Fachbereich Informatik
FG Eingebettete Systeme und
ihre Anwendungen
Prof. A. Koch

TECHNISCHE
UNIVERSITÄT
DARMSTADT

2. Teilprüfung
Allgemeine Informatik I
Wintersemester 2007/08
28. Februar 2007

Name (Nach-, Vorname)	
Matrikel-Nr.	**Lösungsvorschlag**
Unterschrift	
Prüfung Bitte ankreuzen	☐ Bachelor ☐ Master ☐ Diplom Fachrichtung: Wiederholer: ☐ Ja ☐ Nein
Anzahl abgegebene Zusatzblätter:	

Aufgabe	**max. Punkte**	**Erreicht**
1	20	
2	12	
3	12	
4	24	
5	12	
Summe	**80**	
Note		

Aufgabe 1 Grundlagen der Informatik (20 Punkte)

Beantworten Sie die folgenden Fragen möglichst kurz und prägnant (je 1-2 Sätze):

a) Was ist die wesentliche Eigenschaft der Universellen Turing-Maschine, die sie von der einfachen Turing-Maschine unterscheidet?

Aktuelles Programm wird als Daten auf Band gespeichert.

b) Welches ist die folgender Wahrheitstafel zugrundeliegende logische Funktion f mit $y = f(a,b)$? Geben Sie Ihre Antwort als booleschen Ausdruck für $f(a,b)$, bestehend aus den Operatoren $\wedge, \vee, \neg$, an!

a	b	y
0	0	1
1	0	0
0	1	0
1	1	1

$f(a,b) = (\neg a \wedge \neg b) \vee (a \wedge b) = (a \vee \neg b) \wedge (\neg a \vee b)$

c) Geben Sie den booleschen Ausdruck für einen Halbaddierer $f(a,b)$ zur Addition zweier einstelliger Binärzahlen a und b an! Sie können hier alle in der Vorlesung vorgestellten logischen Operatoren verwenden.

$sum(a,b) = (a \neq b) = (a \oplus b) = (a \wedge \neg b) \vee (\neg a \wedge b) = (a \vee b) \wedge (\neg a \vee \neg b)$
$carry(a,b) = (a \wedge b)$

d) Wie würde in den RGB und CMYK-Farbmodellen ein reines Blau dargestellt? Geben Sie jeweils die Werte der einzelnen Komponenten in einem Wertebereich $0 \ldots 255$ an!

RGB = (0,0,255)
CMYK = (255,255,0,0)

e) Was ist der wesentliche Unterschied zwischen den ASCII und ISO-Latin-1 Zeichenkodierungen?

ASCII kann mit 7-Bit nur 128 verschiedene Zeichen darstellen, ISO-Latin-1 mit 8-Bit doppelt so viele .

f) Wo liegt der Programmzähler in einer von-Neumann-Maschine und was ist seine Aufgabe?

Er liegt in der CPU und gibt die Adresse des nächsten auszuführenden Befehls an

g) Was ist die wesentliche Aufgabe eines Caches?

Agiert als schneller Zwischenspeicher zwischen CPU (hier auch OK: Register) und langsamerem Speicher

h) Was ist der Unterschied zwischem einem Programm und einem Prozess?

Mit Programm werden die abgespeicherten Anweisungen bezeichnet , ein Prozess ist das in Ausführung begriffene Programm .

i) Was für eine Kommandozeile könnten Sie in der Shell eingeben, um zu zählen, wieviele Dateien im aktuellen Verzeichnis irgendwo in ihrem Namen die Teile `Hello` und `World` enthalten?

`ls | grep Hello | grep World | wc`
oder `ls | grep World | grep Hello | wc`

`wc -w` ist genauso ok

j) Was ist der Sinn der Port-Nummern in TCP-Paketen?

Damit kann das Paket gezielt einem zuständigen Prozess (oder IP-Server) auf dem Zielrechner zugeordnet werden.

Aufgabe 2 Vererbung (12 Punkte)

Gegeben seien folgende Klassendefinitionen mit anschließendem Task. Geben Sie für jedes der Instanzierungs-/Aufrufpaare der Methode `calc()` aus den Zeilen 31–32, 34–36, 37–38, 40–41, 43–44 und 46–47 an, ob das Anweisungspaar erfolgreich durchgeführt werden kann:

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

Zur Erinnerung: `System.out.println(x)` gibt die Zahl x auf dem Bildschirm aus.

```
1  class A extends Robot {
2    int calcA() {
3      return 2;
4    }
5    int calcB() {
6      return 5;
7    }
8    void calc() {
9      int n = calcA() + calcB();
10     System.out.println(n);
11   }
12 }
13
14 class B extends A {
15   int calcA() {
16     return super.calcA() + super.calcB();
17   }
18 }
19
20 class C extends B {
21   int calcB() {
22     return   super.calcA() + calcA()
23            + super.calcB();
24   }
25 }
26
27 task {
28   A x;
29   C y;
30
31   x = new A(1,1,1,East);
32   x.calc();
33
34   y = new A(7,8,9,East);
35   y.calc();
36
37   x = new B(1,2,3,East);
38   x.calc();
39
40   y = new B(6,5,4,East);
41   y.calc();
42
43   x = new C(4,5,6,East);
44   x.calc();
45
46   y = new C(3,2,1,East);
47   y.calc();
48 }
```

34: `y = new A` und 40: `y = new B` schlagen fehl, da einer Variable mit statischem Typ einer Unterklasse dynamisch kein Objekt der Oberklasse zugewiesen werden kann

31: 7, 37: 12, 43: 26, 46: 26

Aufgabe 3 Elementweises Maximum in Matrizen (12 Punkte)

Implementieren Sie in KarelJ eine Methode

```
double[][] matmax (double[][] A, double[][] B)
```

die das elementweise Maximum der beiden Matrizen `A` und `B` als Matrix zurückgibt. Die Elemente aller Matrizen sind dabei vom Type `double`. Ihre Methode soll für beliebig große Matrizen funktionieren, dabei haben `A`, `B` und das Ergebnis zueinander aber immer die gleiche Anzahl von Zeilen und Spalten.

Es reicht, wenn Sie den Quelltext der Methode selbst angeben, Sie brauchen keine `class` oder `task`-Definitionen aufzuführen.

Beispiel: Für die Werte $A = \begin{pmatrix} 1 & 9 & -1 \\ 3 & 8 & -3 \end{pmatrix}$ und $B = \begin{pmatrix} 2 & 7 & -4 \\ 5 & 6 & -2 \end{pmatrix}$ wäre das Ergebnis von `matmax(A,B)` die Matrix $\begin{pmatrix} 2 & 9 & -1 \\ 5 & 8 & -2 \end{pmatrix}$.

Zur Erinnerung: Mehrdimensionale Arrays sind in KarelJ als Arrays bestehend aus Arrays realisiert.

```
// Deklaration der Methode:
double[][] matmax (double[][] A, double [][] B) {
  // bestimmen der variablen Dimension
  // bei Annahme von konstanter Dimension:.
  int rows = A.length;
  int cols = A[0].length;

  // Anlegen der neuen Matrix
  double[][] M = new double[rows][cols];

  // Korrekter Durchlauf durch Matrizen:
  for (int i = 0; i < rows; ++i)
    for (int j = 0; j < cols; ++j)
      // Elementweise Berechnung:
      if (A[i][j] > B[i][j]) // Loesung mit ?:-Operator ist auch OK
        M[i][j] = A[i][j];
      else
        M[i][j] = B[i][j];

  // Rückgabe des Ergebnis
  return M;
}
```

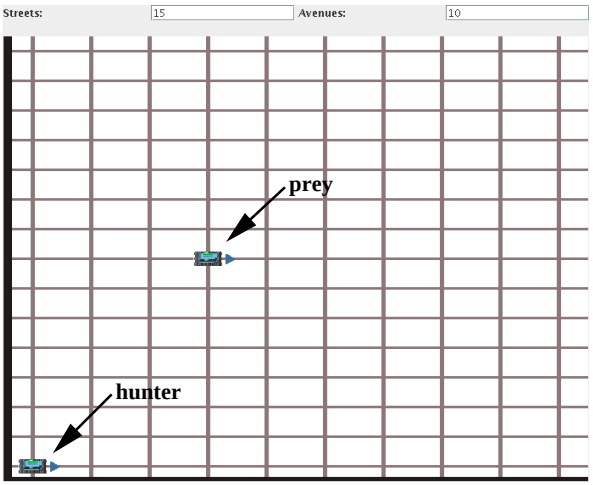
Aufgabe 4 Jäger-Sucher (24 Punkte)

Implementieren Sie eine Klasse `Hunter`. Roboter dieses Typs sollen bei der Konstruktion die Startposition, beschrieben durch Street und Avenue, als `prey` eine andere Roboter-Instanz (die Beute) sowie eine Schrittweite `speed` übergeben bekommen. Jeder Aufruf der `move()`-Methode des `Hunter`-Roboters soll den Jäger dann auf die Beute zubewegen, wobei in einem `move()` maximal `speed` Schritte auf der Straßenkarte gemacht werden dürfen. Dabei zählen als Schritte nur echte Ortswechsel, keine Drehungen.

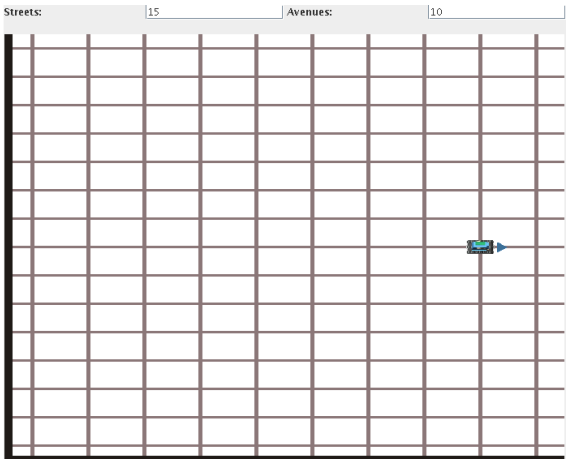
Solange der Jäger sein Ziel noch nicht erreicht hat, soll eine weitere Methode `hit()` den Wahrheitswert `false` liefern. Wenn der Jäger die Beute erreicht, soll er an dieser Stelle einen Beeper ablegen und für `hit()` nun `true` zurückgeben. Da ein `Hunter` nur eine Beute erjagen soll, reicht es, wenn er einen einzelnen Beeper an Bord hat.

Die Benutzung des `Hunter`-Roboters zeigt der folgende Beispiel-`task`: Hier bewegt sich die Beute beginnend bei Street=8, Avenue=4 langsam nach Osten, während sich der Jäger (beginnend bei Street=1, Avenue=1) in Dreierschritten auf sie zu bewegt und sie schliesslich bei Street=8, Avenue=9 erreicht.

```
1  task {
2    // Beute startet auf Street=8, Avenue=4
3    Robot  prey   = new Robot (8, 4, 0, East);
4    // Jäger startet auf Street=1, Avenue=1 und jagt prey in Dreierschritten
5    Hunter hunter = new Hunter(1, 1, prey, 3);
6
7    while (!hunter.hit()) {
8      prey.move();
9      hunter.move();
10   }
11 }
```



Ausgangszustand



Endzustand

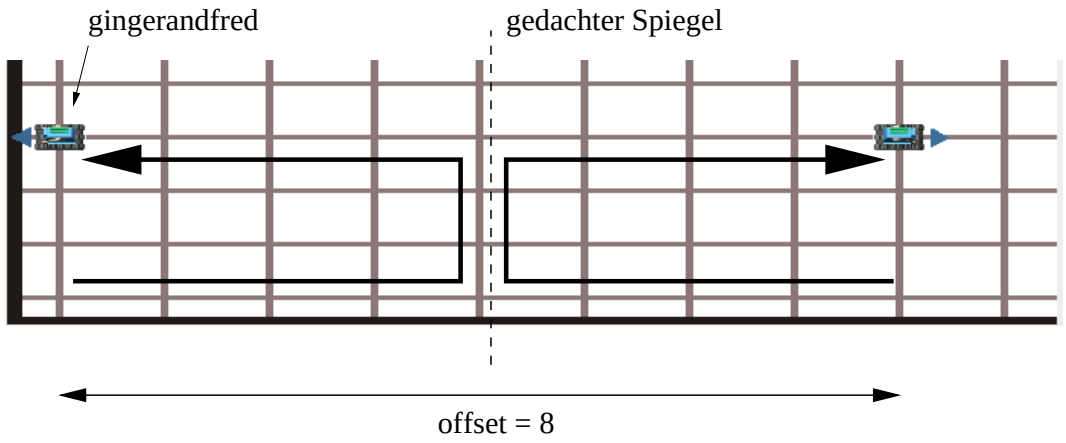
```
class Hunter extends Robot { // Deklaration:

  // je Variable/Attribut, insgesamt
  Robot myprey;
  int myspeed;
```

```
  // Deklaration Konstruktor
  Hunter(int street, int avenue, Robot prey, int speed) {
    // Aufruf super:
    super(street,avenue,1,East);
    // Setzen der Variablen:
    myprey = prey;
    myspeed = speed;
  }

  // Funktionalität turnTo:
  // wenn nicht in eigener Methode und mehrfacher Aufruf nötig:
  void turnTo(direction dir) {
    while (direction() != dir) turnLeft();
  }

  boolean hit() { // Deklaration:
    //  Funktionalität:
    return street() == myprey.street() && avenue() == myprey.avenue();
  }

  void move() { // Deklaration:
    int steps = myspeed;

    while (steps>0 && !hit()) { // Schleife insgesamt (Zähler, Bedingung...):

      // Jagdfunktionalität
      if (avenue() > myprey.avenue()) turnTo(West);
      else if (avenue() < myprey.avenue()) turnTo(East);
      else if (street() > myprey.street()) turnTo(South);
      else if (street() < myprey.street()) turnTo(North);

      super.move(); // super

      --steps;
    }

    // Funktionalität bei Treffer:(nur, wenn in move()-Methode, nicht in hit())
    if (hit()) putBeeper();
  }
}
```

Aufgabe 5 Spiegeltänzer (12 Punkte)

Implementieren Sie eine Klasse `MirrorDancer` als Unterklasse von `Robot`. Dem Konstruktor wird neben den üblichen Angaben auch noch als `int offset` eine ganze Zahl übergeben, die angibt, wieviele Streets weiter rechts (bei positivem `offset`) ein zweiter Roboter als "Spiegelbild" dieses Roboters auftauchen soll. Nun soll sich jede Bewegung einer Instanz von `MirrorDancer` auch an ihrem Spiegelbild zeigen, und zwar so, als ob genau zwischen beiden ein vertikaler Spiegel stehen würde. Auch das Ablegen von Beepern sollen die beiden Tanzpartner zusammen tun.

```
1  task {
2    MirrorDancer gingerandfred = new MirrorDancer(1, 1, 4, East, 8);
3
4    loop (3) gingerandfred.move();
5    gingerandfred.turnLeft();
6    loop (3) gingerandfred.move();
7    gingerandfred.turnLeft();
8    loop (3) gingerandfred.move();
9    gingerandfred.putBeeper();
10 }
```



Endsituation und Fahrtrouten nach Beispiel-`task`

Hinweis: Es geht um die genaue Spiegelung der *Ortswechsel* durch `move()`. Drehungen können (wegen des in `Robot` fehlenden `turnRight()`) nicht genau gespiegelt werden. `MirrorDancer` soll aber trotzdem die Funktionalitäten von `turnLeft()` und `turnRight()` implementieren.

```
class MirrorDancer extends Robot {

  // Deklaration:
  Robot partner;

  // Deklaration Konstruktor
  MirrorDancer(int street, int avenue, int beepers, direction dir, int offset) {
    super(street, avenue, beepers, dir);

    // Funktionalität Spiegeln der Ausgangsrichtung:
    direction mirrorDir = dir;
```

```
    if (dir == East)
      mirrorDir = West;
    else if (dir ==  West)
      mirrorDir = East;

    // Anlegen des Partners um offset versetzt:
    partner = new Robot(street, avenue + offset, beepers, mirrorDir);
  }

  // Funktionalität Bewegung:
  void move() {
    super.move();
    partner.move();
  }

  // Funktionalität gespiegelte Drehung nach Rechts:
  void turnRight() {
    super.turnLeft();
    super.turnLeft();
    super.turnLeft();
    partner.turnLeft();
  }

  // Funktionalität gespiegelte Drehung nach Links:
  void turnLeft() {
    super.turnLeft();
    partner.turnLeft();
    partner.turnLeft();
    partner.turnLeft();
  }

  // Funktionalität Ablegen von Beepers:
  void putBeeper() {
    super.putBeeper();
    partner.putBeeper();
  }
}
```