PRG 1

Klausur

Na[redacted]

A1 40
A2 30
A3 .....
A4 40

Summe 100
Note 1,0

Die Klausur ergibt max. 100 Punkte: **Streichen Sie eine der Aufgaben 3-5!** Schreiben Sie Ihre Lösung leserlich **auf die angefügten Blätter, nutzen Sie die letzten Blätter als Kladde**. Einziges erlaubtes **Hilfsmittel** ist ein einseitig handbeschriebenes A4-Blatt, das **mit abzugeben** ist. Schreiben sie bitte Ihren **Namen auf jedes Blatt** und geben Sie alle Blätter ab!

Klausurtermin: 18.07.2016 10:00 ( ) **letzter Versuch**

Streichen Sie bitte eine der Aufgaben 2-4!

Aufgabe 1. (40 Punkte): *Kreuzen* Sie die richtigen Antworten an und geben Sie Ihre *Begründung* in Stichworten dazu. Ohne Begründung gilt Ihre Antwort als falsch.

( ) Richtig
(X) Falsch
1. Der Rückgabetyp von Konstruktoren ist stets vom Typ Integer.

Grund: Konstruktoren haben per Definition nie einen Rückgabetyp

( ) Richtig
(X) Falsch
2. Ein Interface implementiert man in einer Klasse, die man mit "extends" erweitert.

Grund: extends wird für Vererbung benötigt, Interfaces nutzen Implements

( ) Richtig
(X) Falsch
3. In der geschachtelten Schleife `for(int x = 0; x <= 10; ) for(int y =0; y < 6; y++) b++;` wird b 60 mal inkrementiert..

Grund: Da x nie inkrementiert wird, ist dies eine Endlosschleife

( ) Richtig
(X) Falsch
4. In einer Klasse vom Typ ArrayList<String> können beliebige Objekte gespeichert werden.

Grund: Es können nur Objekte vom Typ String gespeichert werden

( ) Richtig
(X) Falsch
5. Eine Ausnahme(Exception) muss immer in der aufrufenden Methode gefangen werden.

Grund: Sie kann mit dem Throws Schlüsselwort auch nach oben gereicht werden, oder vom Typ RuntimeException sein

(X) Richtig
( ) Falsch
6. Ein Typecast eines Werts vom Typ double mit dem Typ float kann zu einem Genauigkeitsverlust führen.

Grund: Float hat aufgrund seiner kleineren größe nicht die Möglichkeit so genau zu sein wie double

(X) Richtig
( ) Falsch
7. Bei der Implementierung eines Interfaces müssen alle dort deklarierten Methoden überschrieben werden.

Grund: Ein Interface ist eine Vorgabe für Funktionalität * Siehe Rückseite

(X) Richtig
( ) Falsch
8. Eine Methode mit dem Rückgabetyp int hat mindestens eine return-Anweisung, bei der ein Wert zurückgegeben wird.

Grund: Es muss ein Wert vom Typ int bei jedem return zurückgegeben werden

( ) Richtig
(X) Falsch
9. Interfaces können wie Klassen instanziiert werden.

Grund: Interfaces können nicht typ instanziert werden, der Konstruktor ist nicht aufrufbar

(X) Richtig
( ) Falsch
10. Folgende Referenzierung ist möglich, wenn Fahrrad die Superklasse von Rennrad ist.
`Fahrrad fahrrad = new Rennrad();`

Grund: Die Polymorphie erlaubt uns dies

Aufgabe 2. (30 Punkte) Schreiben Sie eine Methode, die überprüfen soll, ob ein Benutzer registriert ist. Die Methode soll folgende Signatur haben:

```
public boolean authenticate(String userName, String passWord)
```

Der Parameter userName ist der übergebene Benutzername, der Parameter passWord das zugehörige Passwort. Die Benutzerdaten sind in der Klasse (die sie nicht kennen müssen) in den beiden Arrays

```
ArrayList<String> users;
ArrayList<String> passwords;
```

gespeichert. Die Methode gibt true zurück, wenn der Benutzer gefunden wurde und das Passwort richtig ist, ansonsten false. Hinweis: Bei Strings verwendet man die Methode equals um zwei Strings auf Gleichheit zu überprüfen.

Lösung auf Seite Nr. 4

Aufgabe 3. (30 Punkte) Schreiben Sie eine animierte Klasse mit dem Namen MessageClient. Diese erhält im Konstruktor ein Objekt vom Typ Queue<Message>. Die Klasse Message ist eine vereinfachte Datenstruktur zum Speichern von Nachrichten.

```
class Message {
   public String from;      // Absender
   public String subject;   // Betreff
   public String text;      // Nachricht
}
```

Queue<Message> bietet die Methoden `public void add(Message message)` und
`Message remove() throws EmptyException`

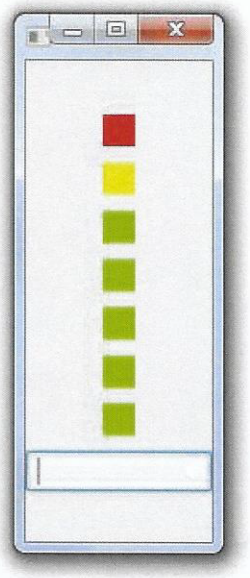
Die Klasse soll eine Liste aller Nachrichten über die Console ausgeben und dabei Absender und Betreff listen. Zusätzlich soll es möglich sein, nur die Nachrichten eines bestimmten Absenders anzuzeigen. Dazu wird das Attribut `showOnlyMessagesFrom` vom Typ String verwendet. Ist dieser String null, werden alle Absender angezeigt.

Lösung auf Seite Nr. /

Aufgabe 4. (30 Punkte): Eine Aussteuerungsanzeige für eine Audioaufnahme stellt die Pegelwerte grafisch als Leuchtbalken mit LEDs dar. Der Aussteuerungsbereich reicht von -100 bis +10 dB. Bei einem Pegelwert unter -40 dB leuchtet keine LED. In einem Bereich von -40 bis -31 dB leuchtet die unterste LED, zwischen -30 und -21 dB die untersten beiden usw. Bei einem Pegel von +6 dB leuchten alle LEDs.

Eine JavaFX-Anwendung kann dies mit folgendem Basiscode simulieren:

```
public class LevelMeter extends Application {
   Node[] node = new Node[8];

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

   @Override
   public void start(Stage primaryStage) throws Exception {
     for(int i = 0; i < 7; i++)
       node[i] = (Node)new Rectangle(20, 20, Color.GRAY);

     // An dieser Stelle soll Ihr Code für die Erweiterung
     // mit einem Textfeld stehen.
     //
     // Fügen Sie das Textfeld als 8. Node-Objekt hinzu

     VBox levelmeter = new VBox(10, node);
     levelmeter.setAlignment(Pos.CENTER);

     Scene scene = new Scene(levelmeter, 50, 300);
     primaryStage.setScene(scene);
     primaryStage.show();

   }
```



Erweitern Sie die VBox um ein Textfeld, mit welchem die Pegelwerte übergeben werden können. Implementieren Sie eine Methode `void showLevel(String strLevel)`, die als ActionEvent aufgerufen wird, wenn der Benutzer nach Eingabe des Pegelwerts in diesem Textfeld die Returntaste drückt. Die Methode hat zwei Aufgaben:

1. zunächst alle LEDs (Rectangle-Objekte) auf die Farbe grau setzen:
   ```
   for(int i = 0; i < 7; i++)
         ((Rectangle)node[i]).setFill(Color.GRAY);
   ```
2. entsprechend des Pegelwerts die jeweiligen LEDs einzufärben: -40, -30, -20, -10, 0 (grün), +3 (gelb), +6 (rot)
   (das ist Ihre Aufgabe ☺)

Lösung auf Seite Nr. 5 und 6

Beginnen Sie hier mit Ihrer Lösung!

Benutzen Sie die letzten Seiten als Schmierblätter und streichen Sie deutlich alles, was nicht gewertet werden soll.

Name: [redacted]

```
public boolean authenticate(String userName, String passWord){
    // Annahme1: die beiden ArrayLists haben die gleiche
    // Länge
    // Annahme2: Die Benutzernamen und Passwörter passen
    jeweils am gleichen Index zusammen.
    for(int i=0; i < users.size(); i++) {
        if(users.get(i).equals(userName)&&
            passwords.get(i).equals(passWord)) {
            return true;
        }
    }
    return false;
}
```

Name: [redacted]

```
TextField field = new TextField();
field.setOnAction((ActionEvent eV) -> {
    showLevel(field.getValue());
});
```

getText()

```
node[7] = field;
```

// jetzt außerhalb von start();

```
private void showLevel(String strLevel) {
    for(int i=0; i<7; i++) {
        ((Rectangle) node[i]).setFill(Color.GRAY);
    }
    int value = Integer.parseInt(strLevel);
    if(value<=-31 && >=-40)
        ((Rectangle) node[6]).setFill(Color.GREEN);
    if(value<=-21 && >=-30)
        ((Rectangle) node[5]).setFill(Color.GREEN);
    if(value<=-11 && >=-20)
        ((Rectangle) node[4]).setFill(Color.GREEN);
    if(value<=-1 && >=-10)
        ((Rectangle) node[3]).setFill(Color.GREEN);
```

Name: [redacted]

```
if(value<=2 && >=0)
    ((Rectangle) node[2].setFill(Color.GREEN));
if(value<=5 && >=3)
    ((Rectangle) node[1].setFill(Color.YELLOW));
if(value>=6)
    ((Rectangle) node[0].setFill(Color.RED));
}
```

✓

```
//TODO die Benutzereingabe validieren, ich denke
//nicht, dass dies Teil der Aufgabe ist, falls doch
//hier Pseudocode:
// try {
// int value = Integer.parseInt(strLevel);
// catch { kann nicht geparsed werden Exception }
// { return; }
// if(value < -100 || > 10)
//   throw new FalscherWertException();
```

Name: [redacted]