Basisprüfung Herbst 2018
0027 – Einführung in die Programmierung

Departement Informatik
ETH Zürich

24. Januar 2019 – Schriftlicher Teil

Nachname:_____ Vorname:_____

Stud.number:_____

Mit Ihrer Unterschrift bestätigen Sie, dass Sie folgende Hinweise zur Kenntnis genommen haben, die Aufgaben selbständig gelöst haben, Sie Ihre eigene Lösung abgeben, Sie keine Kopie der Prüfung mitnehmen, es keine störenden äusseren Einflüsse gab, und Sie keine gesundheitlichen Probleme hatten, die Ihre Leistungen in dieser Prüfung beeinträchtigten.

Signature:_____

Aufgabe	Wert	Punkte	Aufgabe	Wert	Punkte
1	3		5	12	
2	3		6	10	
3	12		7	8	
4	12				
Zsumme			Total	60	

Allgemeine Informationen

1. Öffnen Sie diese Prüfung erst, wenn die Aufsicht den Beginn der Prüfung bekannt gibt.
2. Schreiben Sie zuerst Ihren Namen auf diese Pruefung, damit Sie es später nicht vergessen.
3. Die Prüfung dauert 1 Stunde (60 Minuten). Falls Sie sich durch irgendjemanden oder irgendetwas gestört fühlen, melden Sie dies sofort der Aufsichtsperson.
4. Die Prüfung hat 18 Seiten. Vergewissern Sie sich dass Ihr Exemplar vollständig ist.
5. In dieser Prüfung gibt es 60 Punkte. Benutzen Sie die Anzahl der Punkte als *Hinweis*, wie Sie Ihre Zeit einteilen können.
6. Lesen Sie die Aufgabenstellungen genau durch.
7. Tragen Sie Ihre Antwort(en) direkt in die Prüfungsbögen ein. Falls Sie mehr Platz brauchen, ist Ihre Antwort wahrscheinlich zu lang. Wenn Sie doch mehr Platz brauchen, so benutzen Sie die Rückseiten.
8. *Benutzen Sie einen Kugelschreiber (blau oder schwarz) oder Füller der nicht ausradiert werden kann. Benutzen Sie* keinen *Bleistift. Bitte schreiben Sie deutlich und leserlich!* Wenn Sie etwas durchstreichen wollen, so machen Sie dies bitte klar und deutlich.
9. Es ist wichtig, dass Ihre Antworten die Aufgaben klar und *unzweideutig* behandeln. Die Klarheit der Antworten beeinflusst Ihre Note.

   Wenn Sie Annahmen (über die in den Aufgaben aufgeführten hinaus) treffen, so geben Sie diese bitte an.
10. Wir sammeln die Prüfung zum Schluss ein. Wichtig: stellen Sie unbedingt selbst sicher, dass Ihre Prüfung von einer Aufsichtsperson eingezogen wird. Verlassen Sie den Raum erst, wenn *alle* Prüfungen eingezogen wurden.

    Stecken Sie keine Prüfung ein und lassen Sie Ihre Prüfung nicht einfach am Platz liegen.

    (Gilt nur für Gruppe 2): Wenn Sie früher abgeben wollen, melden Sie sich bitte lautlos, und wir holen die Prüfung ab. Vorzeitige Abgaben sind nur bis 20 Minuten vor Prüfungsende möglich.
11. Wenn Sie zur Toilette müssen, melden Sie dies einer Aufsichtsperson durch Handzeichen. Es darf zur gleichen Zeit immer nur eine Studentin oder ein Student zur Toilette.
12. Wir beantworten keine inhaltlichen Fragen während der Prüfung. Kommentare zur Aufgabe schreiben Sie bitte auf das Aufgabenblatt.
13. Wenn die Aufsicht die Prüfung beendet schliessen Sie bitte die Prüfung und schreiben nicht mehr in die Prüfung. Weiterarbeiten über die erlaubte Zeit gilt als Täuschungsversuch.

Aufgabe 1 (3 Punkte)

Gegeben sei eine Methode `main` in einer Java Klasse.

```
public static void main(String[] args) {
    /* body */
}
```

Die folgenden Anweisungen sollen als "Body" (Rumpf) anstelle des Kommentars `/* body */` eingefügt werden. Geben Sie für jede Anweisung an, was für eine Ausgabe erzeugt wird. Achten Sie auf die korrekte Formatierung der verschiedenen Typen, also z.B. 7.0 statt 7 für eine reelle Zahl (`double`).

1. `System.out.println(4.0 * (1 / 4 +  8 / 4.0 + 12.0 / 4) );`

   _____

2. `System.out.println(2 * 5 + " = " + 5 + 5 + " = " + 1.0 * 10  );`

   _____

3. `System.out.println( 7 / 4 >= 5  / 3 || 6 % 4 >= 7 % (3 / 2 - 4 % 3 ));`

   _____

Aufgabe 2 (3 Punkte)

Gegeben sei eine Java Methode `count`, die rekursiv die Knoten eines Binärbaumes (Exemplar der Klasse `BTree`) besucht.

```
class BTree {
    BTree right;
    BTree left;
    Object item;

    BTree() {
        right = null;
        left = null;
    }

    BTree(BTree r, BTree l) {
        right = r;
        left = l;
    }
}

public class TreeProblem {
    // ...
    public static int count(BTree node) {
        if (node == null) {
            return 0;
        } else {
            return count(node.right) + count(node.left);
        }
    }
}
```

In der Methode `main` der Klasse `TreeProblem` wird diese Methode mit verschiedenen Argumenten aufgerufen. Bitte geben Sie für jeden dieser Aufrufe das Resultat an (entweder String, der gedruckt wird, oder den Laufzeitfehler (exception)).

1. `System.out.println(count(null));`

   _____

2. `System.out.println(count(new BTree(null, null)));`

   _____

3. `System.out.println(count(new BTree(new BTree(), new BTree())));`

   _____

Aufgabe 3 (12 Punkte)

Geben Sie (für die angegebenen Stellen im Programm) an ob die drei Aussagen `a==0` und `a%10==digit` und `count>0` IMMER, MANCHMAL oder NIE wahr sind. Sie können diese drei Möglichkeiten mit I/M/N abkürzen.

```
public static int assertionRaetsel(int a, int digit) {
    int count = 0;

    // Point A
    while (a != 0) {
        // Point B
        if (a % 10 == digit) {
            count++;
            // Point C
        } else if (count > 0) {
            count--;
            // Point D
        }
        a = a / 10;
    }

    // Point E
    return count;
}
```

	a == 0	a % 10 == digit	count>0
Point A			
Point B			
Point C			
Point D			
Point E			

Aufgabe 4 (12 Punkte)

Gegeben seien diese Klassen:

```
class Rombus extends Parallelogram {
    void info() {
        System.out.println("Romb");
    }
}


class Parallelogram extends Polygon {
   void info() {
      System.out.println("Para");
      special();
   }
   void special() {
      System.out.println("P");
   }
}

class Trapezoid  extends Polygon {
    void info() {
       System.out.println("Trap");
    }
    void special() {
      System.out.println("T");
    }
}

class Rectangle extends Parallelogram  {
    void info() {
       System.out.println("Rect");
       super.info();
    }

    void special() {
       System.out.println("R");
    }

    public String toString() {
       return "Rectangle";
    }
}
```

```
class Square extends Rectangle {
    void info() {
        super.info();
        System.out.println("Square");
    }

    void special() {
        System.out.println("S");
    }

    public String toString() {
        return "Square";
    }
}

class Polygon {
    void info() {
            System.out.println("Poly");
        }

        void special() {
            System.out.println(toString());
        }

        public  String toString() {
            return "Polygon";
        }
}
```

Das folgende Programmsegment ist ein Klient dieser Klassen. Welchen Output produziert dieses Programmsegment?

```
public static void main (String[] args) {
  Polygon [] shapes = { new Rectangle(), new Polygon(), new Parallelogram(),
      new Rombus(), new Trapezoid(), new Square() };

  for (int i = 0; i < shapes.length; i++) {
    System.out.println(shapes[i]);
    shapes[i].info();
    shapes[i].special();
  }
}
```

Antwort:

Aufgabe 5 (12 Punkte)

Gegeben seien diese Klassen:

```
interface I1 {
    public void value();
}

class A1 {
    void setup() {
        System.out.println("Setup A");
    }
}

class B1  extends A1 implements I1 {

    void setup() {
        super.setup();
        System.out.println("Setup B");
    }
    public void value() {
        System.out.println("Value B");
    }
}
```

```
class X1 extends A1 {

    void setup() {
        System.out.println("Setup X");
        value();
    }
    void value() {
        System.out.println("Value X");
    }

}

class Y1 extends X1 {

    void value() {
        System.out.println("Value Y");
    }
    void clear() {
        System.out.println("Clear X");
    }

}
```

In einer Klasse `Explore` im selben Package (d.h., die Methoden aller Klassen sind sichtbar) befindet sich die Methode `main`.

```
public static void main (String[] args) {

  /* Body  */

}
```

Ausserdem ist in der Klasse `Explore` noch die Methode `foo` definiert:

```
    static void foo(I1 ivar) {
        ivar.value();
    }
```

Die folgenden Anweisungen sollen als "Body" (Rumpf) anstelle des Kommentars `/* body */` eingefügt werden. Geben Sie für jede Anweisungsfolge an, was für eine Ausgabe erzeugt wird, ob ein Laufzeitfehler (exception) auftritt, oder der Compiler einen Fehler (error) feststellt.

1.
```
A1 a = new A1();
a.setup();
```

_____

2.
```
B1 b = new B1();
b.setup();
b.value();
```

_____

3.
```
Y1 y = new Y1();
((X1)y).setup();
```

_____

4.
```
Y1 y = new Y1();
((X1)y).value();
```

_____

5.
```
Y1 y = new Y1();
((X1)y).clear();
```

_____

6.
```
B1 b = new B1();
Y1 y = new Y1();
b = y;
b.setup();
```

_____

7.
```
I1 i = new Y1();
i.value();
```

_____

8.
```
Y1 y = new Y1();
A1 a = y;
B1 b = (B1) a;
b.value();
```

_____

9.
```
Y1 y = new Y1();
A1 a = y;
((Y1)a).value();
```

_____

10.
```
Y1 y = new Y1();
A1 a = y;
((I1)a).value();
```

_____

11.
```
X1 x = new Y1();
foo(x);
```

_____

Aufgabe 6 (10 Punkte)

Welche dieser Hoare Tripel sind (un)gültig? Bitte geben Sie für ungültige Tripel ein Gegenbeispiel an. Die Zuweisungen sind Teil einer Java Methode. Alle Variablen sind vom Typ `int` und es gibt keinen Overflow.

1. `{ x >= 0 || y >= 0 }  z = x * y;  { z > 0 }`

2. `{ x <= 0 && y >= 0 && x = y }  z = x * y;  { z <= 0 }`

3. `{x > 10 }  z = x % 10;  { z > 0 }`

4. `{x > 0 }  y = x * x; z = y / 2;  { z > 0 }`

5. `{ true }`

```
if (x > y) {
    y = x;
} else {
   y = - x;
}

{ y >= x }
```

6. `{ b > c  }`

```
if (x > b) {
    a = x;
} else {
    a = b
}

{ a > c }
```

Aufgabe 7 (8 Punkte)

Gegeben sei die EBNF Beschreibung von *expr* (aus Übung 11).

$$
\begin{array}{lcl}
\mathit{digit} & \Leftarrow & \boxed{\texttt{0}} \mid \boxed{\texttt{1}} \mid \ldots \mid \boxed{\texttt{9}} \\
\mathit{char} & \Leftarrow & \boxed{\texttt{A}} \mid \boxed{\texttt{B}} \mid \ldots \mid \boxed{\texttt{Z}} \mid \boxed{\texttt{a}} \mid \boxed{\texttt{b}} \mid \ldots \mid \boxed{\texttt{z}} \\
\mathit{num} & \Leftarrow & \mathit{digit}\ \{\ \mathit{digit}\ \}\ [\ \boxed{\texttt{.}}\ \mathit{digit}\ \{\ \mathit{digit}\ \}\ ] \\
\mathit{var} & \Leftarrow & \mathit{char}\ \{\ \mathit{char}\ \} \\
\mathit{func} & \Leftarrow & \mathit{char}\ \{\ \mathit{char}\ \}\ \boxed{\texttt{(}} \\
\mathit{op} & \Leftarrow & \boxed{\texttt{+}} \mid \boxed{\texttt{-}} \mid \boxed{\texttt{*}} \mid \boxed{\texttt{/}} \mid \boxed{\texttt{\^{}}} \\
\mathit{open} & \Leftarrow & \boxed{\texttt{(}} \\
\mathit{close} & \Leftarrow & \boxed{\texttt{)}} \\
 & & \\
\mathit{atom} & \Leftarrow & \mathit{num} \mid \mathit{var} \\
\mathit{term} & \Leftarrow & \mathit{open}\ \mathit{expr}\ \mathit{close} \mid \mathit{func}\ \mathit{expr}\ \mathit{close} \mid \mathit{atom} \\
\mathit{expr} & \Leftarrow & \mathit{term}\ [\ \mathit{op}\ \mathit{term}\ ]
\end{array}
$$

Abbildung 1: **EBNF-Beschreibung** von *expr*

Geben Sie für jeden Ausdruck an, ob er ein gültiger Ausdruck ist.

1. `sin((1.5 * alpha) + t)`
2. `alpha/2 * PI**2`
3. `cos(t/2) + 0.75`
4. `b * (i++)`
5. `myArray[i]*yourArray[j]`
6. `beta/0.0`
7. `expr`
8. `(7 + beta) * (alpha * 2)`

Wir wünschen Ihnen alles Gute für den Rest der Prüfungssession und das nächste Semester. Ihr "Einführung in die Programmierung"-Team.

Aufgabe	Wert	Punkte	Aufgabe	Wert	Punkte
1	3		5	12	
2	3		6	10	
3	12		7	8	
4	12				
Zsumme			Total	60	