Klausur zur Vorlesung

Allgemeine Informatik I

Prof. J. Fürnkranz
Technische Universität Darmstadt — Wintersemester 2006/07
Termin: 6. 9. 2007

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

Fachrichtung: **Wiederholer:** ☐ ja ☐ nein

☐ Diplom ☐ Master ☐ Bachelor

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

- **Aufgaben:** Diese Klausur enthält auf den folgenden Seiten 6 Aufgaben zu insgesamt 80 Punkten. Jede Aufgabe steht auf einem eigenen Blatt. Kontrollieren Sie *sofort*, ob Sie alle sechs Blätter erhalten haben!
- **Zeiteinteilung:** Die Zeit ist knapp bemessen. Wir empfehlen Ihnen, sich zuerst einen kurzen Überblick über die Aufgabenstellungen zu verschaffen, und dann mit den Aufgaben zu beginnen, die Ihnen am besten liegen.
- **Papier:** Verwenden Sie nur Papier, das Sie von uns ausgeteilt bekommen. Bitte lösen Sie die Aufgaben auf den dafür vorgesehenen Seiten (auch auf den Rückseiten). Falls der Platz nicht ausreicht, vermerken Sie dies bitte und setzen die Lösung auf einem Zusatzblatt fort. Brauchen Sie zusätzlich Papier (auch Schmierpaper), bitte melden.
- **Fragen:** Sollten Sie Teile der Aufgabenstellung nicht verstehen, bitte fragen Sie!
- **Abschreiben:** Sollte es sich (wie in den letzten Jahren leider immer wieder) herausstellen, daß Ihre Lösung und die eines Kommilitonen über das zu erwartende Maß hinaus übereinstimmen, werden beide Arbeiten negativ beurteilt (ganz egal wer von wem in welchem Umfang abgeschrieben hat).
- **Ausweis:** Legen Sie Ihren *Studentenausweis* und *Lichtbildausweis* sichtbar auf Ihren Platz. Füllen Sie das Deckblatt sofort aus!
- **Hilfsmittel:** Zur Lösung der Aufgaben sind keine Unterlagen erlaubt. Gedruckte Wörterbücher sind für ausländische Studenten erlaubt, elektronische Hilfsmittel (Taschenrechner, elektronische Wörterbücher, Handy, etc.) sind verboten! Sollten Sie etwas anderes verwenden wollen, bitte klären Sie das *bevor* Sie zu arbeiten beginnen.
- **Aufräumen:** Sonst darf außer Schreibgerät, Essbarem, von uns ausgeteiltem Papier und eventuell Wörterbüchern nichts auf Ihrem Platz liegen. Taschen bitte unter den Tisch!

Gutes Gelingen!

Aufgabe 1 (8 Punkte)

1–a Geben Sie die Auswertungstabelle eines Halbaddierwerks an, d.h. die Ausgabe für jede mögliche Kombination von Eingabewerten.

a	b	
0	0	
0	1	
1	0	
1	1	

1–b Geben Sie logische Ausdrücke (bestehend aus $\wedge$, $\vee$, und der Negation) an, mit denen das Halbaddierwerk realisiert werden kann.

1–c Erklären Sie kurz die wesentliche funktionelle Erweiterung eines Volladdierwerks im Vergleich zum Halbaddierwerk. Wofür wird das Volladdierwerk eingesetzt?

Aufgabe 2 (8 Punkte)

Beantworten Sie die folgenden Fragen möglichst kurz und prägnant (1-2 Sätze pro Teilaufgabe).

2–a Wie viele Bytes benötigen Sie zur Kodierung eines Unicode-Zeichens?

2–b Welche Funktionalität stellt der Network Layer im TCP/IP 5-Schichtenmodell zur Verfügung?

2–c Mit welchem Verfahren versucht TCP Fehler in der Datenübertragung zu erkennen?

2–d Was ist in einem Betriebssystem ein Prozeß?

Aufgabe 3 (12 Punkte)

Gegeben seien folgende Klassendefinitionen mit anschliessendem Task:

```
1   class X extends Robot {
2      int adda() {
3         return 1;
4      }
5      int addb() {
6         return 10;
7      }
8      void calc() {
9        int x = adda() + addb();
10       System.out.println(x);
11     }
12  }
13
14  class Y extends X {
15     int adda() {
16        return super.adda() + super.addb();
17     }
18  }
19
20  class Z extends Y {
21      int addb() {
22         return adda() + super.adda() + super.addb();
23      }
24  }
```

```
27  task {
28     X a;
29     Z c;
30
31     a = new X(1,6,0,East);
32     a.calc();
33
34     a = new Y(1,4,2,East);
35     a.calc();
36
37     a = new Z(1,2,4,East);
38     a.calc();
39
40     c = new X(1,5,1,East);
41     c.calc();
42
43     c = new Y(1,3,3,East);
44     c.calc();
45
46     c = new Z(1,1,5,East);
47     c.calc();
48  }
```

Geben Sie für jeden der Aufrufe der Methode `calc` in den Zeilen 32, 35, 38, 41, 44, 47 an, ob dieser Aufruf erfolgreich durchgeführt werden kann.

- Wenn ja, geben Sie die Ausgabe auf dem Bildschirm an.
- Wenn nein, geben Sie eine kurze Begründung.

Hinweis: `System.out.println(x)` schreibt die Zahl `x` auf den Bildschirm.

Aufgabe 4 (12 Punkte)

Implementieren Sie eine Methode `verdopple`, die einen Array von `double`-Zahlen erhält, und jede Zahl darin verdoppelt. Die Methode soll einen Array zurückgeben, der nach jedem Element des ursprünglichen Arrays das Ergebnis der Verdopplung erhält.

Beispiel:

Array, der an `verdopple` übergeben wird:

5.0	3.5	8.3	4.7

Rückgabe:

5.0	10.0	3.5	7.0	8.3	16.6	4.7	9.4

Hinweise: Die Methode soll natürlich für jeden beliebigen `double`-Array funktionieren (nicht nur für das angegebene Beispiel). Vergessen Sie nicht, einen passenden Methoden-Kopf zu definieren.

Aufgabe 5 (12 Punkte)

Implementieren Sie eine Roboter-Klasse `Navigator`. Ein `Navigator` kann sich, zusätzlich zu allen Fähigkeiten eines `Robot`s, in eine angegebene Richtung drehen, bzw. einen Beeper auf ein beliebiges Nachbarfeld verschieben.

5–a Geben Sie die Klassendefinition an, in die die in der Folge zu implementierenden Methoden integriert werden können

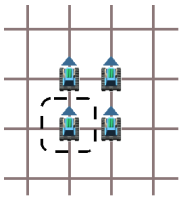
5–b Definieren Sie eine Methode `turn`, die eine Himmelsrichtung (Typ `direction`) als Argument erhält, und den Robot in die im Argument angegebene Richtung dreht.

5–c Definieren Sie eine Methode `moveBeeper`, die eine Himmelsrichtung (Typ `direction`) als Argument erhält, und einen auf dem momentanen Feld befindlichen Beeper auf das in der angegebenen Himmelsrichtung liegende Nachbarfeld verschiebt. Vor und nach der Ausführung der Methode soll der Zustand des Robots unverändert bleiben (gleiche Position, gleiche Anzahl von Beepern, gleiche Orientierung).

Hinweise: Sie können annehmen, daß auf dem Ausgangsfeld ein Beeper liegt. Um nach dem Verschieben den ursprünglichen Zustand des Robots wiederherstellen zu können, müssen Sie sich seine ursprüngliche Blickrichtung merken.

Aufgabe 6 (28 Punkte)

Definieren Sie eine Klasse `FatRobot` als Unterklasse von Robot. Ein `FatRobot` besteht aus 4 Robots, die in einem Quadrat angeordnet sind und sich immer synchron bewegen (siehe nebenstehende Skizze). Der Robot in der linken unteren Ecke ist der Hauptrobot, die anderen drei Roboter sind vom Typ `Navigator` (siehe Aufgabe 5) und werden im Hauptrobot als interne Datenkomponenten gespeichert.



6–a Geben Sie eine Klassendefinition mit geeigneten internen Variablen an, in denen die zusätzlichen Roboter gespeichert werden können.

6–b Definieren Sie einen Konstruktor mit der gleichen Schnittstelle wie ein herkömmlicher Robot, der den Hauptrobot auf den angegebenen Koordinaten, mit der angegebenen Anzahl von Beepern und mit der angegebenen Blickrichtung positioniert und die anderen drei Robots entsprechend anordnet (mit gleicher Blickrichtung und mit 0 Beepern).

6–c Ein `FatRobot` bewegt sich immer ein ganzes 2x2-Feld weiter, d.h. jeder der vier beteiligten Robots macht bei einem `move` 2 Schritte vorwärts. Definieren Sie eine entsprechende `move`-Methode.

Hinweis: Es können (vorübergehend) auch mehrere Roboter auf einem Feld stehen.

6–d Überschreiben Sie die `nextToABeeper`-Methode, sodaß sie `true` zurückgibt, wenn zumindest einer der vier Roboter auf einem Feld mit einem Beeper steht, und `false` sonst.

6–e Überschreiben Sie die Methode `pickBeeper`, sodaß ein Beeper, der unter einem der vier von den Robotern besetzten Feldern liegt, aufgehoben wird. Egal auf welchem Feld der Beeper aufgehoben wird, er muß nach der Ausführung der Methode vom Hauptrobot (links unten) gehalten werden. Sie können annehmen, daß auf zumindest einem der vier Felder ein Beeper liegt.

Hinweise: Um die Implementierung kurz zu halten, müssen Sie die für die Klasse `Navigator` in Aufgabe 5 definierten Methoden nutzen (auch wenn Sie sie nicht selbst implementieren konnten).
Wenn auf mehreren Feldern Beeper liegen, ist es egal welchen Beeper Sie aufheben (Sie dürfen aber nur einen aufheben).

6–f Schreiben Sie eine eine möglichst kurze Methode `pickAllBeepers`, die alle Beeper aufhebt, die unter den vier Robotern liegen. Am Ende der Methode soll der Hauptrobot alle aufgehobenen Beeper halten. Verwenden Sie zu einer möglichst kurzen Implementierung dieser Methode die in den vorhergehenden Punkten definierten Methoden (auch wenn Sie sie nicht selbst implementieren konnten).