Name, Vorname | Matrikelnummer

Universität Regensburg

Klausur zur Vorlesung

Einführung in die Programmierung und Programmiersprachen (OOP)

LVNr. 36 603a, SS 2014, im Studiengang Medieninformatik

Donnerstag, 31. Juli 2014 | 13:00 – 14:00 Uhr (60 Minuten)

Allgemeine Hinweise

1. Maximal erreichbare Punktzahl: 60.
2. Schreiben Sie Ihren Namen, Vornamen und Ihre Matrikelnummer leserlich oben auf das Titelblatt bevor Sie mit der Bearbeitung beginnen! Blätter ohne diese Angaben werden nicht gewertet.
3. Verwenden Sie nur die bereitgestellten Klausurbögen. Wenn Sie die Rückseite eines Blattes verwenden, notieren Sie dies bitte auf der Vorderseite. Kennzeichnen Sie eindeutig, zu welcher Aufgabe eine Lösung gehört.
4. Vermerken Sie Besonderheiten deutlich auf Ihrem Klausurbogen (z.B. falls Sie Probleme mit der Anmeldung in FlexNow hatten).
5. Benutzen Sie keine Bleistifte, keine rotschreibenden Stifte und kein TippEx (oder ähnliche Produkte).
6. Es sind **keine technischen Hilfsmittel** erlaubt.
7. Die Klausur ist als „**Closed Book**"-Klausur angelegt. Sie dürfen keine mitgebrachten Quellen oder Notizen zur Bearbeitung der Aufgaben verwenden.
8. Wenden Sie sich bei Unklarheiten in den Aufgabenstellungen immer an die Klausuraufsicht (Hand heben). Hinweise und Hilfestellungen werden dann, falls erforderlich, offiziell für den gesamten Hörsaal durchgegeben. Aussagen unter vier Augen sind ohne Gewähr.
9. Geben Sie keine mehrdeutigen (oder mehrere) Lösungen an. In solchen Fällen wird stets die Lösung mit der geringeren Punktzahl gewertet. Eine richtige und eine falsche Lösung zu einer Aufgabe ergeben also null Punkte.
10. Formulieren Sie Ihre Antworten (ggf. knapp) aus. Wenn die Aufgabenstellung „Nennen Sie..." oder ähnlich lautet, dann reichen auch Stichwörter.

Viel Erfolg!

Statistische Angaben (freiwillig):

An wie vielen Terminen haben Sie die Vorlesung besucht? _____

An wie vielen Terminen haben Sie die Übung besucht? _____

Wie fair finden Sie diese Klausur (Schulnote 1-6)? _____

Name: Matrikelnummer:

Teil 1: Theorie

Achtung: Falsch gesetzte Kreuze bei den Multiple-Choice-Antworten geben Punktabzug. Für jede Aufgabe gibt es mindestens 0 Punkte, d.h. es gibt keine negativen Punktzahlen für einzelne Aufgaben.

1) Datenstrukturen (6 Punkte)

Geben Sie je eine sinnvolle Datenstruktur für folgende Anwendungsfälle an. Es kann mehrere sinnvolle Antworten geben.

a) alle Pixel eines Graustufen-Bildes mit der Auflösung 120 x 120 Pixel

b) durchsuchbare Liste mit Scrabble-Wörtern und ihren zugehörigen Punktzahlen

c) Telefonbuch von München, Einträge jeweils Nachname, Vorname und Telefonnummer

2) Vererbung (1 Punkt)

Was versteht man unter Vererbung in Java?

- ❑ Dass die Basisklasse erst gelöscht werden muss, bevor eine Unterklasse erstellt wird
- ❑ Dass eine Unterklasse Variablen und Methoden der Basisklasse übernimmt
- ❑ Dass mehrere Basisklassen zu einer neuen Klasse kombiniert werden können

Name: Matrikelnummer:

3) Variablen-Initialisierung (2 Punkte)

Wie wird eine Variable 'x' – mit minimaler Sichtbarkeit - korrekt mit der Ganzahl “7” belegt? Kreuzen Sie jeweils das korrekte Element in jeder Spalte an.

❑ `private`	❑ `variable`	❑ `'x'`	❑ `=`	❑ `“7”`	❑ `.`
❑ `personal`	❑ `int`	❑ `X`	❑ `==`	❑ `'7'`	❑ `;`
❑ `public void`	❑ `float`	❑ `x`	❑ `:=`	❑ `7`	❑ `!`
❑ `paired`	❑ `double`	❑ `[x]`	❑ `is`	❑ `this.7`	❑ `_`

Name: Matrikelnummer:

4) Speichernutzung (3 Punkte)

In welchen Speicherbereichen liegen die Elemente x, y und z im folgenden Beispiel? Beziehen Sie sich bei Objekten auf das Objekt selbst, nicht auf die Referenz.

	Heap	Stack	weder noch
`String x = new String("foo");`	❑	❑	❑
`char[] y = {'a', 'b', 'c'};`	❑	❑	❑
`double z = 123.45;`	❑	❑	❑

5) Interfaces (1 Punkt)

Was macht ein Interface in Java?

❑ Es stellt eine standardisierte Schnittstelle zum Betriebssystem zur Verfügung.

❑ Es erlaubt die Behandlung von Fehlern zur Laufzeit.

❑ Es dokumentiert verbindlich zu implementierende Methoden.

Name: Matrikelnummer:

---

6) Vererbung (2 Punkte)

Angenommen, `SubClass` ist eine Unterklasse von `BaseClass`. Wann ist der folgende Code zulässig?

```
...
SubClass sc = new SubClass();
BaseClass bc = sc;
...
```

❑ immer

❑ niemals

❑ nur, wenn SubClass explizit das Interface von BaseClass implementiert

Und wann ist folgender Code zulässig?

```
...
BaseClass bc = new BaseClass();
SubClass sc = (SubClass) bc;
...
```

❑ immer

❑ niemals

❑ nur, wenn SubClass keine neuen Methoden gegenüber BaseClass implementiert

Name: Matrikelnummer:

Teil 2: Code lesen und verstehen

7) Hangman (5 Punkte)

Gegeben sei die folgende Methode:

```
public boolean doStuff() {
      // hangmanWord and placeHolders always have the same length
      for (int i = 0; i < hangmanWord.length(); i++) {
            char currentCharInWord = hangmanWord.charAt(i);
            currentCharInWord = Character.toUpperCase(currentCharInWord);
            char currentCharInPlaceHolder = placeHolders.charAt(i);
            currentCharInPlaceHolder = Character
                        .toUpperCase(currentCharInPlaceHolder);
            if (currentCharInWord != currentCharInPlaceHolder) {
                  return false;
            }
      }
      return true;
}
```

a) Welchen Typ haben hangmanWord und placeHolders vermutlich?

b) Erklären Sie kurz aber präzise in 1-2 Sätzen, was die Methode konkret macht

c) Mit welcher Methode von hangmanWord könnte man einen Großteil der oben gezeigten Methode ersetzen? Nennung genügt.

8) Scope (4 Punkte)

Was gibt folgende Klasse aus, wenn die print()-Methode aufgerufen wird?
Gehen Sie davon aus, dass der Code fehlerfrei ist.

```
class Test {
      public int m;
      public GameObject n = new GameObject();
      public void set( int i, GameObject j ) {
            i = i+1;
            j.x = 42;
      }
      public void print() {
            m = 23;
            n.x = m;
            set(m, n);
            System.out.println(m);
            System.out.println(n.x);
      }
}
```

Ausgabe:

Name: Matrikelnummer:

9) Debugging (6 Punkte)

In folgendem Code-Ausschnitt sind mehrere Syntaxfehler und Bugs zu finden. Markieren Sie alle Fehler und korrigieren Sie diese.

```
public String buggy_method( String input ) {

  if ( input.equals(“Hase“)  {

    return true;

  }

  if ( Input.contains(“Igel“) ) {

    int len = input.length();

    String result = new String;

    for (int i = 0; i <= len; i++) {

      result += Character.toUpperCase(input.charAt(i))

    }

  }

  return “false“;

}
```

Name: Matrikelnummer:

Teil 3: Code schreiben

Verwenden Sie für alle Aufgaben Java, wählen Sie sinnvolle Namen, und achten Sie auf korrekte Sichtbarkeit und Syntax. Korrekte Einrückung ist <u>nicht</u> erforderlich.

10) Kinder, Kinder (10 Punkte)

Eine Klasse „Haustier“ soll die Eigenschaften „Alter“ und „Name“ haben, die über den Konstruktor gesetzt werden. Außerdem verfügt die Klasse über eine Methode „streicheln“. Die Sichtbarkeit aller Attribute und Methoden soll möglichst minimal sein. Die Klasse „Katze“ soll von „Haustier“ erben und beim Streicheln ihren Namen und *„macht miau“* auf der Konsole ausgeben.
Implementieren Sie beide Klassen und erzeugen Sie ein neues Katzen-Objekt, das „Momo“ heißt und 7 Jahre alt ist.

Name: Matrikelnummer:

11) Umgedreht (6 Punkte)

Schreiben Sie eine öffentliche Funktion `reverse`, die einen String als Parameter entgegennimmt und diesen String rückwärts geschrieben zurückgibt.
Beispiel: `reverse(„Hallo!“) → „!ollaH“`

Name: Matrikelnummer:

---

12) Schachaufgabe (14 Punkte)

Schreiben Sie eine Klasse, die ein 8x8 Felder großes Schachbrett repräsentiert. Die Farbe der einzelnen Felder muss nicht repräsentiert werden.
Repräsentieren Sie die verschiedenen Schachfiguren bzw. möglichen Belegungen eines Feldes nach Möglichkeit mit einem `enum` oder notfalls mit Konstanten mit sinnvollem Datentyp.
Zur Vereinfachung kann ein Feld nur eine von drei Belegungen haben: weisser Turm, schwarzer König oder leer.
Gehen Sie davon aus, dass nur ein einziger schwarzer König, 0-2 weiße Türme und keine weiteren Figuren auf dem Schachbrett stehen. Fügen Sie der Klasse eine Methode `public boolean isBlackCheck()` hinzu, welche zurückgibt, ob der schwarze König im Schach steht oder nicht. (Der König steht dann im Schach, wenn sich mindestens ein Turm in der gleichen Reihe oder Spalte wie der König befindet).