Mo., 12.12.2011 Informatik EF M (Kr)

2. Klausur 11/1 (EF.1) (A)

Dauer: 2 Schulstunden Name: ___www.r-krell.de_____

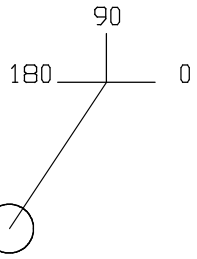
Hilfsmittel: --

* *Achte auf sorgfältige Darstellung mit vollständigem, nachvollziehbarem Lösungsweg!* *

* *Kommentiere deine Programme!* *

(1) Bewegung

Für ein Pendel ist nebenstehendes Programmstück gegeben.




```
01  int winkel = 225;  stift.runter();
02  while (winkel < 270)
03  {
04    stift.radiere(); stift.zeichneKreis(10);
05    stift.bewegeBis (x,y);
06    winkel = winkel + 4;
07    stift.normal(); stift.dreheBis (winkel);
08    stift.bewegeUm(60); stift.zeichneKreis(10);
09    Hilfe.warte(10);
10  }
```

a) Erläutere, was die Programmzeilen 04 und 05 bezwecken, wenn der Stift zuvor in der Mitte des Pendelkörpers steht/stand.

b) Erkläre die Bedeutung der Programmzeile 06!

c) Beschreibe, was die Zeilen 07 und 08 bewirken. Erläutere auch, warum dort *bewegeUm(60)* und nicht *bewegeBis(60)* steht. Welche Bedeutung hat dabei die 60?

d) Erläutere den Sinn von Zeile 09. Und begründe außerdem jeweils, ob *Hilfe.warte(10)* ebenso gut statt nach der Zeile 08 auch direkt vor Zeile 08 (praktisch als Zeile 07a) oder auch vor Zeile 04 (als Zeile 03a)) stehen könnte.

e) Beschreibe, von wo bis wo das Pendel durch das angegebene Programmstück bewegt wird (drei Stift&Co-Winkel sind eingezeichnet). Und: ist das Pendel am Ende der Schleife noch sichtbar?

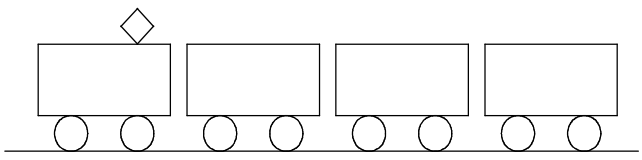
f) Für Aufgabenteil a) wurde angegeben, dass der Stift vor Zeile 04 in der Mitte des Pendelkörpers steht. Überlege und erläutere, ob dies wirklich bei jedem Durchlauf der *while*-Schleife sicher ist. Sofern möglich nenne einen Fall, wo der Stift woanders stehen könnte und überlege, ob das die Darstellung stört.

g) Die Pendelbewegung geht zu schnell. Nenne zwei Möglichkeiten, wie durch Ändern von jeweils einer Programmzeile die Pendelbewegung etwa halb so schnell gemacht werden kann. Begründe außerdem, welche der beiden möglichen Änderungen zu einem flüssigeren Bewegungsablauf führt!

h) Angenommen, das obige Programmstück steht im Konstruktor der Klasse Pendel (und dem Konstruktor werden außerdem Werte für die Koordinaten x und y des Dreh- bzw. Aufhängepunktes übergeben). In einer neuen Klasse *VielePendel* werden dann zwei Pendel durch nebenstehenden Programmtext erzeugt. Begründe, wann/wie sich die beiden Pendel bewegen: Bewegen sich beide gleichzeitig und gleich schnell oder kann man sagen, ob das linke oder das rechte Pendel zuerst am Ende sind?

```
import stiftUndCo.*;
public class VielePendel
{
  Bildschirm b = new Bildschirm();
  Pendel p1 = new Pendel(200,50);
  Pendel p2 = new Pendel( 80, 50);
  ...
```

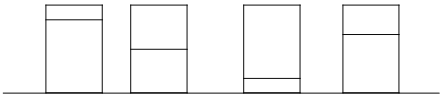
(2) Aufteilung in Klassen I



Eine Bahnszene zeigt einen Eisenbahnzug aus einer Lok und mehreren Wagen, die auf einem Gleis stehen oder fahren können. Erläutere nur in Deutsch (ohne Java-Text), welche Klassen sinnvoll sein können, ob für Lok und Wagen unbedingt verschiedene Klassen benötigt werden (oder wie das Problem des Stromabnehmers gelöst wird), in welchen Klassen ein Bildschirm gebraucht wird, und wo überall für das Fahren geeignete Methoden bereit gestellt werden müssen.

© R. Krell www.r-krell.de

3 Aufteilung in Klassen II



Ein einzelnes Altbierglas, von der Seite gesehen, besteht aus einem Rechteck der Höhe 110 Pixel und einem Querstrich, der die Höhe des Flüssigkeitsspiegels anzeigt (Füllstand zwischen 0 = leer und 100 = bis zum Eichstrich). Mehrere Altbiergläser können nebeneinander auf einem Tisch stehen. Das Bild wird einmal gezeichnet und bleibt (bei a) bis d)) unverändert.

a) Schreibe in Java die Klasse für ein *Bierglas* mit einem Konstruktor, dem eine Position (linke untere Ecke) und ein Füllstand übergeben werden können, der dann das Bierglas zeichnet! (Achtung: bei *stift.zeichneRechteck...* muss der Stift an der oberen linken Ecke stehen!)

b) Schreibe eine Klasse *TischMitBier* mit eigenem Bildschirm, um die Tischplatte zu zeichnen und darauf vier *Biergläse*r mit geeigneten Ortsangaben und den Füllständen 100, 60, 15 und 82 zu erzeugen.

c) Schreibe noch die Java-Startdatei mit der *main*-Methode (passend zu a) bzw. b))!

d) Erläutere kurz,
   - d1) warum die Startdatei ohne *import stiftUndCo.*;* auskommt
   - d2) warum in der Startdatei kein Bierglas erwähnt wird (und trotzdem das ganze Bild von einer Tischplatte und vier Gläsern erscheint, wenn die Startdatei gestartet wird)
   - d3) wie viele Buntstifte an der Entstehung des gezeigten Bildes beteiligt sind
   - d4) ob Tisch und Biergläser gleichzeitig gezeichnet werden, weil alles offenbar auf einen Schlag erscheint (und es ja auch mehrere Stifte gibt). Falls nicht, erläutere die Reihenfolge!

e) Kneipenbesucher haben es immer vermutet: Altbier verdunstet aus dem Glas! Erläutere nur in Deutsch (ohne Javatext, aber mit groben Funktionsbeschreibungen), welche zusätzlichen Methoden in den Java-Dateien a) bis c) definiert werden müssten, damit in allen Gläsern bzw. in jedem Bierglas der Füllstand pro Sekunde um einen Pixel sinken kann. Erkläre dabei auch, wo eine *while*-Schleife hin muss/darf und (von) wo die neuen Methoden aufgerufen werden!

4 Verkehrsampel

Eine Verkehrsampel enthält 3 Lampen.

a) Schreibe für die Klasse *Lampe* noch die Methode *an(),* die die Lampe in der anfangs gewählten Farbe leuchten lässt.

b) Schreibe eine neue Klasse *Ampel* mit einem Konstruktor, der ein Rechteck (=Kasten) geeigneter Größe zeichnet, in dem die drei Lampen passender Farbe zu sehen sind. Anfangs soll die Ampel rot zeigen!

c) Schreibe eine Methode *lichtwechsel*, die in einer Endlosschleife mit kurzen Wartezeiten (*Hilfe.warte...*) die Ampel auf rot & gelb, dann auf grün, dann auf gelb und schließlich wieder auf rot stellt -- und diesen Zyklus immer wiederholt.
Begründe außerdem, in welche Klasse die Methode *lichtwechsel* gehört.

```
import stiftUndCo.*;
public class Lampe
{
  BuntStift stift = new BuntStift();

  public Lampe (int xM, int yM, int farbNr)
  {
   stift.bewegeBis (xM, yM); // Mittelpunkt
   stift.setzeFuellMuster (Muster.GEFUELLT);
   switch (farbNr)
   {
     case 1: stift.setzeFarbe (Farbe.ROT); break;
     case 2: stift.setzeFarbe (Farbe.GELB); break;
     case 3: stift.setzeFarbe (Farbe.GRUEN); break;
   }
  }

  public void aus()  // aus = unsichtbar = Licht aus
  {
   stift.radiere();
   stift.zeichneKreis(10);  // löschen
  }
```