

# Technische Universität Braunschweig

Institut für Anwendungssicherheit

## Programmieren I

Prof. Dr. Martin Johns

27. August 2018

Name:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|

Vorname:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|

Matrikelnummer:

|  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|
|  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|

Kennnummer: 1234

Anrede: ☐ Frau ☐ Herr

Studiengang: ☐ Bachelor ☐ Master ☐ Diplom ☐ Frühstudium ☐ Erasmus

Fachrichtung: ☐ Informatik ☐ Wirtschaftsinformatik ☐ Mathematik ☐ Physik

☐ Mobilität und Verkehr ☐ Psychologie ☐ Maschinenbau (☐ Mechatronik)

☐ Elektrotechnik ☐ Wirtschaftsingenieurwesen (☐ Bauing ☐ E-Technik ☐ Maschbau)

☐ Finanz- und Wirtschaftsmathematik ☐ IST ☐ Sonstige: \_\_\_\_\_

Versuch der Notenverbesserung: ☐

Bitte kreuzen Sie an, falls Sie teilgenommen haben: ☐ Programmierlabor ☐ Legolabor.

Die Bearbeitungszeit beträgt 120 Minuten. Die Klausur besteht aus 6 Aufgaben. Sie haben die Klausur bestanden, wenn Sie mindestens 30 von 60 möglichen Punkten erreicht haben.

| Aufgabe     | 1 | 2 | 3  | 4 | 5 | 6  | $\Sigma$ |
|-------------|---|---|----|---|---|----|----------|
| max. Punkte | 5 | 8 | 13 | 8 | 8 | 18 | 60       |
| Punkte      |   |   |    |   |   |    |          |

**Note:**

Die Ergebnisse der Klausur und der Termin für die Klausureinsicht werden im StudIP bekanntgegeben. Bitte prägen Sie sich Ihre Kennnummer gut ein. Aus Datenschutzgründen wird das Klausurergebnis nur unter dieser Kennnummer bekannt gegeben. Aus den gleichen Gründen können Ergebnisse weder telefonisch noch per E-Mail mitgeteilt werden.

**Aufgabe 1:** Bitte kreuzen Sie die wahren Aussagen an. Für jede vollständige richtige Antwort erhalten Sie einen Punkt. Alle Fragen dieser Aufgabe beziehen sich auf Java 10.

|  |  |                          |  |
|--|--|--------------------------|--|
| <i>Deklaration:</i>                      |  |                          |  |
| Diese Deklaration ist zulässig:          | <code>double a = 2.2F;</code>  | <input type="checkbox"/> |  |
| Diese Deklaration ist zulässig:          | <code>double b = 2e2D;</code>  | <input type="checkbox"/> |  |
| Diese Deklaration ist zulässig:          | <code>double d = .2d;</code>   | <input type="checkbox"/> |  |
| <i>Methoden:</i>                         |  |                          |  |
| Diese Signatur der Methode ist zulässig: | <code>static int f (int a, int b)</code>   | <input type="checkbox"/> |  |
| Diese Signatur der Methode ist zulässig: | <code>static int g (int... a, int b)</code>  | <input type="checkbox"/> |  |
| Diese Signatur der Methode ist zulässig: | <code>static int h (int... a)</code>   | <input type="checkbox"/> |  |
| <i>Objektorientierung:</i>               |  |                          |  |
|  | Zwei abgeleitete Klassen können dieselbe Basisklasse besitzen.                                 | <input type="checkbox"/> |  |
|  | Von abstrakten Klassen können Objekte erzeugt werden.  | <input type="checkbox"/> |  |
|  | Nicht jede Klasse kann vererbt werden.   | <input type="checkbox"/> |  |
| <i>Interface:</i>                        |  |                          |  |
|  | Ein Interface muss mindestens eine abstrakte oder konkrete Methode enthalten:                  | <input type="checkbox"/> |  |
|  | Es gibt Möglichkeiten für ein Interface den Modifikator <code>static</code> zu verwenden:      | <input type="checkbox"/> |  |
|  | Ein Interface kann mehrere Interfaces erweitern:   | <input type="checkbox"/> |  |
| <i>Ausnahmebehandlung:</i>               |  |                          |  |
|  | Die <code>catch-or-throw</code> -Regel gilt für <code>RuntimeException</code> nicht:           | <input type="checkbox"/> |  |
|  | <code>int x = -2/0;</code> löst bei der Übersetzung eine <code>ArithmeticException</code> aus: | <input type="checkbox"/> |  |
|  | Eine <code>catch</code> -Klausel kann mehr als einen Parameter bekommen:                       | <input type="checkbox"/> |  |

5 Punkte

**Aufgabe 2:** (*Zahldarstellung*) Schreiben Sie die Hexadezimalzahl  $(5E)_{16}$  als Binär-, Oktal- und Dezimalzahl. Welchen Wert ergibt der Ausdruck `23 - 016 + 0x1f + 0b10` in Java?

a)  $(5E)_{16}$  als Binärzahl: \_\_\_\_\_

b)  $(5E)_{16}$  als Oktalzahl: \_\_\_\_\_

c)  $(5E)_{16}$  als Dezimalzahl: \_\_\_\_\_

d) Der Wert von `23 - 016 + 0x1f + 0b10` : \_\_\_\_\_

8 Punkte

**Aufgabe 3:** (*Kontrollstrukturen, Operatoren, Felder*) Welche der folgenden Schleifen terminieren? Geben Sie im Falle der Terminierung an, welche Werte die Variablen `x` und `y` besitzen, nachdem die jeweiligen Anweisungen ausgeführt wurden. (4 Punkte)

a) 

```
int x = 6;
int y = 2;
for (int i = 0; i < x+y; i *= 2) {
    y = y + x/y;
    if (y == 7) {
        break;
    } else {
        x = x + x%y + 1;
    }
}
```

Die Schleife terminiert: ja ☐ nein ☐ x = \_\_\_\_\_ y = \_\_\_\_\_

b) 

```
int x = 7;
int y = 1;
do {
    y = x + y;
    switch (y % 3) {
        case 1 : y = x - 4;
        case 2 : y = x + 3; break;
    }
    x--;
} while (y < 11);
```

Die Schleife terminiert: ja ☐ nein ☐ x = \_\_\_\_\_ y = \_\_\_\_\_

Geben Sie für jeden der folgenden Ausdrücke den Typ und den Wert des Ausdrucks an. Setzen Sie vor jedem Ausdruck die Deklaration `int x = 13;` voraus. (3 Punkte)

c) `((x&5)|3)` Typ: \_\_\_\_\_ Wert: \_\_\_\_\_

d) `(x>>2)^2` Typ: \_\_\_\_\_ Wert: \_\_\_\_\_

e) `(x/3) != (x%3)` Typ: \_\_\_\_\_ Wert: \_\_\_\_\_

Es sei `a` durch `int[] a = new int[6]` deklariert. Geben Sie den Inhalt des Felds nach Ausführung der beiden folgenden Anweisungen an: (6 Punkte)

f) 

```
for (int i = 0; i <= 10; i = i+2) a[i/2] = i < 5 ? 5 - i : 0;
for (int i = a.length; i > 0; i--) a[i-1] = a[a[i-1]] + i;
```

`a[0]` = \_\_\_\_\_ `a[1]` = \_\_\_\_\_ `a[2]` = \_\_\_\_\_

`a[3]` = \_\_\_\_\_ `a[4]` = \_\_\_\_\_ `a[5]` = \_\_\_\_\_

13 Punkte

**Aufgabe 4:** (*Objektorientierung*) Gegeben sei das folgende Java-Programm:

```
class Person {
    protected String name;
    public Person(String name) {
        _____.name = name;
    }
    public String toString() {
        return name;
    }
}
class Mitarbeiter _____ Person {
    protected _____ int nummer;
    protected _____ int zaehler;
    public Mitarbeiter(String name) {
        _____(name);
        nummer = _____.nummer;
    }
    public String toString() {
        return name + " (Mitarbeiter " + nummer + ")";
    }
}
class Main {
    public static _____ main(String[] args) {
        Person heinz = new Person("Heinz Kunz");
        Mitarbeiter peter = new Mitarbeiter("Peter Gibbons");
        Person roy = new _____("Roy Trenneman");
        System.out.println(heinz);
        System.out.println(peter);
        System.out.println(roy);
    }
}
```

Füllen Sie die Lücken im obigen Programm, so dass es die nachfolgende Ausgabe erzeugt:

```
Heinz Kunz
Peter Gibbons (Mitarbeiter 1)
Roy Trenneman (Mitarbeiter 2)
```

Sie dürfen dabei den vorhandenen Code nicht ändern und auch nicht streichen. Zusätzliche Ausgaben einzubauen ist ebenfalls nicht zulässig. Nicht jede Lücke muss notwendigerweise auch gefüllt werden, falsche oder überflüssige Ergänzungen führen zu Punktabzug.

8 Punkte

**Aufgabe 5:** (*Rekursion*) Gegeben sei die folgende rekursive Methode:

```
static int f(int x) {  
    if (x < 3)  
        return x + 2;  
    else  
        return 2 * f(x-1) + f(x-3);  
}
```

- a) Welchen Wert liefert der Aufruf  $f(6)$ ? In welcher Reihenfolge und mit welchen Parametern wird  $f$  dabei aufgerufen? Geben Sie die Reihenfolge der Aufrufe explizit an. Wie groß ist die maximale Rekursionstiefe, das heißt die maximale Anzahl gleichzeitig aktiver Aufrufe? (*6 Punkte*)
- b) Liefert der Aufruf  $f(x)$  mit  $x \geq 0$  und unter der Annahme dass der Aufruf erfolgreich terminiert immer einen positiven `int`-Wert? Begründen Sie Ihre Antwort. (*2 Punkte*)

8 Punkte

**Lösung:**

**Aufgabe 6:** (*Programmerstellung*) Erstellen Sie ein Programm, welches eine einfache Kompression auf einem String erzeugt, in dem es sich wiederholende Zeichen durch einen Zähler ersetzt.

*Beispiel:* Der String `aaaaabbbbcaaa` soll zu `a5b3c1a3` komprimiert werden.

- a) Erstellen Sie eine Hilfsmethode, welche die Kompression implementiert. Sie können annehmen, dass der per Parameter übergebene String nur Groß- und Kleinbuchstaben (A-Z) enthält. Falls der komprimierte String nicht kürzer als die ursprüngliche Eingabe ist, soll die Methode eine `IllegalArgumentException` werfen. Verwenden Sie aussagekräftige Variablennamen und kommentieren Sie Ihren Code! (*13 Punkte*)
- b) Erstellen Sie eine `main`-Methode, welche den zu komprimierenden String als Kommandozeilen-Argument entgegennimmt und das Ergebnis der Kompression unter Verwendung der Hilfsmethode aus a) ausgibt. Geben Sie eine sinnvolle Fehlermeldung aus, falls kein Argument übergeben wird. Fangen Sie außerdem die Ausnahme Ihrer Hilfsmethode und geben Sie stattdessen ebenfalls eine Fehlermeldung aus. (*5 Punkte*)

**Für beide Teilaufgaben gilt:** Sie erhalten nur dann Punkte, wenn Sie ein Java-Programm schreiben das erkennbar geeignet ist die Problemstellung zu lösen. Sie dürfen (weitere) eigene Hilfsmethoden schreiben und verwenden, aber weder Klassen noch Methoden importieren.

18 Punkte

**Lösung:**

[illegible]



[illegible]