Semestralklausur zur Vorlesung

Allgemeine Informatik I

Prof. J. Fürnkranz
Technische Universität Darmstadt — Wintersemester 2004/05
Termin: 14. 2. 2005

Name: **Vorname:** **Matrikelnummer:**

Fachrichtung:

Punkte: (1) .. (2) .. (3) .. (4) .. (5) .. (6) .. (7) .. **Summe:**

Aufgabe 1 (8 Punkte)

In der RGB Farb-Kodierung werden 256 verschiedene Farbtöne pro Farbe verwendet. Sie wollen eine Farbe mit einem Rotanteil von 187 kodieren.

1–a Wie viele Bytes benötigen Sie zur Kodierung des roten Farbanteils?

1–b Geben Sie die Binärdarstellung für den gewünschten Rotanteil an.

1–c Geben Sie die Hexadezimaldarstellung für den gewünschten Rotanteil an.

Aufgabe 2 (14 Punkte)

Für die Unix-Anweisung `ls -l ~tom/texte/../mails/mbox` wird die folgende Ausgabe erzeugt:

```
-rwxr-xr-- 1 tom users 12822 Sep 29 18:45 mbox
```

2–a Erklären Sie die einzelnen Komponenten der Ausgabe. Gehen Sie insbesondere auf die genaue Bedeutung der einzelnen Zeichen des Teilstrings `-rwxr-xr--` ein. Die `1` vor `tom` müssen Sie nicht erklären.

2–b Erklären Sie kurz die Pfad-Angabe `~tom/texte/../mails/mbox` und skizzieren Sie die zu Grunde liegende Verzeichnis-Struktur.

Aufgabe 3 (8 Punkte)

In der Vorlesung haben Sie die logischen Operationen $\wedge$ (und), $\vee$ (oder), $\leftrightarrow$ (exklusives oder) und $\neg$ (Negation) kennen gelernt.

3–a Geben Sie für den logischen Ausdruck $a \vee (b \wedge (\neg a \leftrightarrow \neg b))$ die Wahrheitstafel an. Geben Sie hierbei auch für jeden Teilausdruck den Wahrheitswert an.

a	b	$(\neg a \leftrightarrow \neg b)$	$(b \wedge (\neg a \leftrightarrow \neg b))$	$a \vee (b \wedge (\neg a \leftrightarrow \neg b))$
0	0			
0	1			
1	0			
1	1			

3–b Geben Sie einen einfacheren logischen Ausdruck an, der die Ergebnis-Spalte der Wahrheitstabelle beschreibt.

Aufgabe 4 (15 Punkte)

Betrachten Sie folgendes KarelJ Programm.

```
1     class Karel extend Robot
2     {
3       Int steps = 0;
4
5       void move()
6       {
7         super.move(),
8         steps = steps + 1;
9       }
10
11      void getSteps()
12      {
13        return steps;
14      }
15
16      void move_n(int n)
17      {
18        while(n - steps)
19        karel1.move();
20      }
21
22      task
23      {
24        Karel karel1 = new Karel(1,0,0, West);
25        Robot karel2 = new Karel(1,0,0, East);
26
27        karel2.move();
28        karel1,move();
29
30        karel1.move_n();
31        karel2.move_n(100);
32      }
33
```

4–a Das Programm enthält 10 Fehler. Finden und verbessern Sie die Fehler. Geben Sie hierzu für jeden Fehler die Zeilennummer an, begründen Sie den Fehler, und geben Sie an, wie der Fehler zu korrigieren ist.

4–b Nehmen Sie an, daß die Deklarationen von `karel1` und `karel2` (Zeilen 24 und 25) korrekt seien. Was ist der dynamische und was ist der statische Typ der beiden Variablen? Erklären Sie kurz den Unterschied.

Aufgabe 5 (15 Punkte)

Implementieren Sie die folgende Methode

```
1       int[] reverse(int[] a)
2       {
3         /* ... */
4       }
5
```

Die Methode soll für den Array `a` einen neuen Array erzeugen, der die gleichen Elemente wie Array `a` enthält aber in umgekehrter Reihenfolge. Dieser Array soll als Ergebnis der Methode zurückgegeben werden. Beispielsweise soll die Methode für den Array

1	2	3	4	5

den Array

5	4	3	2	1

als Resultat liefern

Aufgabe 6 (25 Punkte)

Programmieren Sie eine Klasse von Robotern namens `PathFollower`, die in der Lage ist, einem von Beepern ausgelegten Pfad zu folgen. Im Detail sollen folgende Komponenten erstellt werden:

6–a Eine Klassendefinition, die alle folgenden Methoden enthält.

6–b die Methode `move` soll so modifiziert werden, daß nach jedem Schritt vorwärts, ein auf dem Zielfeld befindlicher Beeper aufgehoben wird. Wenn sich kein Beeper auf dem Zielfeld befindet, soll (natürlich) kein Beeper aufgehoben werden.

Sie können annehmen, daß sich nicht mehr als ein Beeper auf einem Feld befindet.

6–c eine Methode `isBeeperInFront`, die feststellt, ob unmittelbar vor dem Roboter ein Beeper liegt. Die Methode soll `true` retournieren, wenn ein Beeper vor dem Roboter liegt, und `false`, wenn dort kein Beeper liegt.

Sie dürfen annehmen, daß sich keine Mauern oder sonstige Hindernisse vor dem Roboter befinden.

6–d eine Methode `turnToBeeper`, die den Roboter so lange dreht, bis er vor einem Beeper steht. Falls sich kein Beeper auf den vier benachbarten Feldern befindet, soll der Roboter am Ende wieder in die ursprüngliche Richtung blicken.

6–e eine Methode `followPath`, die den Roboter so lange einem Pfad von Beepern folgen und alle dabei gefundenen Beeper aufheben läßt, bis sich kein Beeper mehr in der Nachbarschaft findet.

Hinweis: *Sie dürfen zur Definition jeder Teil-Methode die jeweils davor liegenden Methoden verwenden (egal, ob Sie deren Implementierung geschafft haben oder nicht).*

Aufgabe 7 (15 Punkte)

Schreiben Sie (wie in den Übungen) die Methode `paint` für ein Java-Applet, das die Gitterlinien der Roboter-Welt zeichnet.

Rufen Sie sich aus den Übungen in Erinnerung, daß Sie die Aufrufe

```
int b = g.getClipBounds().width;
int h = g.getClipBounds().height;
```

verwenden können, um die Breite und Höhe des Fensters zu berechnen, in dem das Java-Applet läuft (unter der Annahme, daß `g` der Name eines `Graphics`-Objektes ist).

Die Methode `paint` soll im Abstand von 10 Punkten vertikale bzw. horizontale Gitterlinien zeichnen. Das Applet soll für beliebig große Fenstergrößen funktionieren.

Die erste vertikale Gitterlinie soll an den Koordinaten (0,0) beginnen und an (0,h) enden, die zweite vertikale Linie soll bei (10,0) beginnen und bei (10,h) enden, die dritte bei (20,0) beginnen usw. Die letzte vertikale Linie beginnt beim Punkt (x,0), wobei x das größte Vielfache von 10 ist, das kleiner b ist.

Die horizontalen Gitterlinien sollen analog dazu bei den Koordinaten (0,0), (0,10), (0,20), ... beginnen und bei (b,0), (b,10), (b,20), ... enden. Die letzte horizontale Linie beginnt beim Punkt (0,y), wobei y das letzte Vielfache von 10 ist, das kleiner h ist.

Lösung 1

1–a 1 Byte = 8 bit

1–b 10111011_2

1–c BB_{16} ($1011_2 = B_{16}$)

Lösung 2

2–a
- Das erste Minus besagt: normales File.
- `tom` ist der Besitzer der Datei.
- `users` ist die besitzende Gruppe der Datei.
- `12822` Grösse der Datei.
- `Sep 29 18:45` der Zeitstempel der letzen Modifikation.
- `mbox` ist der Dateiname.
- `rwxr-xr--`:
  - `tom` hat Lese-, Schreib- und Ausführrechte auf das File `mbox`
  - Alle Mitglieder der Gruppe `users` haben Lese- und Ausführrechte auf dieses File.
  - alle anderen Benutzer haben nur Leserechte.

2–b `~tom` ist der Name des Homeverzeichnisses des Benutzers `tom`.
Verzeichnisstruktur:

```
-- ~tom --+-- texte
          |
          +-- mails --+-- mbox
```

Lösung 3

3–a

a	b	$(\neg a \leftrightarrow \neg b)$	$(b \wedge (\neg a \leftrightarrow b))$	$a \vee (b \wedge (\neg a \leftrightarrow b))$
0	0	0	0	0
0	1	1	1	1
1	0	1	0	1
1	1	0	0	1

3–b Der vereinfachte Ausdruck lautet: $a \vee b$.

Lösung 4

4–a
- Zeile 1: `extend` ist falsch geschrieben. Statt `extend` muss `extends` stehen.
- Zeile 2: Gross- und Kleinschreibung: Statt `Int` muss `int` stehen.
- Zeile 7: Das abschliessende Komma muss durch ein Semikolon ersetzt werden.
- Zeile 11: Der Rückgabewert muss von `void` nach `int` geändert werden, da eine `int` von der Methode zurückgegeben wird.
  Korrekt ist auch der Korrektur-Vorschlag, keinen Wert zurückzugeben, also `return steps` durch `return` zu ersetzen.
- Zeile 18: Statt `while` muss `loop` stehen, da ein `int`-Wert in den Klammern steht.
  Korrekt ist auch der Lösungsvorschlag, einen Bool'schen Wert in die Klammer zu schreiben (also z.B. `n - steps > 0`).
- Zeile 19: `karel1` ist ein unbekannter Bezeichner an dieser Stelle. `karel1.` kann komplett entfernt werden oder durch `super.` ersetzt werden.
- Zeile 21: Die schliessende Klammer für die Klassendefinition fehlt.
- Zeile 28: Das Komma muss durch einen Punkt ersetzt werden.
- Zeile 30: Es fehlt der Parameter für die Methode. Statt `()` muss z.B. `(100)` stehen.

- Zeile 31: `karel2` hat den statischen Typ `Robot`. Für die Klasse existiert keine Methode `move_n`.

4–b statischer Typ von `karel1` : `Karel`

dynamischer Typ von `karel1` : `Karel`

statischer Typ von `karel2` : `Robot`

dynamischer Typ von `karel2` : `Karel`

- Statischer Typ:
  - der Typ mit dem die Variable deklariert wird.
  - dieser Typ bestimmt, welche Werte der Variablen zugewiesen werden können und welche Methoden verwendet werden können (nur die, die für den statischen Typ deklariert wurden).
  - eine Variable kann im Gültigkeitsbereich ihrer Deklaration immer nur einen Statischen Typ haben
- Dynamischer Typ:
  - der Typ des Objekts, das der Variablen zugewiesen wird
  - muss ein Untertyp des Statischen Typs sein und bei Aufruf einer Methode wird die Methode des Dynamischen Typs verwendet.
  - eine Variable kann im Gültigkeitsbereich ihrer Deklaration auch mehrere dynamische Typen haben.

Lösung 5

```
1       public int[] reverse(int[] a)
2       {
3         int[] b = new int[a.length];
4
5         for (int i = 0; i < a.length; ++i)
6           b[i] = a[a.length - i - 1];
7
8         return b;
9       }
10
```

Lösung 6

```
1       class Karel extends Robot
2       {
3         void move()
4         {
5           super.move();
6
7           if (nextToABeeper())
8             pickBeeper();
9         }
10
11        boolean isBeeperInFront()
12        {
13          super.move();
14
15          boolean value = nextToABeeper();
16
17          loop (2)
18            turnLeft();
```

```
19
20          super.move();
21
22          loop (2)
23            turnLeft();
24
25          return value;
26        }
27
28        // Variante 1:
29
30        void turnToBeeper()
31        {
32          loop(4)
33          {
34            if (!isBeeperInFront())
35              turnLeft();
36            else          // ohne else-Zweig auch richtig aber ineffizienter
37              return;     // oder: break;
38          }
39        }
40
41        // Variante 2:
42
43        void turnToBeeper()
44        {
45          int i = 0;
46
47          while (!isBeeperInFront() && i < 4)
48          {
49            ++i;
50            turnLeft();
51          }
52        }
53
54        void followPath()
55        {
56          turnToBeeper();
57
58          while (isBeeperInFront())
59          {
60            move();
61            turnToBeeper();
62          }
63        }
64
65      }
66
```

Lösung 7

```
1
2       public void paint(Graphics g)
3       {
4         int b = g.getClipBounds().width;
5         int h = g.getClipBounds().height;
```

```
6
7         for (int x = 0; x <= w; x += 10)
8           g.drawLine(x, 0, x, h);
9
10        for (int y = 0; y <= h; y += 10)
11          g.drawLine(0, y, b, y);
12      }
13
14
15      // Alternative Loesung
16
17      public void paint(Graphics g)
18      {
19        int b = g.getClipBounds().width;
20        int h = g.getClipBounds().height;
21
22        int x = 0;
23        int y = 0;
24
25        while (x <= w)
26        {
27          g.drawLine(x, 0, x, h);
28          x = x + 10;
29        }
30
31        while (y <= h)
32        {
33          g.drawLine(0, y, b, y);
34          y = y + 10;
35        }
36
37      }
38
```