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 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: 04.02.2016 12:00 ( ) letzter Versuch

Streichen Sie bitte eine der Aufgaben 3-5!

A1 20
A2 ~~6~~ 11
A3 35
A4 ~~-~~
A5 35
Summe ~~96~~ 101
Note 1,0 c
5.2.16

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.

(X) Richtig ( ) Falsch 1. Die Anweisung `int[10] liste;` erzeugt ein Objekt namens `liste` vom Typ `int[]` mit 10 leeren Speicherplätzen vom Typ `int`.

Grund: Dabei wird noch Speicherplatz für int[10] reserviert.

(X) Richtig ( ) Falsch 2. Eine Klasse kann mehrere Konstruktoren haben, wenn sie alle ***dieselbe Anzahl*** und ***dieselben Typen*** von Parametern haben. Konstruktoren heißen alle gleich

Grund: wenn sie alle einen unterschiedlichen Namen besitzen

( ) Richtig (X) Falsch 3. Wenn man einen abstrakten Datentyp (ADT) ***erweitert***, hat man vollen Zugriff auf seine ***private Datenstruktur***. gut

Grund: Die Datenstrukturen ist zwar als „privat" gesetzt, somit nicht einsehbar.

( ) Richtig (X) Falsch 4. Mit `Integer.parseInt( "12345")` wird der String "12345" in einen Zahlenwert umgewandelt. Die Methode `int parseInt(String number)` ist folglich ***in der Klasse String definiert***.

Grund: Sie ist in der Klasse Integer definiert, da man sie durch Integer. aufruft

(X) Richtig ( ) Falsch 5. Weist man einer Referenz auf ein Objekt **null** zu, ***so ist das Objekt verloren*** und wird vom Garbage Collector entsorgt.

Grund: Es sei denn es gibt noch weitere Referenzen auf das Objekt,

(X) Richtig (X) Falsch 6. In der folgenden geschachtelten Schleife wird ***das kleine Einmaleins ausgegeben***:
`for(int i=1;i<10;i++) for(int j=1;j<10;j++) Console.println("j * i = j*i");`

Grund: Es wird das Einmaleins nur bis 9 ausgegeben

( ) Richtig (X) Falsch 7. Mit `catch (Exception exc)` werden ***alle*** aufgetretenen Exceptions gefangen, ***mit Ausnahme*** der RuntimeExceptions.

Grund: Exception ist die Oberklasse von RuntimeException

( ) Richtig (X) Falsch 8. Nach der Deklaration `ArrayList liste = new ArrayList<String>();` kann man in die `liste` ***nur String-Elemente*** eintragen.

Grund: Es führt zu einem Compilefehler, da bei der Deklaration von gut
Liste der generische Typ erwartet wird

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

Grund: Die Parameter a und b werden addiert und in a gespeichert und

( ) Richtig (X) Falsch 10. Die Ausgabe der folgenden Anweisungsfolge ist HALLO.
`String msg="HALLO"; Object fog=msg; msg="TSCHÜSS"; Console.print((String)fog);`
a wird durch „return" zurückgegeben.

Grund: Es wird „TSCHÜSS" ausgegeben

20

Aufgabe 2. (5/10 Punkte) Die unten stehende Klasse realisiert einen LiFo-Container für Object-Elemente.

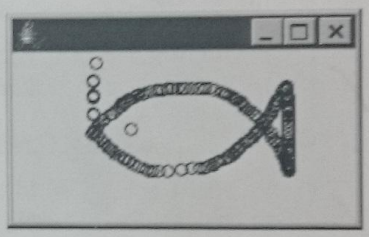
1. Schreiben Sie sie um für String-Elemente (5 Punkte) **ODER**
2. Schreiben Sie sie um in eine generische Klasse, die für beliebige Elementtypen instanziiert werden kann (10 P.)

```
public class Stack {
   List list = new ArrayList ();
   public void add(Object foo)
   {    list.add(0, foo); }
   public Object remove() throws EmptyException
   {    return list.remove(0); }
}
```

Lösung auf Seite Nr. 3

Aufgabe 3. (30 Punkte) Sie haben Scribble in JavaFX programmiert, indem Sie auf ein `Group`-Node ein `Rectangle` gelegt und darauf dann „gezeichnet" haben. Bauen Sie jetzt wieder so ein Group-Node auf und definieren Sie die erforderlichen EventHandler, um darauf mit der Maus eine „Luftblasenspur" zu malen. Das Skelett für die FX-Application finden Sie unten. Verwenden Sie ***möglichst*** Lamda-Ausdrücke für die EventHandler.

```
public class BubbleDraw extends Application {
      Group root = new Group();
      public void start(Stage primaryStage) {
            Scene scene = new Scene(root, 400, 200);
            primaryStage.setScene(scene);
            primaryStage.show();
      }
      public static void main(String[] args) {
            Launch(args);
      }
}
```



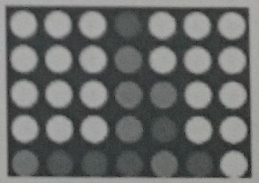
Freiwillige Zusatzaufgabe (5 Punkte): Wechseln Sie bei jeder neuen Blasenspur die Farbe.

Lösung auf Blatt Nr. 4

Aufgabe 4. (30 Punkte): Schreiben Sie eine ***animierte Klasse FerryLoader,*** die den Einweiser in den Wartebereich einer Autofähre simuliert. Ein FerryLoader verwaltet 5 Warteschlangen für Fahrzeuge vom Typ ArrayList<Vehicle>, die ihm als 5 Konstruktorparameter übergeben werden. Bei jedem Arbeitsschritt entnimmt er ein Fahrzeug aus der Eingangsschlange und weist es einer der 4 Ausgangsschlangen zu, je nachdem, ob es höher oder niedriger als 1,80 m ist und ob es gebucht hat. Die Klasse Vehicle bietet dafür die Methoden **`double getHeight()`** und **`boolean isBooked();`**

Lösung auf Seite Nr. _____

Aufgabe 5. (30 Punkte): Sie kennen das Spiel „Vier gewinnt": Zwei Spieler lassen in die Spalten eines 7x5-er Rahmen abwechselnd Chips fallen: Wer zuerst 4 Chips hintereinander in einer Reihe, Spalte oder Diagonale hat, hat gewonnen.



Schreiben sie eine Klasse, die die Datenstruktur für das Spiel darstellt. Sie soll folgende Methoden definieren:

```
public void addChip(int player, int column) throws ColumnFullException; // Spieler 1 und 2
```

```
pubic boolean isFull(int column);
pubic int nextPlayer(); //Spieler 1 und 2, Spieler 1 beginnt.
```

Freiwillige Zusatzaufgabe (5 Punkte):
Schreiben Sie ***eine*** der folgenden Prüfmethoden, die prüfen, ob ***das oberste Chip*** einen Vierer komplettiert:

```
public boolean verticalFour(int column); // prüft auf vertikalen Vierer
public boolean horizontalFour(int column); // prüft auf horizontalen Vierer
public boolean diagonalFour(int column); // prüft auf diagonalen Vierer
```

Lösung auf Seite Nr. 5/6

Beginnen Sie hier mit Ihrer Lösung:

Name: _____

Aufgabe 2)

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

Super! 10+1

Name: _____

3)

```
n:
  Color c = ~~new~~ Color.RED();
  Rectangle r = new Rectangle(0,0,400,200, Color.WHITE);
  r.setOnMousePressed(e -> {
      if(c == Color.RED) c = Color.BLUE();
      else if(c == Color.BLUE()) c = Color.GREEN();
      else if(c == Color.green()) c = Color.RED();
      // gegebenenfalls noch weitere Farben
  });

  r.setOnMouseDragged(e -> {
      Circle ci = new Circle(e.getX(), e.getY(), 5);
      ci.setStroke(c);
      ci.setFill(null);
      r.getChildren().add(ci);
  });

  root.getChildren().add(r);
```

Super! 30+5

Name: _____

5)

```
public class VierGewinntData {
    int[][] field = new int[7][5];
    int currentPlayer = 1;

    public int nextPlayer() {
        currentPlayer = (currentPlayer == 1) ? 2 : 1;
        return currentPlayer;
    }

    public boolean isFull(int column) {
        for(int i=0; i < field[column].length; i++) {
            if( field[column][i] == null)
                return false;
            else if (i == field[column].length -1)
                return true;
        }
    }

// weiter auf der nächsten Seite
```

hier hätte return field[column][4]!=null genügt ☺

Name: _____

```
public void addChip(int player, int column){

    for(int i=0; i<field[column].length; i++){
        if(field[column][i]==null){
            field[column][i]=player;
            i=field[column].length; // oder return;
        } else if(i==field[column].length-1)
            throw new ColumnFullException();
    }
}

public boolean vertialFour(int column){
    for(int i=0; i<field[column].length; i++){
        if(i==0) int player=field[column][i];
        if(field[column][i]==player && player!=0 && player!=null){
            count++;
            if(count==3) return true;
        } else player=field[column][i];
        if(i==field[column].length-1) return false;
    }
}
```

~~if(i==field[column][i])~~

sehr aufwändig. es geht nur um die letzten 4.
if ~~field[column][top]~~(top<3) return;
dann nur die nächsten 4 prüfen.