

Einführung in die Programmierung

Semestralklausur – Java

(Lösungsvorschlag)

1 Die Klasse ArrayList

In einer Dokumentation der Bibliothek `java.lang` finden Sie (auszugsweise) folgende Definition der Klasse `java.lang.ArrayList`:

```
/**
 * Eine ArrayList ist eine Liste, auf deren Elemente sowohl per Index
 * (wie bei einem array) als auch über die klassischen Listenoperationen
 * zugegriffen werden kann.
 * Die Indizes laufen von 0 bis einschließlich size()-1, wobei size()
 * die Anzahl der Elemente einer ArrayList angibt.
 */
public class ArrayList extends AbstractList
    implements List, RandomAccess, Cloneable, Serializable
{
    /** Konstruktor zum Erzeugen einer leeren Liste */
    public ArrayList()
    { /* ... */ }

    /** liefert die Anzahl der enthaltenen Elemente */
    public int size()
    { /* ... */ }

    /** fügt das übergebene Objekt an der angegebenen Position 'index' ein.
     * Das Objekt an der Position index, sowie alle nachfolgenden Objekte
     * werden um eine Position nach hinten verschoben.
     * - Liefert eine IndexOutOfBoundsException, wenn 'index' < 0 oder
     *   index > size() ist.
     */
}
```

```

    * - Falls index==size() wird das Element an letzter Stelle angehängt.
    */
public void add(int index, Object o)
{ /* ... */ }

/** hängt das übergebene Objekt an das Ende der Liste an */
public boolean add(Object o)
{ /* ... */ }

/** liefert das Objekt an Position 'index' als Ergebnis */
public Object get(int index)
{ /* ... */ }

/** liefert das Objekt an Position 'index' als Ergebnis.
 * Das Objekt wird zudem gelöscht und alle nachfolgenden Elemente
 * werden um eine Position nach vorne verschoben.
 */
public Object remove(int index)
{ /* ... */ }
}

```

1.1 Aufbau von Listen

Welche Code-Sequenzen erzeugen die Liste ["Meier", "Müller", "Huber"] in genau dieser Reihenfolge? Die ArrayList `a` enthalte zu Beginn jeweils eine leere Liste.

- `a.add("Meier");`
`a.add("Müller");`
`a.add("Huber");`
- `a.add(0,"Müller");`
`a.add(1,"Huber");`
`a.add(0,"Meier");`
- `a.add("Huber");`
`a.add("Müller");`
`a.add("Meier");`
- `a.add(2,"Huber");`
`a.add(1,"Müller");`
`a.add(0,"Meier");`

Nur die ersten beiden Sequenzen sind richtig. Die dritte Code-Sequenz erzeugt die Liste in umgekehrter Reihenfolge. Die letzte Sequenz erzeugt bereits nach dem ersten Befehl einen Fehler (`IndexOutOfBoundsException`).

1.2 Erzeugen einer ArrayList

Welche der folgenden Befehle erzeugen ein **neues** Objekt vom Typ ArrayList?

- ArrayList a;
- a = new ArrayList;
- a = new ArrayList();
- a = ArrayList.ArrayList();
- a = b; (b enthalte ein Objekt vom Typ ArrayList)

Nur der dritte Befehl ist richtig. Der zweite sowie der vierte Befehl sind keine syntaktisch korrekten Aufrufe des Konstruktors. Der erste Befehl definiert lediglich eine Variable vom Typ ArrayList, erzeugt aber noch kein Objekt. Nach der Zuweisung a=b verweisen a und b auf das selbe Objekt. Es wird jedoch kein neues Objekt erzeugt.

1.3 Löschen von Elementen in einer ArrayList

Welche Befehle zum Löschen eines Elements in der ArrayList a sind richtig (a sei nicht leer)?

- a.remove("Herbert");
- ArrayList.remove(a,7);
- a.remove();
- a.remove(a.size()-1);
- a.remove(0);

Nur die letzten beiden Befehle sind richtig – sie löschen das letzte bzw. das erste Element der Liste. Der erste und dritte Befehl stellen nicht den erforderlichen Parameter vom Typ int bereit, der zweite Befehl wäre nur für eine static-Methode richtig.

1.4 Schleifen

Gegeben seien die folgenden beiden Schleifen. Beschreiben Sie den Inhalt der Listen a und b nach Beendigung der beiden Schleifen.

- a)

```
ArrayList a = new ArrayList();
for(int i=0;i<5;i++) {
    a.add("#"+i);
}
```

```

b)  ArrayList b = new ArrayList();
    for(int i=0;i<5;i++) {
        b.add(0, "#"+i);
    }

```

a enthält die Zeichenketten "#0", "#1", ..., "#4" (jeweils vom Typ *String*) in aufsteigender Reihenfolge. *b* enthält die gleichen Zeichen in absteigender Reihenfolge, also "#4", "#3", ..., "#0".

1.5 Schleifen, Teil 2

Gegeben sei eine *ArrayList a*, die Zeichenketten vom Typ *String* (z.B. Vornamen) enthält. Beschreiben Sie kurz, was die beiden folgenden Methoden bewirken, wenn Sie mit *a* als Parameter aufgerufen werden.

```

a)  public int methode1(ArrayList a) {
        for(int i=0;i<a.size();i++)
            if ( ((String)a.get(i)).equals("Andreas") ) return i;
        return -1;
    }

```

```

b)  public void methode2(ArrayList a) {
        while ( ((String)a.get(0)).equals("Andreas") == false )
            a.remove(0);
    }

```

methode1 überprüft von vorne alle Element der *ArrayList*, ob sie die Zeichenkette "Andreas" enthalten. Sobald die erste solche Zeichenkette gefunden wird, wird deren Index als Funktionswert geliefert. Wird der Name "Andreas" nicht gefunden, liefert die Funktion -1 als Wert.

methode2 löscht, beginnend von vorne, jedes Element der Liste, das nicht gleich der Zeichenkette "Andreas" ist. Sobald das erste Element gefunden wird, bricht die Methode ab. Die Methode liefert keinen Rückgabewert. Nach dem Aufruf enthält *a* jedoch nur noch den (hinteren) Teil der ursprünglichen Liste, der mit dem ersten Auftreten von "Andreas" beginnt. Ist in der Liste der Vorname "Andreas" überhaupt nicht vorhanden, wird die Methode eine *Exception* liefern!

2 KFZ-Zulassungsstelle

Für eine KFZ-Zulassungsstelle sollen Klassen zur Speicherung von Fahrzeugen und Fahrzeughaltern gespeichert werden.

2.1 Fahrzeuge

Implementieren Sie zur Speicherung von Fahrzeugen als Objekte eine Klasse Fahrzeug. Für jedes Fahrzeug soll nur der Hubraum (in ccm, ganzzahliger Wert) und das Kennzeichen des Fahrzeugs (Zeichenkette) gespeichert werden. Als Methoden sind zu implementieren:

- ein Konstruktor, der Hubraum und Kennzeichen des Fahrzeug erhält
- eine Methode `getKennzeichen`, die das Kennzeichen liefert
- eine Methode `kfzSteuer`, die die KFZ-Steuer berechnet, die für dieses Fahrzeug zu entrichten ist. Die KFZ-Steuer betrage für normale Fahrzeuge pro Jahr 5 Euro je 100 ccm.

```
public class Fahrzeug
{
    protected int hubraum;
    private String kennzeichen;

    public Fahrzeug(int hubraum, String kennzeichen)
    {
        // Hubraum in Kubikzentimeter
        this.hubraum = hubraum;
        // amtliches Kennzeichen
        this.kennzeichen = kennzeichen;
    }

    public String getKennzeichen()
    {
        return kennzeichen;
    }

    public int kfzSteuer()
    {
        return 5*hubraum/100;
    }
}
```

2.2 Fahrzeughalter

Implementieren Sie zur Speicherung der Fahrzeughalter entsprechend eine Klasse Halter: für jeden Halter ist der Name (als Zeichenkette) zu speichern, sowie eine Liste der Fahrzeuge, die er aktuell angemeldet hat.

Neben den Membervariablen soll die Klasse Halter folgende Methoden aufweisen:

- einen Konstruktor, der den Namen des Halters als Parameter hat. Die Liste der Fahrzeuge soll zu Beginn leer sein.
- eine Methode anmelden, die ein Fahrzeug anmeldet, d.h. in die Liste der Fahrzeuge des Halters aufnimmt.
- eine Methode kfzSteuer, die die **gesamte** KFZ-Steuer berechnet, die der Halter für seine Fahrzeuge zu entrichten hat.

Verwenden Sie für die Implementierung der Listen die Klasse ArrayList.

```
public class Halter
{
    // Name des Fahrzeughalters
    private String name;
    ArrayList fahrzeuge;

    public Halter(String name)
    {
        this.name = name;
        fahrzeuge = new ArrayList();
    }

    public void anmelden(Fahrzeug kfz)
    {
        fahrzeuge.add(kfz);
    }

    public void abmelden(String kennzeichen)
    {
        for(int i=0; i<fahrzeuge.size(); i++) {
            Fahrzeug fz = (Fahrzeug) fahrzeuge.get(i);
            if ( kennzeichen.equals(
                fz.getKennzeichen() ) )
            {
                fahrzeuge.remove(i);
            }
        }
    }

    public int kfzSteuer()
    {
        int steuer = 0;
    }
}
```

```

        for(int i=0; i<fahrzeuge.size(); i++)
            steuer = steuer
                + ((Fahrzeug) fahrzeuge.get(i)).kfzSteuer();
        return steuer;
    }
}

```

2.3 Fahrzeuge mit Saisonkennzeichen

Nachdem das System bereits längere Zeit erfolgreich eingesetzt wurde, ermöglicht eine Gesetzesänderung die Ausstellung von Saisonkennzeichen (gültig z.B. von April bis September jedes Jahres). Die KFZ-Steuer für Fahrzeuge mit Saisonkennzeichen beträgt **pro Monat 1 Euro je 200 ccm**.

Sie sollen deshalb eine Klasse SaisonFahrzeug implementieren, die:

- Fahrzeuge mit Saisonkennzeichen als Objekte modelliert und die jährliche KFZ-Steuer berechnet;
- bereits implementierte Klasse Fahrzeug per Vererbung **so weit wie möglich** wiederverwendet;
- zusätzlich die Gültigkeitsdauer des Saisonkennzeichen speichert, d.h. deren ersten und letzten Monat (als Zahl);
- so gestaltet ist, dass die Klasse Halter ohne Änderung auch mit der Klasse SaisonFahrzeug zusammenarbeiten kann.

```

public class SaisonFahrzeug extends Fahrzeug
{
    // erster und letzter Monat der Zulassung
    private int ersterMonat;
    private int letzterMonat;

    public SaisonFahrzeug(int hubraum, String kennzeichen,
                          int ersterMonat, int letzterMonat)
    {
        super(hubraum, kennzeichen);
        this.ersterMonat = ersterMonat;
        this.letzterMonat = letzterMonat;
    }

    public int kfzSteuer()
    {
        return hubraum*(letzterMonat-ersterMonat+1)/200;
    }
}

```

}

2.4 Bonusfrage: Abmelden von Fahrzeugen (keine Punkte, nur Ruhm und Ehre!)

Erweitern Sie die Klasse `Halter` um eine Methode, die ein Fahrzeug eines Halters abmeldet. Die Methode soll das Kennzeichen des Fahrzeugs als Parameter erhalten.

(Lösung siehe obige Klassendefinition)