Prüfung Grundlagen der Programmierung 2
Prüfung Objektorientierte Programmierung
Prüfung Praktische Informatik 2
Prüfung Software-Entwicklung 2

Prüfer: Glavina, Grauschopf, Hahndel, U. Schmidt, Tröscher, Windisch
Prüfungsdauer: 90 Minuten
Hilfsmittel: keine

Aufgabe 1: (Basisalgorithmen und Container-Klassen, etwa 20% = 6+14)

a) Gegeben ist eine Liste mit Bezeichnungen von Tombola-Preisen (etwa "Teddy", "Uhr", "Reise", "Trostpreis", ...), sowie eine Liste mit den bereits vergebenen Preisen (etwa "Uhr", "Trostpreis", ...). Geben Sie eine Java-Methode `nochZuHaben` (mit den Parametern `preise` und `vergeben`) an, die einem potentiellen Loskäufer die Liste der noch zu gewinnenden Preise liefert.

b) Gegeben ist ein String mit Wörtern, die durch ein oder mehrere Leerzeichen voneinander getrennt sind (Wort = beliebige Zeichenfolge ohne Leerzeichen). Geben Sie eine Java-Methode `woerterListe` an, die aus einem solchen, als Parameter `woerter` übergebenen String die einzelnen Wörter extrahiert und alle diese Wörter als Liste zurückliefert:

```
LinkedList<String> woerterListe(String woerter) { ... }
```

Aufgabe 2: (Klassenmodellierung, etwa 25% = 8+3+8+6)

Gegeben sind die (Bruchstücke der) folgenden Klassen.

```
abstract class GeometrieObjekt { ... }

abstract class Koerper extends GeometrieObjekt { ... }

class Kugel extends Koerper {
  private double mx, my, mz;
  private double r;
}

class Quader extends Koerper {
  private double x1, y1, z1;
  private double x2, y2, z2;
}
```

```
abstract class Flaeche extends GeometrieObjekt { ... }

class Kreis extends Flaeche {
  private double mx, my; // Koordinaten des Mittelpunkts
  private double r; // Radius
}

class Polygon extends Flaeche
{
  private double[] xwerte; private double[] ywerte;
  // erste Ecke hat die Koordinaten (xwerte[0], ywerte[0]), usw.

  public Polygon(int n) { // lege Polygon mit n Ecken an
    xwerte = new double[n]; ywerte = new double[n];
  }

  public void setzeEckpunkt(int i, double x, double y) {
    xwerte[i] = x; ywerte[i] = y;
  }
}

class Rechteck extends Polygon
{ // Rechteck mit Seiten parallel zu den Koordinatenachsen
  public Rechteck(double hoehe, double breite) { ... }
}

class RWDreieck extends Polygon
{ // rechtwinkeliges Dreieck, mit rechtem Winkel rechts unten
  public RWDreieck(double grundlinie, double hoehe) { ... }
}
```

a) Zeichnen Sie für alle Klassen ein Klassendiagramm, aus dem auch die Vererbungsbeziehungen ersichtlich sind. Für jede Klasse sollte neben ihrem Namen auch Bezeichnung und Typ der Attribute sowie Bezeichnung, Ergebnistyp und Parameterliste der Methoden erkennbar sein. (Hinweis: Lassen Sie für Teilaufgabe b bereits hier bei jeder Klasse Platz für ein weiteres Attribut sowie eine weitere Methode!)

b) Ergänzen Sie nun Ihr Klassendiagramm um folgende Programm-Elemente:
-- `double rauminhalt;` // Rauminhalt bei Körpern
-- `double flaeche;` // Flächeninhalt bei 2D-Elementen
-- `String name;` // Name des geometrischen Elements
-- `void verschiebe(double x, double y)` // verschiebe ...
// ... ein 2D-Element um Vektor (x,y)
Vermeiden Sie hierbei unnötige (!) Redundanzen.

c) Implementieren Sie die Konstruktoren
-- `Kreis(double mx, double my, double r)`
-- `Rechteck(double hoehe, double breite)`
-- `RWDreieck(double grundlinie, double hoehe)`

d) Implementieren Sie weitere Teile des Programms (welche?), so dass der folgende Code in der Methode main funktionieren kann.

```
LinkedList<Flaeche> menge = new LinkedList<Flaeche>();
menge.add(new Kreis(2, 2, 5));
menge.add(new Rechteck(10, 3));
for (Flaeche f : menge)
  f.verschiebe(4, -1);
```

Aufgabe 3: (Code-Verständnis und OO-Konzepte, etwa 25% = 7+2+6+10)

Ein Stadtmagazin nutzt eine in Java geschriebene Applikation zur Verwaltung von Veranstaltungsdaten. Es sind unter anderem folgende Klassen innerhalb eines Pakets (package) definiert:

```
class Adresse {
    private String straße;
    private String hausNr;
    …
}
```

```
class Ort {
    private String name;
    private Adresse adresse;
    …
}
```

```
class Veranstaltung implements Comparable<Veranstaltung> {
    private int tag;
    private int monat;
    private String motto;
    private Ort ort;
    private Typ typ;
    …
}
```

```
import java.util.LinkedList;

class Szene extends LinkedList<Veranstaltung> {
    …
}
```

Für jede dieser Klassen können Getter und Setter für deren Attribute vorausgesetzt werden.

a) Überschreiben Sie für die Klassen `Adresse`, `Ort` und `Veranstaltung` die Methode `Object.toString`. Dabei soll `toString` von `Veranstaltung` die `toString`-Methoden der beiden anderen Klassen nutzen.

Beispiel: Für eine Veranstaltung mit dem Motto "Fegefeuer in Ingolstadt" am 13.7 im Stadttheater Ingolstadt mit der Adresse "Schlosslände 1" sollte `toString` liefern:
"`13.7, Stadttheater Ingolstadt, Schlosslände 1: Fegefeuer in Ingolstadt`"

b) Die Klasse `Veranstaltung` enthalte eine Aufzählung (enumeration) namens `Typ`, die die Werte `PARTY, KONZERT, VORTRAG, AUFFÜHRUNG` definiere. Geben Sie die Definition an.

Beachten Sie im weiteren, dass das Attribut `typ` von `Veranstaltung` ein solcher Enumeration-Wert ist.

c) Das parametrisierte `Comparable`-Interface schreibt vor, dass die Klasse `Veranstaltung` eine Methode zum Vergleichen zweier Veranstaltungen implementiert mit der Signatur:

```
public int compareTo(Veranstaltung other)
```

Dabei soll eine Veranstaltung "kleiner" sein als die andere, wenn sie an einem früheren Termin stattfindet. Sind die Termine gleich, so wird der Wert des lexikographischen Vergleichs der beiden Motto-Strings zurückgegeben.

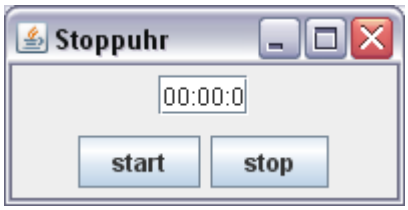
Geben Sie die Implementierung der Methode an.

d) Die Klasse `Szene` habe eine Methode `wasIstGeboten`, die als Parameter Veranstaltungstag und -monat sowie einen Veranstaltungstyp erhalte. `wasIstGeboten` liefert den Teil der Veranstaltungen des Szene-Objekts zurück, die an dem gegebenen Termin stattfinden und den gegebenen Typ haben. Der Rückgabeparameter soll dabei wieder eine `LinkedList<Veranstaltung>` sein.

Geben Sie die Implementierung der Methode an.

Aufgabe 4: (GUI-Programmierung, etwa 30% = 7+10+8+5)

Eine einfache Stoppuhr soll mithilfe von Swing realisiert werden (siehe Abbildung). Die Uhr zeigt die Zeit in der Form <Minuten>:<Sekunden>:<Zehntel> an. Gestartet wird die Zeitmessung mithilfe des start-Buttons. Während der Zeitmessung wird die jeweils vergangene Zeit angezeigt. Nach Drücken des stop-Buttons wird die Uhr angehalten. Bei erneutem Drücken des start-Buttons wird mit der Zeitmessung wieder bei "00:00:0" begonnen.



a) Skizzieren Sie die Komponenten-Hierarchie der GUI-Elemente des gezeigten Fensters. Geben Sie ggf. eingesetzte Layout-Manager sowie benötigte Layout-steuernde Informationen zu den Komponenten an.

b) Ergänzen Sie das nachfolgend vorgegebene Java-Programm so, dass es die oben beschriebene Stoppuhr-Funktionalität bietet und ein Fenster wie das oben Abgebildete erzeugt. Im einzelnen sind dabei folgende Methoden zu implementieren:
-- `run()` // in Klasse Timer
-- `Stoppuhr()` // Konstruktor der Klasse Stoppuhr
-- `actionPerformed()` // in Klasse Stoppuhr

Hinweise:
Zur korrekten Realisierung der Stoppuhr wird ein Timer-Thread eingesetzt, der die Zeitmessung (s. nächsten Hinweis) sowie die Aktualisierung der Anzeige vornimmt.
Mithilfe der Operation `sleep(int millisekunden)` kann man einen Thread für die angegebene Anzahl an Millisekunden anhalten.
Der Aufruf `String.format("%02d:%02d:%1d", m, s, z)` liefert einen String im für die Zeitanzeige benötigten Format.
Den Programm-Rahmen finden Sie auf der nächsten Seite.

```
public class Stoppuhr extends JFrame implements ActionListener {

    class Timer extends Thread {
        private Stoppuhr stoppuhr;
        private boolean terminiert = false;

        public Timer(Stoppuhr stoppuhr) {
            this.stoppuhr = stoppuhr;
        }

        public void run() {
            int min, sek, zehntel;

            //
            // bitte ergänzen!
            //
        }

        public void terminiere() {
            terminiert = true;
        }
    }

    private JButton startButton = new JButton("start");
    private JButton stopButton = new JButton("stop");
    private JTextField zeitFeld = new JTextField("00:00:0");
    private Timer timer;

    public Stoppuhr() {
        //
        // bitte ergänzen!
        //
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource==startButton && timer==null) {
            //
            // bitte ergänzen!
            //
        }
        else if (e.getSource==stopButton && timer==!null) {
            //
            // bitte ergänzen!
            //
        }
    }

    static public void main(String[] args) {
        Stoppuhr uhr = new Stoppuhr();
        uhr.setVisible(true);
    }
}
```

(Ende des Aufgabentextes) Viel Erfolg!