 C:\Users\dibo\.asadminpass
Fri May 07 16:35:29 CEST 2010 <DIR> rw 0 C:\Users\dibo\.jalopy.15
Mon Nov 29 16:57:28 CET 2010 <DIR> r- 0 C:\Users\dibo\.jeliot
Tue Dec 14 16:15:51 CET 2010     rw 172 C:\Users\dibo\.jline-jython.history
...

```

Aufgabe 7:

In einem Kreis stehen n Kinder (durchnummeriert von 1 bis n). Mit Hilfe eines m -silbigen Abzählreims wird das jeweils m -te unter den noch im Kreis befindlichen Kindern ausgeschieden, bis kein Kind mehr im Kreis steht.

Schreiben Sie ein Java-Programm, das nach Vorgabe von n (positive Zahl) und m (positive Zahl) die Nummern der Kinder in der Reihenfolge ihres Ausscheidens angibt.

Beispiel: Für $n=6$ und $m=5$ ergibt sich die Folge 5, 4, 6, 2, 3, 1.

Hinweis: Nutzen Sie die Klasse `java.util.ArrayList`

Aufgabe 8:

Mit der **Java Reflection API** kann man zur Laufzeit Informationen über Objekte und Klassen erhalten. Genau das sollen Sie in dieser Aufgabe auch tun. Sie benötigen dafür folgende Klassen des API:

```
package java.lang;
public class Object {

    // zu jeder Klasse gibt es in Java ein so genanntes
    // Klassenobjekt; dieses kann man durch den Aufruf
    // dieser Methode für ein Objekt der Klasse erfragen;
    // durch Aufruf von (new String()).getClass() lässt sich
    // bspw. das Klassenobjekt der Klasse String ermitteln
    public java.lang.Class getClass()
}

```

```
package java.lang;
// repräsentiert eine Klasse
public class Class {

    // liefert den vollständigen Namen der Klasse
    public String getName()

    // liefert alle Attribute der Klasse
    public java.lang.reflect.Field[] getDeclaredFields()
}

```

```
package java.lang.reflect;
// repräsentiert ein Attribut
public class Field {

    // liefert eine Kodierung aller Modifier
    // (public, static, final, ...) eines Attributs;
    // übergibt man den Code der Methode Modifier.toString,
    // erhält man eine passende Stringrepräsentation
    public int getModifiers()
}

```

```

// jeder Typ wird in Java durch eine Klasse
// repräsentiert; diese Methode liefert das Klassenobjekt
// des Typs des Attributs; ist das
// Attribut bspw. vom Typ int, wird das Klassenobjekt
// des Standardtyps int geliefert
public java.lang.Class getType()

// liefert den Namen eines Attributes
public String getName()
}

```

Aufgabe: Implementieren Sie auf der Grundlage der Java Reflection API eine Funktion

```
static void printClass(Object obj)
```

die die Klassendefinition (eingeschränkt auf Klassenname und Attribute (jeweils Modifier, Typ und Name)) der Klasse von `obj` auf den Bildschirm ausgibt.

Beispiel:

Gegeben sei die folgende Klasse

```
class CX {
    int i;
    static int s;
    final public static int k = 4711;
}

```

Ein Aufruf der Funktion `printClass` mittels

```
printClass(new CX());
```

sollte dann folgende Bildschirmausgabe produzieren:

```
class CX {
    int i;
    static int s;
    public static final int k;
}

```

Ein Aufruf mittels

```
printClass(new java.lang.Boolean(false));
```

sollte folgende Ausgabe liefern:

```
class java.lang.Boolean {
    public static final java.lang.Boolean TRUE;
    public static final java.lang.Boolean FALSE;
}

```

```

public static final java.lang.Class TYPE;
private final boolean value;
private static final long serialVersionUID;
}

```

Aufgabe 9:

In nahezu jedem Lehrbuch finden Sie im hinteren Teil so genannte Indexe bzw. Sachwortverzeichnisse. Hier werden wichtige Begriffe, die im Buch vorkommen, sowie die jeweiligen Seitenzahlen, auf denen die Begriffe vorkommen, aufgelistet.

Implementieren Sie eine ADT-Klasse `Index`, die einen solchen Index repräsentiert. Die Klasse soll folgende Methoden zur Verfügung stellen:

- Einen Default-Konstruktor, der einen leeren Index erstellt.
- Einen Copy-Konstruktor; der neue Index soll dabei alle Einträge enthalten, die der als Parameter übergebene Index zum Zeitpunkt des Aufrufs enthält.
- Eine Methode `clone` zum Klonieren eines Index (Überschreiben der von der Klasse `Object` geerbten Methode `clone`). Der neue Index soll dabei alle Einträge enthalten, die der aufgerufene Index zum Zeitpunkt des Aufrufs enthält.
- Eine Methode `hinzufuegen`, die einen Begriff (String) sowie eine Seitenzahl (int-Wert) als Parameter übergeben bekommt und die den Begriff mit der Seitenzahl im Index vermerkt. Dabei gilt: Begriffe dürfen nicht mehrfach im Index vorkommen und zu jedem Begriff darf eine Seitenzahl höchstens einmal im Index vorkommen.
- Eine Methode `toString`, die eine String-Repräsentation des aktuellen Index liefert (Überschreiben der von der Klasse `Object` geerbten Methode `toString`). Die String-Repräsentation soll dabei folgendermaßen aufgebaut sein: Für jeden im Index vermerkten Begriff soll der String jeweils eine Zeile der folgenden Form enthalten: `<begriff>: <seitenzahl 1>`
`<seitenzahl 2> ... <seitenzahl n>` (also bspw.: `Hamster: 2 45`
`123`)

Zusatz:

Geben sie sowohl die Einträge des Index als auch die Seitenzahlen sortiert aus (lexikographische bzw. natürliche Ordnung der Begriffe bzw. der Seitenzahlen). Suchen Sie im JDK nach geeigneten Klassen bzw. Methoden zum Sortieren.

Testen:

Ob Ihre Klasse (zumindest teilweise) korrekt ist, können Sie testen, indem Sie das folgende Testprogramm ausführen:

```

public class IndexTest {

    public static void main(String[] args) {
        Index index = new Index();
    }
}

```

```
index.hinzufuegen("Objekt", 1);
index.hinzufuegen("Objekt", 15);
index.hinzufuegen("Objekt", 3);
index.hinzufuegen("Objekt", 15);

Index index2 = new Index(index);

index.hinzufuegen("Hamster", 45);
index.hinzufuegen("Hamster", 2);
index.hinzufuegen("Hamster", 199);
index.hinzufuegen("Hamster", 45);

System.out.println(index);
System.out.println(index2);

    }

}
```

Es sollte folgende Ausgabe (unsortiert) erscheinen:

```
Objekt: 1 15 3
Hamster: 45 2 199
```

```
Objekt: 1 15 3
```