Assessment-Prüfung in Informatik I

Teil I | Einführung in das Programmieren

20. Dezember 2012

Aufgabe 1	Aufgabe 2	Aufgabe 3	Aufgabe 4
/16	/17	/18	/29

Name: _____

Matrikelnummer: _____ Hauptfach: _____

Aufgabe 1 — 16 Punkte

Kreuzen Sie Zutreffendes an. Zu jeder Frage können mehrere Antworten richtig sein. Bitte beachten Sie, dass Ihnen für jedes falsch gesetzte Kreuz gleich viele Punkte abgezogen werden, wie Sie für ein korrektes Ankreuzen erhalten. Negative Punktzahlen ergeben null Punkte für die betreffende Frage.

(a) **2 Punkte**

Richtige Aussagen sind anzukreuzen:

- ☐ In Java kann eine Klasse von mehreren Klassen erben.
- ☐ Eine Klasse mit einer abstrakten Methode muss ebenfalls abstrakt sein.
- ☐ In Java werden Konstruktoren vererbt.
- ☐ Abstrakte Methoden müssen in einer nicht-abstrakten Subklasse immer implementiert werden.

(b) **2 Punkte**

Richtige Aussagen sind anzukreuzen:

- ☐ Jede Iteration lässt sich auch als Rekursion implementieren.
- ☐ Jede Rekursion lässt sich auch als Iteration implementieren.
- ☐ Mit *protected* deklarierte Methoden sind in nicht Subklassen sichtbar.
- ☐ Mit *protected* deklarierte Methoden können nur von anderen *protected* Methoden aufgerufen werden.

(c) **2 Punkte**

Richtige Aussagen sind anzukreuzen:

- ☐ Wenn eine Klasse die Methode einer Superklasse mit gleicher Signatur erneut implementiert, wird dies als Überladen (overloading) bezeichnet.
- ☐ Wenn eine Klasse die Methode einer Superklasse mit gleicher Signatur erneut implementiert, wird dies als Überschreiben (overriding) bezeichnet.
- ☐ Attribute (Felder) beschreiben das Verhalten eines Objektes.
- ☐ Attribute (Felder) beschreiben den Zustand eines Objektes.

(d) **2 Punkte**

Welche der folgenden Werte sind von einem primitiven Datentyp:

- ☐ "h"
- ☐ 10.0d
- ☐ true
- ☐ new String("Hello Java")

(e) **2 Punkte**

Gegeben sei folgender Code:

```
1 public class Code {
2    private String[] shoppingList = new String[10];
3    public static void main(String[] args) {
4       shoppingList[0] = "Milk";
5       shoppingList[1] = "Eggs";
6       shoppingList[2] = "Coca Cola";
7       System.out.println(shoppingList.length);
8    }
9 }
```

Richtige Aussagen sind anzukreuzen:

- ☐ Es wird 10 ausgegeben.
- ☐ Es wird 3 ausgegeben.
- ☐ Der Code kompiliert nicht.
- ☐ Es wird eine Exception geworfen.

(f) **3 Punkte**

Gegeben sei folgender Code:

```
1  import java.util.ArrayList;
2  public class ShoppingList {
3     public static void main(String[] args) {
4        // *1*
5        shoppingList.add("Milk");
6        shoppingList.add("Eggs");
7        shoppingList.add("Coke");
8        System.out.println(shoppingList.size());
9     }
10 }
```

Welche Code Snippets können an der Stelle // *1* eingefügt werden, so dass der Code ohne Syntaxfehler kompiliert und ohne Exceptions ausgeführt werden kann. Kreuzen Sie alle zutreffend Code Snippets an:

- ☐ ArrayList<String> shoppingList = new ArrayList<String>();
- ☐ ArrayList<Object> shoppingList = new ArrayList<Object>();
- ☐ ArrayList<String> shoppingList = new ArrayList<Object>();
- ☐ ArrayList<Object> shoppingList = new ArrayList<String>();
- ☐ ArrayList<ShoppingList> shoppingList = new ArrayList<ShoppingList>();
- ☐ Keines der Code Snippets ist korrekt.

(g) **3 Punkte**

Gegeben sei folgender Code:

```
1 public interface A {}
2 public abstract class B extends C implements A{}
3 public class C{}
4 public class D extends B{}
```

Kreuzen Sie die syntaktisch korrekten Zuweisungen an:

- ☐ A a = new A();
- ☐ B b = new A();
- ☐ C c = new C();
- ☐ D d = new A();
- ☐ C c = new D();
- ☐ A a = new D();

Aufgabe 2 — 17 Punkte

(a) **7 Punkte**

Das Ausprobieren aller möglichen Teiler ist eine Methode um zu testen, ob eine Zahl *n* eine Primzahl ist. Dabei werden alle potentiellen Teiler von *2* bis *n - 1* ausprobiert: Ist keiner dieser potentiellen Teiler ein echter Teiler, handelt es sich bei *n* um eine Primzahl. Vervollständigen Sie die Methode *public static boolean isPrime(int n, int m)* und implementieren Sie das beschriebene Verfahren mit Hilfe einer Rekursion. Die *main*-Methode im *TestDriver* soll Ihnen zeigen, wie diese Methode verwendet wird um zu testen, ob eine Zahl *n* eine Primzahl ist.

```
1 public class TestDriver {
2    public static void main(String[] args) {
3       System.out.println(isPrime(n, n - 1));
4    }
5 }
```

Hinweis: Echte Teiler sind die Teiler einer Zahl, ausser 1 und der Zahl selbst.

Ihre Lösung:

```
public static boolean isPrime(int n, int m) {




}
```

(b) **5 Punkte**

Gegeben sei folgende Methode die einen bestimmten Algorithmus mit Hilfe einer Rekursion implementiert:

```
1  public int mysteryMethod(int[] a) {
2     if (a.length == 0) {
3        return 0;
4     } else {
5        int[] b = new int[a.length - 1];
6        for (int i = 0; i < a.length - 1; i++) {
7           b[i] = a[i];
8        }
9        return a[a.length - 1] + mysteryMethod(b);
10    }
11 }
```

Implementieren Sie diesen Algorithmus erneut. Anstelle einer Rekursion sollen Sie eine Iteration verwenden.

Ihre Lösung:

```
public int mysteryMethod(int[] a) {




}
```

(c) **5 Punkte**

Der gegebene Code in Aufgabe b) verwendet in Zeile 6 bis 8 eine Iteration innerhalb der Rekursion. Implementieren Sie den Algorithmus aus Aufgabe b) als Rekursion, ohne eine Iteration innerhalb der Methode zu verwenden:

```
public int mysteryMethod(int[] a, int index) {




}
```

Aufgabe 3 — 18 Punkte

(a) *6 Punkte*

Implementieren Sie eine Klasse *Fraction* die einen Bruch (beispielsweise $\frac{2}{3}$) darstellen soll. Verwenden Sie dabei jeweils eine passende Instanzvariable für den Nenner und den Zähler. Beachten Sie zusätzlich, dass alle Methoden dieser Klasse ein neues Bruch-Objekt erzeugen sollen. (D.h. die Klasse Bruch soll *immutable* sein, ähnlich wie die Klasse *java.lang.String*).

(b) *4 Punkte*

Fügen Sie der Klasse *Fraction* eine Methode hinzu, die als Parameter einen anderen Bruch erwartet und diesen Bruch addiert. Die Summe dieser Addition soll ebenfalls als Bruch zurückgeliefert werden.

(c) *4 Punkte*

Erweitern Sie die *Fraction* Klasse um eine Methode, die den Bruch so weit wie möglich kürzt.

(d) *4 Punkte*

Überschreiben Sie die *equals* Methode in Ihrer *Fraction*-Klasse. Die Methode soll *true* zurückliefern, wenn die Brüche die miteinander verglichen werden entweder den gleichen Bruch darstellen, oder ein Bruch ein Vielfaches von dem anderen Bruch ist. Andernfalls soll die Methode *false* zurückliefern. Vervollständigen Sie auch die Methodensignatur.

Ihre Lösung:

```
public _____ equals( _____ o) {




}
```

Aufgabe 4 — 29 Punkte

Bilden Sie nachfolgenden Sachverhalt in Java auf (wo angebracht abstrakte) Klassen, Interfaces, Attribute und Methoden ab.

Problemstellung

Mathematische Ausdrücke können in einem Baum dargestellt werden. Ein Ausdruck ist dabei entweder:

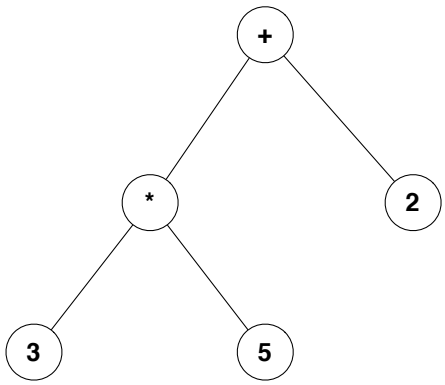
- eine Operation mit linkem und rechtem Operanden, oder
- eine ganze Zahl.

Eine Operation ist entweder:

- eine Addition ($+$)
- eine Subtraktion ($-$)
- eine Multiplikation ($*$), oder
- eine Division ($/$).

Ein Operand ist wiederum ein Ausdruck.

Die Rechenregel "Punkt vor Strich" wird immer eingehalten. Der Ausdruck $3 * 5 + 2$ wird beispielsweise wie folgt dargestellt:



Als Test sollen sie den nachfolgenden Ausdruck in einem *TestDriver* instanzieren. Nach der Instanzierung soll es möglich sein, die Berechnung des gesamten Ausdrucks mit der Abfrage des Wertes des obersten Ausdrucks auszulösen.

$6 * (12 - 2) + 24$

Achtung: Sie müssen keinen *String*-Parser schreiben, sondern die Baumstruktur mittels Klassen modellieren.

Ihre Lösung:

Diese Seite enthält keine Aufgabenstellung. Sie stellt zusätzlichen Raum für Ihre Lösungen zur Verfügung, falls Ihnen der Platz in den Aufgabenstellungen nicht reicht.

Diese Seite enthält keine Aufgabenstellung. Sie stellt zusätzlichen Raum für Ihre Lösungen zur Verfügung, falls Ihnen der Platz in den Aufgabenstellungen nicht reicht.

Diese Seite enthält keine Aufgabenstellung. Sie stellt zusätzlichen Raum für Ihre Lösungen zur Verfügung, falls Ihnen der Platz in den Aufgabenstellungen nicht reicht.