Basispruefung Herbst 2016/17
0027 – Einführung in die Programmierung

Departement Informatik
ETH Zürich

Nachname:_____ Vorname:_____ Stud.number:_____

Mit Ihrer Unterschrift bestätigen Sie, dass Sie folgenden 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	6		7	8	
2	15		8	8	
3	10		9	15	
4	8		10	10	
5	15		11	12	
6	5		12	8	
Zsumme			Total	120	

ALLGEMEINE INFORMATIONEN

1. Öffnen Sie diese Prüfung erst, wenn die Aufsicht den Beginn der Prüfung bekannt gibt.
2. Schreiben Sie zuerst Ihren Namen auf dieses Blatt, damit Sie es später nicht vergessen.
3. Die Prüfung dauert 2 Stunden (120 Minuten). Falls Sie sich durch irgendjemanden oder irgendetwas gestört fühlen, melden Sie dies sofort der Aufsichtsperson.
4. Die Prüfung hat 14 Seiten. Vergewissern Sie sich dass Ihr Examplar vollständig ist.
5. In dieser Prüfung gibt es 120 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. Stecken Sie keine Prüfung ein und lassen Sie Ihre Prüfung nicht einfach am Platz liegen. Dasselbe gilt, wenn Sie früher abgeben wollen: bitte melden Sie sich lautlos, und wir holen die Prüfung ab. Vorzeitige Abgaben sind nur bis 30 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 ueber die erlaubte Zeit gilt als Täuschungsversuch.

Aufgabe 1 (6 Punkte)

Gegeben sei eine Java Klasse:

```
class Test {
  public void foo (int x, int y) { }
}
```

Die folgende Liste von Methoden soll zu der obigen Klasse hinzugefügt werden. Entscheiden Sie für jede Methode einzeln (d.h. jede Methode für sich), ob das Hinzufügen dieser Methode zulässig ist oder ob es eine Fehlermeldung geben würde.

```
public void bar (int x, int y) { }                // zulaessig: ja/nein
public int  foo (int x, int y) { return 0; }      // zulaessig: ja/nein
public int  foo (int a, int b) { return 0; }      // zulaessig: ja/nein
public int  foo (float f, float g) { return 0; } // zulaessig: ja/nein
public void foo (float f, float g) { }            // zulaessig: ja/nein
int  foo (int x, int y) { return 0; }             // zulaessig: ja/nein
void foo (int x, int y) { }                       // zulaessig: ja/nein
private int  foo (int x, int y) { return 0; }     // zulaessig: ja/nein
private void foo (int x, int y) { }               // zulaessig: ja/nein
public void Test() {}                             // zulaessig: ja/nein
public int Test() { return 0; }                   // zulaessig: ja/nein
public Test Test() { return null; }               // zulaessig: ja/nein
```

Aufgabe 2 (15 Punkte)

Gegeben sei eine Methode `main` in einer Java Klasse mit verschiedenen Deklarationen von Variablen in der Methode.

```
  public static void main(String[] args) {
    int i, j, k, l;
    double e, f, g;

 /* body */
}
```

Die folgenden Anweisungen sollen als "Body" (Rumpf) anstelle des Kommentars `/* body */` eingefuegt werden. Geben Sie fuer jede Gruppe von Anweisungen an, was fuer eine Ausgabe erzeugt wird.

1. ```
   j = 3 + 12 % 7 / 5 + 4 % 2;
   System.out.println(j);
   ```
2. ```
   i = 3  / 4 * 7 % 4 + 1 * 2 / 5;
   System.out.println(i);
   ```
3. ```
   k = 48 + 12 / (8 % 4) * 2;
   ```
4. ```
   l = (7 + 14 / 5) % 11;
   System.out.println(l);
   ```
5. ```
   e = 5 / (30 / 15.0) * 6 / 10;
   System.out.println(e);
   ```
6. ```
   f = 108 / 20 * 3 / 4 / 2.0 + 1.0 / 2;
   System.out.println(f);
   ```
7. ```
   g = 3.0 / (2 / 3) + 14 % 10;
   System.out.println(g);
   ```
8. ```
   System.out.println(3 * 2 + 4 + ``+'' + 2 + 3 * 4);
   ```

Aufgabe 3 (10 Punkte)

Schreiben Sie eine Methode nurUngeradeZiffern die zurueck gibt ob alle Ziffern einer positiven ganzen Zahl ungerade sind. Ihre Methode sollte den Wert true zurueck geben wenn die Zahl nur aus ungeraden Ziffern besteht und false wenn mindestens eine der Ziffern gerade ist. 0, 2, 4, 6, 8 sind gerade Ziffern, 1, 3, 5, 7, 9 sind ungerade Ziffern.

Zwei Beispiele: nurUngeradeZiffern(9593117) ergibt true, nurUngeradeZiffern(39105389) ergibt false.

Aufgabe 4 (8 Punkte)

Schreiben Sie eine Methode ersteZiffer, die die erste Ziffer einer ganzen Zahl zurueck gibt. ersteZiffer(9320) sollte 9 zurueck geben. Die Methode sollte auch fuer negative Zahlen funktionieren, zum Beispiel sollte ersteZiffer(-318) 3 und ersteZiffer(0) 0 ergeben.

Aufgabe 5 (15 Punkte)

Schreiben Sie eine Methode goodTriple die drei ganze Zahlen als Parameter akzeptiert und den Wert true zurueck gibt wenn eine der Zahlen genau in der Mitte der anderen beiden Zahlen liegt. Z.B. sollte goodTriple(5, 7, 3) true ergeben weil einer der Parameter (5) genau in der Mitte zwischen 7 und 3 liegt. Hingegen sollte goodTriple(4, 6, 10) false ergeben.

Aufgabe 6 (5 Punkte)

Fuer jede der folgenden EBNF Beschreibungen edesc, geben Sie ein Beispiel an, das legal ist, sowie ein Beispiel, das nicht legal ist. (<= ist das Symbol das "ist definiert als" bedeutet).

1. `edesc <= [A] [B]`
2. `edesc <= [A | B]`
3. `edesc <= A | [B { C }]`
4. `edesc <= { A | B } { B | C }`
5. `one <= { A } [ B | { C }]`
   `edesc <= one D one`

Aufgabe 7 (8 Punkte)

Schreiben Sie eine EBNF Beschreibung fuer aufzaehlung. Eine aufzaehlung besteht aus einem oder mehreren X die nach den Regeln der Kommatrennung aneinander gereiht sind. Genauer: eine aufzaehlung besteht entweder

- aus einem X
- aus zwei X die durch das Wort `und` getrennt sind
- aus $n >= 3$ X von denen die ersten $n - 1$ X durch ein Komma (`,`) getrennt sind und das Wort `und` die letzten beiden trennt.

Legal Beispiele fuer aufzaehlung sind

- X
- X `und` X
- X, X, X, X `und` X

Illegale Beispiele fuer aufzaehlung sind

- X, X
- `und` X
- X, X, X `und` X `und` X
- X `und` X, X, X, X

Aufgabe 8 (8 Punkte)

Gegeben sei eine Methode `istVokal` die prueft ob ein gegebener String ein Vokal ist.

```
// Gibt zurueck, ob ein gegebener String ein Vokal ist:
// a, e, i, o oder u, inkl. Grossbuchstaben

public static boolean isVowel(String s) {
    if (s == ``a'') {
        return true;
    } else if (s == ``e'') {
        return true;
    } else if (s == ``i'') {
        return true;
    } else if (s == ``o'') {
        return true;
    } else if (s == ``u'') {
        return true;
    } else {
        return false;
    }
}
```

Wie koennen Sie diese Methode verbessern bzw korrigieren? Geben Sie den Code an.

Aufgabe 9 (15 Punkte)

Gegeben sei eine Methode `sonderbar`.

```
public static void sonderbar(int n) {
    int x = 1;
    int y = 2;
    while (y < n) {
        if (n % y == 0) {
            n = n / y;
            x++;
        } else {
            y++;
        }
    }
    System.out.println(x + `` `` + n);
}
```

Geben Sie fuer jeden der Aufrufe an, welchen Output diese Methode produziert.

1. `sonderbar(2);`
2. `sonderbar(5);`
3. `sonderbar(24);`
4. `sonderbar(28);`

Aufgabe 10 (10 Punkte)

Gegeben sei die Methode arrayMystery.

```
public static void arrayMystery(int[] a) {
    for (int i = 1; i < a.length - 1; i++) {
        a[i] = a[i - 1] - a[i] + a[i + 1];
    }
}
```

Geben Sie Fuer jeden der folgenden Arrays an, welche Werte die Elemente dieses Arrays haben, nachdem der Array dieser Methode als Parameter uebergeben wurde.

1. `{42, 42}`
2. `{6, 2, 4}`
3. `{7, 7, 3, 8, 2}`
4. `{4, 2, 3, 1, 2, 5}`
5. `{6, 0, -1, 3, 5, 0, -3}`

Aufgabe 11 (12 Punkte)

Gegeben sei die Klasse BasicPoint.

```
public class BasicPoint {
    int x;
    int y;

    public BasicPoint(int initialX, int initialY) {
        x = initialX;
        y = initialY;
    }

}
```

Das folgende Programm gibt 5 Zeilen aus. Geben Sie jede Zeile der Ausgabe an wie sie auf der Konsole erscheinen wuerde.

```
public class ReferenceMystery {

    public static void main(String[] args) {
        BasicPoint p = new BasicPoint(11, 22);
        int[] a = {33, 44};
        int n = 55;

        System.out.println(p.x + ``,'' + p.y + `` `` + Arrays.toString(a) + `` `` + n);
 // OUTPUT:

        mystery(p, a, n);
 // OUTPUT:

        System.out.println(p.x + ``,'' + p.y + `` `` + Arrays.toString(a) + `` `` + n);
 // OUTPUT:

        a[0] = a[1];
        p.x = p.y;
        mystery(p, a, n);
 // OUTPUT:

        System.out.println(p.x + ``,'' + p.y + `` `` + Arrays.toString(a) + `` `` + n);
 // OUTPUT:

    }


    public static int mystery(BasicPoint p, int[] a, int n) {
        n = 0;
        a[0] = a[0] + 11;
        a[1] = 77;
        p.x = p.x + 33;
        System.out.println(p.x + ``,'' + p.y + `` `` + Arrays.toString(a) + `` `` + n);
        return n;
    }

}
```

Aufgabe 12 (8 Punkte)

Gegeben seien diese Klassen:

```
public class A extends B {
    public void method2() {
        System.out.println(``a 2'');
    }
}

public class D extends B {
    public void method1() {
        System.out.println(``d 1'');
    }
}

public class C {
    public String toString() {
        return ``c'';
    }

    public void method1() {
        System.out.println(``c 1'');
    }

    public void method2() {
        System.out.println(``c 2'');
    }
}

public class B extends C {
    public String toString() {
        return ``b'';
    }

    public void method2() {
        System.out.println(``b 2'');
    }
}
```

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

```
C[] elements = {new A(),
                new B(),
                new C(),
                new D()};

for (int i = 0; i < elements.length; i++) {

    System.out.println(elements[i]);

    elements[i].method1();

    elements[i].method2();
}
```

Wir wünschen Ihnen alles Gute für den Rest der Pruefungssession und das naechste Semester. Ihr "Einführung in das Programmieren"–Team.