PRG 1 Hauptklausur

Name: _____ **Matrikel-Nr**.: _____ **Platz:** ____

Die Klausur ergibt max. 100 Punkte: **Streichen Sie eine der Aufgaben 3-5!** Schreiben Sie Ihre Lösung leserlich **auf die beiden angefügten Blätter**. Einziges erlaubtes **Hilfsmittel** ist ein einseitig handbeschriebenes A4-Blatt, das **mit abzugeben** ist. Die Abgabe der Schmierblätter wird empfohlen. Schreiben sie bitte Ihren **Namen auf jedes Blatt**!

Klausurtermin: 05.02.2015 12:00 ( ) **letzter Versuch**

Streichen Sie bitte eine der Aufgaben 3-5!

A1.....
A2 .....
A3 ……..
A4 ……....
A5 ……..
Summe ……....
Note .....

Aufgabe 1. (30 Punkte): *Kreuzen* Sie die richtigen Antworten an und geben Sie Ihre *Begründung* in Stichworten dazu. Ohne Begründung gilt Ihre Antwort als falsch.

( ) Richtig
( ) Falsch
1. In der folgenden geschachtelten Schleife wird ***zahl*** insgesamt 200 mal hochgezählt:
```
for(int i=0;i<10;i++) for(int j=0;j<20;j++) zahl++;
```
Grund: _____

( ) Richtig
( ) Falsch
2. Weist man einer Referenz auf ein Objekt null zu, so werden auch ***alle anderen*** Referenzen auf dieses Objekt auf null gesetzt.

Grund: _____

( ) Richtig
( ) Falsch
3. Beim "Fangen" (catch) werden Ausnahmen ***nach ihrem Typ*** unterschieden.

Grund: _____

( ) Richtig
( ) Falsch
4. Alle Implementierungen von `java.util.List` haben eine ***feste maximale Größe***.

Grund: _____

( ) Richtig
( ) Falsch
5. Um eine Methode zu überschreiben, müssen der ***Name und der Rückgabetyp*** übereinstimmen.

Grund: _____

( ) Richtig
( ) Falsch
6. Eine Funktion kann ***wie eine void-Methode*** aufgerufen werden.

Grund: _____

( ) Richtig
( ) Falsch
7. Eine Klasse kann ***mehr als einen Konstruktor*** besitzen.

Grund: _____

( ) Richtig
( ) Falsch
8. Ein Objekt der Klasse `java.util.ArrayList<String>` ist ***ohne Veränderungen im Programm*** gegen ein Objekt der Klasse `java.util.Vector` austauschbar.

Grund: _____

( ) Richtig
( ) Falsch
9. Ein Aufruf der folgenden Methode ist ***wirkungslos***, d.h. er kann einfach gestrichen werden:
```
public void addiere(double a, double b) { a = a + b; }
```
Grund: _____

( ) Richtig
( ) Falsch
10. Das interface ABC wird von den Klassen A, B, C implementiert. Damit ist die folgende Zuweisung zulässig: `A var = new ABC();`

Grund: _____

Aufgabe 2. (10 Punkte) Realisiert die unten stehende Klasse einen LIFO-Container oder einen FIFO-Container? Elemente welchen Typs von Elementen kann darin gespeicher werden, und wieviele? Begründen Sie bitte Ihre Aussage!

```
public class FolderHolder {
    List<Folder> list = new ArrayList<Folder>();
    public void add(Folder foo)
    {    list.add(0, foo); }
    public Folder remove() throws EmptyException
    {    return list.remove(0); }
}
```

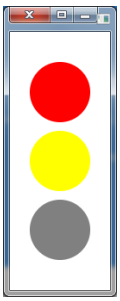
Lösung auf Seite Nr. _____

Aufgabe 3. (30 Punkte) Hier sehen sie den Basiscode für die Darstellung einer Ampel in JavaFX.

```
public class TrafficLightGui extends Application {
      Circle redlight, yellowlight, greenlight;

      public void start(Stage primaryStage) {
            redlight = new Circle(35, Color.RED);
            yellowlight = new Circle(35, Color.YELLOW);
            greenlight = new Circle(35, Color.DARKGREEN);
            // Startzustand: Rot-Gelb
            greenlight.setFill(Color.GREY);
            VBox lightbox = new VBox(10, redlight, yellowlight, greenlight);
            lightbox.setAlignment(Pos.CENTER);

            Scene scene = new Scene(lightbox, 50, 300);
            primaryStage.setScene(scene);
            primaryStage.show();
      }
      public static void main(String[] args) {
            launch(args);
      }
  }
```



Schreiben Sie eine Methode `weiter()`, die den jeweils nächsten Ampelzustand anzeigen lässt, gemäß der deutschen Reihenfolge Rot, Rot-Gelb, Grün, Gelb, Rot. Fügen Sie der VBox einen Button "weiter" hinzu, der diese Methode ruft. Sie können einen `EventHandler<ActionEvent>` oder einen Lambda-Ausdruck verwenden.

Tipp: Nummerieren sie die Ampelzustände durch!

Lösung auf Blatt Nr. _____

Aufgabe 4.(30 Punkte): Schreiben Sie eine ***animierte Klasse Collector***. Ein Collector erhält als Konstruktorparameter ein ***Array von Queues*** als Eingangskanäle und ***eine weitere Queue*** als Ausgangskanal. Er soll reihum Elemente aus den Eingangskanälen lesen und in den Ausgangskanal schreiben. Ist ein Eingang leer, wird null in den Ausgang geschrieben. Die Konstruktor-Signatur ist:

```
public Collector(Queue[] eingangskanaele, Queue ausgangskanal)
```

Queue hat die drei Methoden:

```
public void add(Object), public Object remove()throws Emptyexception, public int size()
```

Lösung auf Seite Nr. _____

Aufgabe 5. (30 Punkte): Am Ende eines Tages lassen Sie alle Ereignisse noch einmal Revue passieren: Es sind ***String-Einträge in einer Liste*** von der Form "guter Einkauf", "schlechtes Wetter", etc. Schreiben Sie eine Methode

```
public String wieWarDerTag(List<String> ereignisse);
```

Sie soll "gut" für einen guten Tag, "ok" für einen ausgeglichenen Tag und "schlecht" für einen schlechten Tag liefern, je nachdem, ob die Menge der guten oder schlechten Einträge in der liste "ereignisse" überwiegt. Es gibt vier besondere Ereignisse, die ***Tagesretter*** "Date"und "Gehaltserhöhung", d.h. der Tag ist dann auf jeden Fall gut, und die ***Tageskiller*** "Shitstorm" und "Unfall", die jeden ***nicht geretten*** Tag schlecht machen.

Folgende Methoden könnten für Sie nützlich sein:

Die String-Methoden `public boolean startsWith(String s), public boolen equals(String s)`

Die List<String>-Methoden `public int contains(String s), public String get(int i), public int size()`

Lösung auf Seite Nr. _____

Beginnen Sie hier mit Ihrer Lösung:

Name: _____

Name: _____

Name: _____

Name: _____