Prüfung Software-Entwicklung 2
Prüfung Praktische Informatik 2
Prüfung Grundlagen Informatik 2

Aufgabensteller: Dr. B.Glavina, Dr. T.Grauschopf, Dr. S.Hahndel, Dr. U.Schmidt
Prüfungsdauer: 90 Minuten
Hilfsmittel: keine

Aufgabe 1 (Datenstrukturen; etwa 22%)

a) Schreiben Sie eine generische Klasse `BinBaum` (in Java mit "Generics" oder in C++ mit "Templates"), die einen Binärbaum modelliert. Die Klasse enthalte zunächst nur Attribute und Konstruktor(en), aber noch keine Methoden.

b) Nehmen wir an, dass für jeden Knoten k eines Binärbaums gilt: die Knoten im linken Unterbaum von k sind alle kleiner als (oder gleich groß wie) der Wert von k und die Knoten im rechten Unterbaum von k sind alle größer als (oder gleich groß wie) der Wert von k.
Schreiben Sie eine Methode `reversePrint`, die die Bauminhalte in absteigender Reihenfolge ausgibt.
Skizzieren Sie die Struktur eines Baums, der die folgende Ausgabe liefern würde:

```
     11  7  5  3  2  1
```

c) Schreiben Sie zur obigen Klasse `BinBaum` eine Methode `anzBlaetter`, die die Anzahl der Blätter (=Knoten, die keine Unterbäume haben) des Baums ermittelt.

Aufgabe 2 (Klassen, Vererbung, Polymorphie; etwa 26%)

Erstellen Sie ein Programm in Java oder C++ zur Verwaltung von Kfz-Steuern. Je nach Motor und Schadstoffausstoß gelten folgende Steuersätze (in Euro) je angefangene 100 ccm Hubraum:

Motorarten	schadstoffarm	schadstoffreich
Ottomotor	10	20
Dieselmotor	15	35

Nur für Kraftfahrzeuge mit Dieselmotoren gilt außerdem, dass ein Rußpartikelfilter vorhanden sein kann. In diesem Fall wird die Kfz-Steuer um 500 Euro ermäßigt, allerdings höchstens auf 0 Euro.

a) Entwerfen Sie eine Klassenhierarchie mit Klassen für die verschiedenen Motorarten, mit Methoden zur Berechnung des Steuersatzes.

b) Ein Fuhrpark besteht aus vielen Kfz mit unterschiedlichen Motorarten. Erstellen Sie eine entsprechende Klasse mit einer Methode zur Ermittlung des gesamten Steueraufkommens des Fuhrparks.

c) Schreiben Sie eine Methode `main`, die einen Fuhrpark aus vier nach Motor und Schadstoffausstoß verschiedenen Kfz (davon eines mit Rußpartikelfilter) anlegt und dessen Steueraufkommen ausgibt.

Aufgabe 3 (eigene und vorgefertigte Klassen; etwa 26%)

In dieser Aufgabe soll ein Mobilfunknetz rudimentär simuliert werden. Implementieren Sie die im Folgenden erwähnten Klassen mit allen geforderten Attributen und Methoden. Zu einigen vorgefertigten Klassen finden Sie im Anschluss an diese Aufgabe kurze, exemplarische Codestücke.

a) Schreiben Sie eine Klasse `Sms`, die eine SMS-Kurznachricht modelliert. Sie hat als Attribute die Nummer des Senders der Nachricht (als `String`) und den Text. Der Konstruktor soll diese beiden Attribute mit gegebenen Werten belegen. Ist der Text länger als 160 Zeichen, so soll der Konstruktor eine Exception auslösen. Schreiben Sie dazu eine zusätzliche Klasse `SmsZuLangException`.

b) Schreiben Sie eine Klasse `Handy`, die als Attribute die Nummer des Handys und eine Liste empfangener SMS-Kurznachrichten hat. Sie dürfen dazu die (korrekt parametrisierte) Klasse `java.util.LinkedList` nutzen. Fügen Sie zur Klasse `Handy` die Methode `addNachricht` hinzu, die eine neue SMS zur Liste hinzufügt.

c) Schreiben Sie eine Klasse `Basisstation`, die eine Funkzelle und die darin befindlichen Handys repräsentiert. Versehen Sie die Klasse `Basisstation` mit einer geeigneten Datenstruktur als Attribut, in der die Handys gespeichert sind (denkbar sind z.B. eine Liste von Handy-Objekten oder ein `HashMap` mit den Handynummern als Schlüsseln und Handy-Objekten als Werten). Schreiben Sie zu `Basisstation` den Konstruktor und eine Methode

```
public boolean sendeSms(String nummer, Sms nachricht)
```

die versucht, die SMS `nachricht` an das Handy mit Nummer `nummer` in dieser Funkzelle durchzustellen, und die `true` zurückgibt, falls das gelingt.

Hinweis: Zur Unterstützung Ihres Gedächtnisses finden Sie auf der nächsten Seite Codestücke, die die Verwendung einiger Java-Klassen zeigen.

```
// java.util.LinkedList (mit Strings):
LinkedList<String> listOfStrings = new LinkedList<String>();
// String einfuegen:
listOfStrings.add("Hallo");
// Iteration ueber die Listen-Eintraege:
for (String s : listOfStrings)
  System.out.println(s);

// java.util.HashMap (mit String-Schluesseln und -Werten):
HashMap<String, String> map = new HashMap<String, String>();
// Schluessel-Wert-Paar hinzufuegen:
map.put("Key1", "Hallo");
// suche nach dem Wert zu einem Schluessel:
String s1 = map.get("Key1"); // s1 ist "Hallo"
s1 = map.get("xyz"); // s1 ist null
```

Aufgabe 4 (GUI-Programmierung in Java; etwa 26%)

Gegeben ist das Fenster einer Anwendung "Key Generator" im Zustand nach dem Start (siehe folgende Abbildung).

Der Benutzer kann nun in das Namen-Feld den Namen des Besitzers eines Programms eingeben, danach wird durch Drücken der Schaltfläche "Go!" die Erzeugung eines Registrierungsschlüssels gestartet. Der aus dem Namen erzeugte Schlüssel wird zentriert unterhalb des Namens angezeigt (initial mit "---" belegt). Es können hintereinander mehrere Schlüssel zu verschiedenen Namen erzeugt werden; nach getaner Arbeit wird das Programm mit der Schaltfläche "Exit" beendet.

a) Benennen Sie die im Fenster vorkommenden sechs GUI-Elemente; geben Sie dabei die Namen der `awt`-Klasse an.

b) Ergänzen Sie das folgende Programm (an den mit Punkten gekennzeichneten Stellen) so, dass das vom Programm erzeugte Fenster der obigen Abbildung gleicht und auch das Verhalten des Programms mit der obigen Beschreibung übereinstimmt. Achten Sie darauf, dass Ihr Programm vollständig ist, wobei Sie auf das Abschreiben der beiden Methoden `main` und `keygen` verzichten dürfen.

```
/* GUIaufgabe.java */
...

public class GUIaufgabe
...

{
  private TextField name;
  private Label key;

  public GUIaufgabe()
  {
    ...
    ...

  }

  public static void main(String[] args)
  {
    GUIaufgabe myWindow = new GUIaufgabe();
    myWindow.setVisible(true);
  }

  public static String keygen(String n)
  // liefert zu einem Namen einen gueltigen Schluessel
  {
    // braucht NICHT programmiert werden!!!
  }

  public void actionPerformed(ActionEvent e)
  {
    String buttonCommand = e.getActionCommand();
    ...
    ...

  }

} // end class GUIaufgabe

// end file GUIaufgabe.java
```

Hinweise:

- Die GUI-Elemente mit Textkomponenten haben zum Setzen bzw. Lesen des Textes die beiden Methoden
  ```
  public void setText(String neuerText)
  public String getText()
  ```
- Zur weiteren Unterstützung finden Sie im Anschluss an die Prüfungsangabe (als letztes Blatt) einen Ausschnitt aus dem im Praktikum bearbeiteten Programm "GUIdemo.java".

Hier folgt der Ausschnitt aus dem Programm "GUIdemo.java" (zu Aufgabe 4b):

```
...
// GUI-Komponente Label:
Panel panel1 = new Panel();
panel1.setLayout(new GridLayout(4,1));
panel1.add(new Label("(siehe Datei \"GUIdemo.txt\"!)"));
panel1.add(new Label("Links",Label.LEFT));
panel1.add(new Label("Zentriert",Label.CENTER));
panel1.add(new Label("Rechts",Label.RIGHT));
// Label sind passive Elemente (koennen keine Ereignisse ausloesen)
//
// GUI-Komponente Button:
Panel panel2 = new Panel();
Button button = new Button("Bing");
button.addActionListener(this);
panel2.add(button);
//
// GUI-Komponente Checkbox:
Panel panel3 = new Panel();
panel3.setLayout(new GridLayout(3,1));
Checkbox cb = new Checkbox("Doppelportion");
cb.addItemListener(this); // register me for events from cb
panel3.add(cb);
cb = new Checkbox("mit Sahne", true);
cb.addItemListener(this);
panel3.add(cb);
cb = new Checkbox("zum Mitnehmen", false);
cb.addItemListener(this);
panel3.add(cb);
//
// GUI-Komponente CheckboxGroup (RadioButtonGroup):
...
//
// GUI-Komponente TextField:
Panel panel5 = new Panel();
TextField tf = new TextField("Meier",20);
tf.addActionListener(this);
tf.addTextListener(this);
panel5.add(tf);
//
// GUI-Komponente TextArea:
Panel panel6 = new Panel();
TextArea ta = new TextArea("Java forever ...", 3, 15);
ta.addTextListener(this);
panel6.add(ta);
//
// GUI-Komponente Choice:
Panel panel7 = new Panel();
Choice choice = new Choice();
choice.addItemListener(this);
choice.add("rot");
choice.add("grün");
panel7.add(choice);
//
// GUI-Komponente List (ListBox):
Panel panel8 = new Panel();
List list = new List(4,false);
list.addActionListener(this);
list.addItemListener(this);
list.add("Äpfel");
list.add("Birnen");
list.select(1);
panel8.add(list);
//
// GUI-Komponente Scrollbar:
...
//
```