Studiengänge: Elektro- und Informationstechnik, Mechatronik, Informatik, Flug- und Fahrzeuginformatik, Wirtschaftsinformatik

Prüfung
Grundlagen der Programmierung 2
Objektorientierte Programmierung

Prüfer: Glavina, Hahndel, F.Regensburger, Schmidt, Windisch

Prüfungsdauer: 90 Minuten

Hilfsmittel: keine

Studiengang	Dozent	Matrikelnummer	Semester	Raum	Platz

Aufgabe	1	2	3	4	5	Σ	Note
Punkte							

Bitte beachten:

Tragen Sie Ihre persönlichen Angaben auf dieses Deckblatt ein.

Schreiben Sie Ihre Antworten direkt in die dafür vorgesehenen freien Stellen des Angabentextes.

Geben Sie alle Blätter wieder ab, auch wenn einzelne Seiten nicht beschrieben sein sollten.

Alle Blätter der Angabe müssen bei der Abgabe wieder richtig sortiert und geheftet sein!

Viel Erfolg!

Aufgabe 1 (Verständnisfragen; ca. 15 %)

a) Welche Ausgaben erzeugt das folgende Programm?
Schreiben Sie die Ausgabe in den untenstehenden Rahmen.

```
public class Tier {
   protected String name;
   public Tier(String name) {
      this.name = name;
      System.out.println("Tier " + name + " erzeugt");
   }
   public String toString() {
      return "Animal with name " + name;
   }
}

public class Insekt extends Tier {
   public Insekt(String name) {
      super(name);
      System.out.println(super.toString());
   }

   public String toString() {
      return "Insect " + name;
   }
}

public class Biene extends Insekt {
   public Biene() {
      this("Maja");
      System.out.println("Insekt");
      System.out.println(this);
   }

   public Biene(String s) {
      super(s);
      System.out.println(s);
   }

   public static void main(String[] args) {
      new Biene();
   }
}
```

Aufgabe 2 (`Comparable` und `Comparator`; ca. 15 %)

Die bei Jung und Alt beliebte Casting-Show DsdOb (Deutschland sucht den Oberblödel) läuft nach folgenden Regeln ab:

Jeder Kandidat versucht, eine möglichst lustige Grimasse zu schneiden. Seine Bemühungen werden von drei Juroren mit Einzelwertungen von 1 (sehr schlecht) bis 10 (sehr gut) bewertet. Die Gesamtwertung ergibt sich als arithmetisches Mittel der beiden höchsten Einzelwertungen; die verbleibende dritte Einzelwertung geht nicht in die Berechnung der Gesamtwertung ein.

a) Vervollständigen Sie die im folgenden Kasten angegebene Methode `setWertungen`, die zum Speichern der Einzelwertungen verwendet wird

```
import java.util.Arrays;
public class Kandidat implements Comparable<Kandidat> {
  private String name;
  private int wertungA, wertungB, wertungC;

  public Kandidat(String name){ this.name = name; }
  public String getName() { return name; }

  public void setWertungen(int wertungA, int wertungB, int wertungC){









  }
```

b) Implementieren Sie die Methode `compareTo`, die die natürliche Ordnung für die Klasse `Kandidat` festlegt. Die Reihenfolge der Kandidaten soll sich aus der lexikografischen Reihenfolge ihrer Namen ergeben.

```
  public int compareTo(Kandidat k) {









  }
```

c) Schreiben Sie für die Klasse `Kandidat` die Methode `berechneGesamtwertung`, die die Gesamtwertung eines Kandidaten nach den eingangs erwähnten Regeln berechnet.

```
  public double berechneGesamtwertung() {
    int[] tmp = new int[] {wertungA, wertungB, wertungC};









  }
```

Am Ende der Show wird die Rangliste der Kandidaten berechnet. Dabei sollen die Kandidaten absteigend nach der von `berechneGesamtwertung` ermittelten Gesamtpunktzahl sortiert sein. Der Kandidat mit den meisten Punkten kommt also zuerst. Falls zwei Kandidaten die gleiche Gesamtpunktzahl erhalten haben, entscheidet die aufsteigende lexikografische Ordnung der Kandidatennamen.

Ein Beispiel für eine Rangliste könnte so aussehen:

```
Rangliste der Kandidaten:
Name              A  B  C Gesamt
FlashMaster R     9  8  9  9,0
Zap Doodle        6  6  7  6,5
Zoe Maja          6  7  6  6,5
Il Bello          4  6  6  6,0
```

d) Implementieren Sie für die Klasse `WertungsComparator` die Methode `compare`, die zwei `Objekte` der Klasse `Kandidat` wie oben beschrieben vergleicht.

```
import java.util.Comparator;
public class WertungsComparator implements Comparator<Kandidat> {

  public int compare(Kandidat k1, Kandidat k2) {









  }
}
```

Aufgabe 3 (Generische Listen, Maps und Enum-Typen; ca. 25 %)

Gegeben sei eine Klasse `AuftragsListe` zur Verarbeitung von Fertigungsaufträgen für eine Firma, die Feuerwerkskörper herstellt. Desweiteren seien der Aufzählungstyp `ProduktTyp` und die Klasse `AuftragsPosition` gegeben. Die Implementierung der Klassen ist in den folgenden drei Rahmen in Auszügen angegeben.

```
public enum ProduktTyp { RAKETE, KERZE, BOELLER }
```

```
public class AuftragsPosition {
  private int nummer;
  private ProduktTyp typ;
  private int anzahl;

  public AuftragsPosition(int nummer, ProduktTyp typ, int anzahl) {
    this.nummer = nummer;
    this.typ = typ;
    this.anzahl = anzahl;
  }

  public ProduktTyp getTyp() { return typ; }
  public int getAnzahl() { return anzahl; }
}
```

```
public class AuftragsListe {
  private List<AuftragsPosition> aliste =
    new LinkedList<AuftragsPosition>();

  public static void main(String[] args) {
    AuftragsListe al = new AuftragsListe();

    al.add(new AuftragsPosition(3456, ProduktTyp.BOELLER, 10));
    al.add(new AuftragsPosition(3457, ProduktTyp.KERZE, 5));
    al.add(new AuftragsPosition(3458, ProduktTyp.RAKETE, 4));
    al.add(new AuftragsPosition(3459, ProduktTyp.BOELLER, 5));
    al.add(new AuftragsPosition(3460, ProduktTyp.KERZE, 3));
    al.add(new AuftragsPosition(3461, ProduktTyp.RAKETE, 8));

    Map<ProduktTyp, List<AuftragsPosition>> auftragsMap =
        al.gruppiereAuftraege();

    for (ProduktTyp typ : auftragsMap.keySet()) {
      System.out.printf("%3d Stueck vom Typ %s\n",
          AuftragsListe.berechneStueckzahl(auftragsMap.get(typ)), typ);
    }
  }
    . . .
}
```

Der Aufruf der Methode `main` der Klasse `AuftragsListe` liefert folgende Ausgabe:

```
 12 Stueck vom Typ RAKETE
  8 Stueck vom Typ KERZE
 15 Stueck vom Typ BOELLER
```

In den folgenden drei Teilaufgaben müssen Sie die fehlenden Methoden `add`, `gruppiereAuftraege` und `berechneStueckzahl` implementieren.

Bei der Implementierung der fehlenden Methoden können Sie unter anderem auf folgende Methoden der generischen Klasse `Map<K,V>` über Schlüsseltyp `K` und Wertetyp `V` zurückgreifen:

```
boolean containsKey(Object key):
   Prüft, ob mit dem Schlüssel key ein Wert assoziiert ist

V get(Object key):
   liefert den mit dem Schlüssel key assoziierten Wert zurück

V put(K key, V value):
   assoziiert den Wert Value mit dem Schlüssel key.
   Liefert den bisher mit key assoziierten Wert zurück oder null, falls
   bisher kein Wert mit key assoziiert war.
```

a) Implementieren Sie in der Klasse `AuftragsListe` die Methode

```
void add(AuftragsPosition apos)
```

die dem Attribut `aliste` der Klasse `AuftragsListe` einen neuen Eintrag hinzufügt, falls die übergebene Referenz `apos` nicht `null` ist.

```
public void add(AuftragsPosition apos) {




}
```

b) Implementieren Sie in der Klasse `AuftragsListe` die Methode

```
public static int berechneStueckzahl(List<AuftragsPosition> al)
```

die die Anzahl aller in der Liste enthaltenen Feuerwerkskörper berechnet und zurückgibt.

```
public static int berechneStueckzahl(List<AuftragsPosition> al) {









}
```

c) Implementieren Sie in der Klasse `AuftragsListe` die Methode

```
public Map<ProduktTyp, List<AuftragsPosition>> gruppiereAuftraege()
```

die ausgehend vom Inhalt des Attributs `aliste` eine Map erzeugt und als Resultat zurückgibt. Diese Map soll zu Schlüsseln vom Typ `ProduktTyp` jeweils eine Liste aller Auftragspositionen speichern, die in `aliste` enthalten sind und einen dem Schlüssel entsprechenden Produkttyp haben. Sorgen Sie dafür, dass die Map nur Schlüssel enthält, für die es Auftragspositionen im Attribut `aliste` gibt (keine leeren Listen assoziieren).

```
public Map<ProduktTyp, List<AuftragsPosition>> gruppiereAuftraege() {

   Map<ProduktTyp, List<AuftragsPosition>> map =
      new TreeMap<ProduktTyp, List<AuftragsPosition>>();









    return map;
}
```

Aufgabe 4 (String-Verarbeitung und Exceptions, ca. 25 %)

Diese Aufgabe behandelt die Berechnung der Prüfziffer einer ISBN-10 Nummer. Die Abkürzung ISBN steht für *International Standard Book Number*. Eine ISBN-10 Nummer besteht aus genau 13 Zeichen und hat das folgende Format:

d_1 - d_2 d_3 d_4 - d_5 d_6 d_7 d_8 d_9 - p

Dabei bezeichnen d_1 bis d_9 Ziffern aus dem Bereich 0 bis 9 (Dezimalziffern). Das Zeichen p ist die sogenannte Prüfziffer, die entweder auch eine Dezimalziffer ist oder aber das Zeichen X sein kann. Beispiele für ISBN-10 Nummern sind:

0-262-03293-7, 3-540-46660-6 und 0-201-40009-X

Die Prüfziffer p wird durch die Bildung einer gewichteten Quersumme der Ziffern d_1 bis d_9 und anschließender Anwendung von Arithmetik modulo 11 berechnet. Dadurch können Reste von 0 bis 10 entstehen. Die Prüfziffer p soll aber stets einstellig sein, und daher wird bei Auftreten eines Rests 10 das Zeichen X als Prüfziffer verwendet . Der Algorithmus zur Berechnung der Prüfziffer p ist im nachfolgenden Pseudo-Code beschrieben.

Pseudo-Code für Algorithmus zur Berechnung der Prüfziffer p:

Schritt 1: Bilde die gewichtete Summe s
$s = 1*d_1 + 2*d_2 + . . . + 8*d_8 + 9*d_9$

Schritt 2: Berechne den ganzzahligen Rest r bei Division von s durch 11

Schritt 3: Leite die Prüfziffer p aus dem Rest r ab
Falls r < 10 dann entspricht die Prüfziffer p der Dezimalziffer mit Wert r
ansonsten entspricht p dem Zeichen X (großes X für römisch 10)

Implementieren Sie für die Klasse `ISBN10` die Methode `check` mit folgender Signatur:

```
static public boolean check(String isbn) throws InvalidFormatException
```

Die Exception-Klasse `InvalidFormatException` stellt folgenden Konstruktor bereit:

```
public InvalidFormatException(String msg)
```

Die Methode `check` soll sich wie folgt verhalten:

- Falls einer der folgenden Fälle vorliegt, soll eine `InvalidFormatException` mit einer aussagekräftigen Fehlermeldung geworfen werden.
  - der übergebene String `isbn` ist nicht genau 13 Zeichen lang
  - an den Indexpositionen 1, 5 und 11 des übergebenen Strings `isbn` steht nicht das Zeichen `-`
  - die Zeichen d_1 bis d_9 sind nicht alle Dezimalziffern oder das letzte Zeichen des übergebenen Strings `isbn` ist weder eine Dezimalziffer noch das Zeichen X
- Falls jedoch alle syntaktischen Regeln eingehalten sind, soll die Methode check aus den Ziffern d_1 bis d_9 eine Prüfziffer p nach obigem Pseudo-Code berechnen und als Funktionsresultat einen Wahrheitswert zurückliefern, der aussagt ob die berechnete Prüfziffer p mit dem letzten Zeichen des übergebenen Strings `isbn` übereinstimmt.

Beispiele zur Verdeutlichung der Aufgabenstellung:

- `ISBN10.check("0-262-03293-7")` liefert `true` denn
  `1*0+2*2+3*6+4*2+5*0+6*3+7*2+8*9+9*3 == 161 und 161 % 11 == 7`
- `ISBN10.check("0-201-40009-X")` liefert `true` denn
  `1*0+2*2+3*0+4*1+5*4+6*0+7*0+8*0+9*9 == 109 und 109 % 11 == 10`
- `ISBN10.check("0-362-03293-7")` liefert `false` denn
  `1*0+3*2+3*6+4*2+5*0+6*3+7*2+8*9+9*3 == 163 und 163 % 11 == 9`
- `ISBN10.check("0:201-40009-X")` wirft eine `InvalidFormatException`

Bei der Implementierung der Methode `check` könnten Ihnen folgende Methoden der Klassen `String` und `Character` nützlich sein:

Klasse `String`:

`public char charAt(int index):`
liefert das Zeichen an Position `index` im String

Klasse `Character`:

`public static boolean isDigit(char ch):`
prüft, ob das Zeichen `ch` eine Dezimalziffer ist

`public static int digit(char ch, int radix):`
berechnet den numerischen Wert des Zeichens `ch` im Zahlsystem mit Basis `radix`.

Implementieren Sie die Methode `check` im nachfolgenden Kasten.

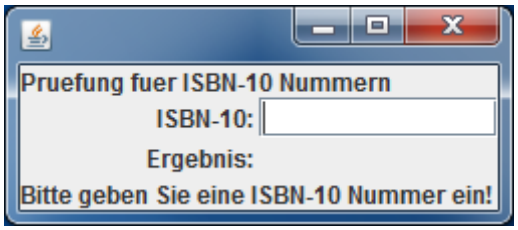
```
public class ISBN10 {
  static public boolean check(String isbn) throws InvalidFormatException{
    int sum = 0;
    int factor = 1;









   // Fortsetzung auf nächster Seite
```

```
   }  // Ende Methode check
} // Ende Klasse ISBN10
```

Aufgabe 5 (GUI und Events, ca. 20 %)

In dieser Aufgabe sollen Sie die nebenstehende grafische Oberfläche für die Prüfung von ISBN10 Nummern realisieren.

Die Aufgabe kann vollkommen unabhängig von der Aufgabe 4 dieser Klausur bearbeitet werden!



Auf der nächsten Seite ist der Rumpf der Klasse `CheckIsbnGui` abgedruckt, deren Implementierung Sie vervollständigen müssen.

Der Anwender kann in der Oberfläche eine ISBN10-Nummer eingeben; nach Betätigung der Enter-Taste (Eingabe-Taste) prüft die Anwendung, ob der eingegeben String der Syntax einer ISBN10-Nummer entspricht und ob es sich um eine gültige ISBN10-Nummer handelt.

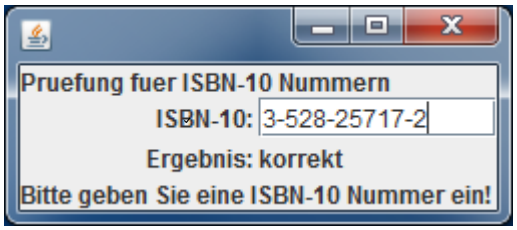
Die Prüfung erfolgt in einer beim zuständigen `JTextField` registrierten Callback-Methode. Bei der Implementierung dieser Callback-Methode können Sie die statische Methode

```
static public boolean check(String isbn) throws InvalidFormatException
```

der Klasse `ISBN10` einsetzen, die Sie hier als gegeben voraussetzen können.

Die Methode `ISBN10.check` löst, abhängig vom übergebenen Argument, entweder eine Ausnahme aus oder liefert einen Wahrheitswert zurück. Nachfolgend wird für jeden der drei Fälle das gewünschte Verhalten der Callback-Methode und damit der Oberfläche spezifiziert:

- Falls der übergebene String `isbn` der Syntax einer ISBN10-Nummer entspricht und die eingegebene ISBN10-Nummer eine gültige Ziffernkombination darstellt, liefert `ISBN10.check` den Warheitswert `true`. In diesem Fall soll die Callback-Methode in der Oberfläche im Ergebnisfeld den Text `"korrekt"` ausgeben



- Falls der übergebene String `isbn` nicht der Syntax einer ISBN10-Nummer entspricht, löst `ISBN10.check` eine Ausnahme `InvalidFormatException` aus.
  In diesem Fall soll die Callback-Methode die Ausnahme fangen und in der Oberfläche im Ergebnisfeld den Text `"falsches Format"` ausgeben

- Falls der übergebene String `isbn` zwar der Syntax einer ISBN10-Nummer entspricht, aber die eingegebene ISBN10-Nummer keine gültige Ziffernkombination darstellt (Prüfziffer stimmt nicht), liefert `ISBN10.check` den Wahrheitswert `false`.

In diesem Fall soll die Callback-Methode in der Oberfläche im Ergebnisfeld den Text "`falsche Pruefziffer`" ausgeben.

Hinweise

`JTextField` sendet einen `ActionEvent`, wenn die Enter-Taste gedrückt wird.

Die Beschriftung von `JLabel` bzw. der Text von `JTextField` lassen sich lesen und schreiben mit

```
String getText()
void setText(String s)
```

Geben Sie die vollständige Implementierung der Callback-Methode und des Konstruktors im nachfolgenden Kasten an.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class CheckIsbnGui extends JFrame implements ActionListener {
  private JTextField isbn10Feld;
  private JLabel pruefFeld;

  public static void main(String[] args) {
     new CheckIsbnGui();
  }

  // Hier die vollständige Implementierung der Callback-Methode
  // und des Konstruktors angeben









// Fortsetzung nächste Seite
```

```
} // Ende der Klasse CheckIsbnGui
```