HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Heinrich-Heine-Universität Düsseldorf
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik, Abteilung Betriebssysteme
Prof. Dr. Michael Schöttner
Kevin Beineke
Ludmila Himmelspach
Stefan Nothaas

9. Übungsblatt „Programmierung“ im Wintersemester 2016/17

Abgabe bis 19. Dezember 2016, 07:59 Uhr über das Abgabesystem

Aufgabe 1: Rekursion — 5 Punkte

Schreiben Sie eine Klasse `PotRec`, die eine rekursive Methode zur Berechnung von x^y (x, y seien natuerliche Zahlen) implementiert. Testen Sie diese Methode in `main` für unterschiedliche Parameter.

Aufgabe 2: Fehlersuche - Teil 2 — 15 Punkte

Finden Sie alle Fehler in den nachfolgenden Funktionen. Gehen Sie davon aus, dass die Code-Stücke alleine in einer sonst leeren Klasse eingebettet sind. Beschreiben Sie kurz und genau was den Fehler verursacht und wie dieser zu beheben ist.

a) Umwandeln: $0, 1 \rightarrow A$; $2 \rightarrow B$; $3, 4, 5 \rightarrow C$; $6 \rightarrow D$; $7, 8 \rightarrow E$; $9 \rightarrow F$; $Ansonsten \rightarrow G$

```
public static void transform(int div) {
  switch (div) {
    case 0:
    case 1:
      System.out.print("A");
    case 2:
      System.out.print("B");
    case 3:
    case 4:
    case 5:
      System.out.print("C");
    case 6:
      System.out.print("D");
    case 7:
    case 8:
      System.out.print("E");
    case 9:
      System.out.print("F");
    default:
      System.out.print("G");
  }
}
```

b) Berechnet die Fakultät von n

```
public static int faculty(int n) {
```

```
  if (n < 1) {
    return -1;
  } else {
    long fac = 1;
    while (n > 0) {
      fac = fac * n;
      n--;
    }
    return fac;
  }
}
```

c) Gibt die Zahlen von 1 bis 10 auf der Konsole aus

```
public static void printNumbers() {
  int i = 0;
  for (int i = 0; i < 10; i++) {
    System.out.println(i + 1);
  }
}
```

d) Überprüfe ob eine Zahl zwischen [-10, 0) (Ausgabe -1), (0, 10] (Ausgabe 1) oder 0 (Ausgabe 0) liegt. Falls die Zahl in keinem der Intervalle liegt, so gibt die Methode -1 zurück.

```
public int fits(short v) {
  if (v <= -10 && v < 0) {
    return -1;
  } else if (v > 0 && v <= 10) {
    return 1;
  } else {
    return 0;
  }
  return -1;
}
```

Hinweise:

- *Gesucht sind semantische Fehler (keine syntaktischen)*
- *Versuchen Sie die Fehler ohne Testen auf einem Computer zu finden*

Aufgabe 3: Reguläre Ausdrücke — 20 Punkte

a) Schreiben Sie eine Klasse `IPCheck` mit einer Methode `check`, die eine IP-Adresse (IPv4) entgegen nimmt und mithilfe eines regulären Ausdrucks überprüft, ob es sich um eine gültige IP-Adresse für ein privates B-Netz handelt. Das Methode soll `true` zurückgeben, wenn die IP-Adresse gültig ist, andernfalls soll `false` zurückgegeben werden.

b) Testen Sie Ihre Methode, indem Sie einen Unit-Test schreiben, der eine Menge (z.B. 1.000.000) an zufällig gültigen, sowie ungültige Adressen erzeugt und überprüft. Außerdem soll der Unit-Test explizit die Grenzfälle überprüfen.

Hinweise:

- *Informationen zum Aufbau einer IP-Adresse finden Sie unter https://de.wikipedia.org/wiki/IPv4.*
- *Der Adressbereich eines privaten B-Netzes ist 172.16.0.0–172.31.255.255.*
- *Neben den Bausteinen für reguläre Ausdrücke, die in der Vorlesung vorgestellt wurden, gibt es noch weitere. Im Internet unter http://openbook.galileocomputing.de/javainsel9/javainsel_04_007.htm werden alle gängigen Bausteine beschrieben.*

Aufgabe 4: Fehlersuche Rekursion — 5 Punkte

Finden Sie die Fehler in den nachfolgenden Funktionen. Gehen Sie davon aus, dass die Code-Stücke alleine in einer sonst leeren Klasse eingebettet sind. Erklären Sie kurz die Ursache des Fehlers und wie dieser zu beheben ist.

a) Berechnet die Fakultät für den gegebenen Parameter n.

```
public static long fac(int n) {
  return n * fac(n -1);
}
```

b) Berechnet die Fibonaccizahl für den gegebenen Parameter n.

```
public static long fib(int n) {
  if (n < = 2) {
    return 1;
  }
  return fib(n--) + fib(n - 2);
}
```

Aufgabe 5: Programmstrukturen im Hauptspeicher — 15 Punkte

Gegeben sei das folgende Java-Programm.

```
public class Programmstrukturen {
  public static int f(int n, int m) {
    if (0 == n) {
      return m + 1;
    }
    if (0 == m) {
       return n + 1;
    }
    return m + n;
  }

  public static void main(String[] args) {
    if (0 < args.length) {
      int v = Integer.parseInt(args[0]);
      v = f(v, Integer.parseInt(args[1]));
      System.out.println(v);
    }
  }
}
```

a) Untersuchen Sie für alle Variablen und Funktionsargumente, die in dem Programm vorkommen, ob sie auf dem Heap oder auf dem Stack abgelegt werden.

b) Skizzieren Sie den Zustand des Stacks nach dem Eintritt in die Funktion `f`.