PRG 2 — Klausur

Name [redacted]

Die Klausur ergibt max. 100 Punkte: **Streichen Sie eine der Aufgaben 2-4!** 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: 23.01.2017 12:15 ( ) **letzter Versuch**

Streichen Sie bitte eine der Aufgaben 2-4!

A1	40
A2	30
A3	30
A4	-
Summe	100
Note	1,0

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. Ein push überschreibt die Dateien im remote-repository. Daher muss man immer vorher eine Sicherheitskopie machen.

Grund: Ein push speichert zwar den aktuellen Stand in dem Repo, die vorherigen Bearbeitungen sind aber immer im Log gespeichert 4

( ) Richtig (X) Falsch — 2. Bei einer Rekursion darf eine Methode sich selbst nur einmal aufrufen.

Grund: wenn gewünscht, kann sie sich beliebig oft aufrufen (C im Rahmen des Speichers beliebig) * 4

( ) Richtig (X) Falsch — 3. Ein Failure wird in JUnit ausgegeben, wenn das Programm an dieser Stelle eine Exception werfen würde.

Grund: Failure und Exceptions sind nicht gleich in JUnit 4

(X) Richtig ( ) Falsch — 4. Anstatt in einer HashMap kann man Objekte auch stets in einer ArrayList speichern.

Grund: Sie sind dann nicht über einen Key erreichbar 4

(X) Richtig ( ) Falsch — 5. Ein Key-Value-Pair ermöglicht es, Felder eines Arrays über einen Begriff (Key) anzusprechen.

Grund: In Fall von Java handelt es sich um eine Hash Map 4

( ) Richtig (X) Falsch — 6. Variablen einer Datenhaltungsklasse werden als public deklariert, damit die Getter- und Setter auf sie zugreifen können.

Grund: Falsch, getter und setter werden implementiert, damit die Attribute nicht public sein müssen 4

( ) Richtig (X) Falsch — 7. Um FXML-Ressourcen verwenden zu können, reicht es aus, diese in den Projektordner zu kopieren.

Grund: Sie müssen zusätzlich auch im Controller mit der korrekten ID instanziiert werden 4

( ) Richtig (X) Falsch — 8. Mit der Methode `assertTrue(value > 20 && value < 80)` wird geprüft, ob der Wert von value den Wert 50 mit einer Toleranzgrenze von +/- 30 hat.

Grund: value muss echt größer 20 und echt kleiner 80 sein 4
daher ist die Toleranz +/- 29

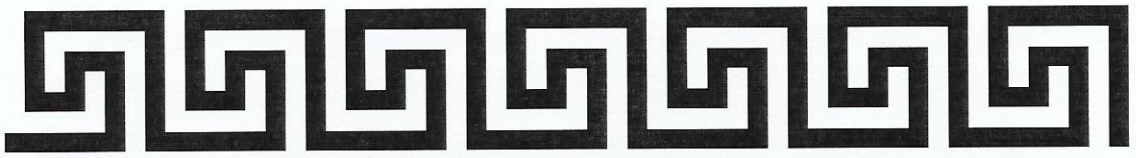
(X) Richtig ( ) Falsch — 9. Inhalte von Objekten werden aus dem Speicher entfernt, wenn das Programm beendet wurde.

Grund: Die Speicherbereiche im RAM sind nach Laufzeitende wieder überschreibbar 4

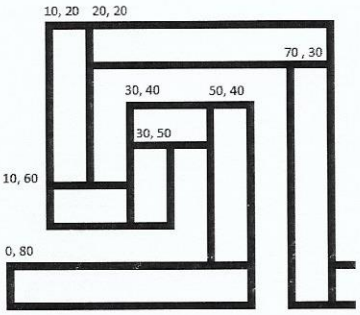
( ) Richtig (X) Falsch — 10. Unter einem branch versteht man in git ein Repository das noch nicht gepusht wurde.

Grund: Ein Branch ist ein ausgezweigter Arbeitsstand der nicht auf das laufende System Auswirkungen hat 4

Aufgabe 2. (30 Punkte) Schreiben Sie eine rekursive Methode, welche ein Mäandermuster (siehe Skizze 1) ausgibt. Das Muster wird mit 8 gefüllten schwarzen Rechtecken (siehe Skizze 2) gezeichnet und dann rekursiv fortgesetzt.



Skizze 1: Mäandermuster



Skizze 2: Aufbau des Mäandermusters mit Rechtecken. Die Zahlen geben die Koordinaten der Rechtecke (links-oben) an

Um die Rekursion übersichtlich zu halten implementieren Sie die Hilfsmethoden

My
```
Rectangle rectangle(int x, int y, int width, int height) und
void draw(Node node)
```

Die Parameter x und y der Methode rectangle sind Koordinaten von links-oben des Rechtecks. Die Methode draw übergibt das Rechteck an die Szene. Die Rekursion wird abgebrochen, wenn das Muster so oft wiederholt wurde, wie Ihr Vorname Buchstaben hat.

Lösung auf Seite Nr. 3, 4, 5

Aufgabe 3. (30 Punkte) In einem Adressbuch werden von einer Person Name, Vorname, Telefonnummer und Emailadresse erfasst. Geben Sie die Implementierung für folgendes Szenario an:

Bei der Instanziierung von Person werden dem Konstruktor Name und Vorname übergeben. Setzen Sie als Werte hierfür Ihren Namen ein.
Anschließend werden Telefon und Emailadresse über die Zugriffsmethoden übergeben.
Das Objekt wird dem Adressbuch hinzugefügt. Über die Methode getString werden alle Inhalte des Adressbuchs ausgegeben.

Lösung auf Seite Nr. 5, 6

Aufgabe 4. (30 Punkte): In einer Playlist werden zu den einzelnen Songs folgende Daten: Titel, Komponist, Album, Dauer. Geben Sie die Implementierung für folgendes Szenario an:

Eine Methode prüft, ob die Inhalte zweier Song-Objekte gleich sind.
Alle Song-Objekte werden in einer CSV-Datei gespeichert.
In die Playlist können alle Songs über eine CSV-Datei eingelesen werden.

Lösung auf Seite Nr. _____

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]

```
My
class Rectangle extends Rectangle {
// Ich überschreibe Rectangle, da ich nicht sicher
// bin, dass der Konstruktor wie angegeben existiert

    My Rectangle (int x, int y, int width, int height) {
        super( width, height );
        this.setX(x);
        this.setY(y);

    }

}

class Maeander extends Application {

    //main aus Zeitgründen weggelassen
    Group root;
    //
    public static void start (Stage PrimaryStage) {
        // Scene und Group initialisieren und zeigen
        // aus Zeitgründen weggelassen
        drawRecursive(0);
        stage.show();
    }
                                                    ,1
```

Name: _____

```
int depth
private static void drawRecursive(int startX) {
*  Rectangle rect1 = new MyRectangle(startX, 80, 60, 10);
   draw(rect1);
   Rectangle rect2 = new MyRectangle(startX+50, 40, 10, 40);
   draw(rect2);
   Rectangle rect3 = new MyRectangle(startX+30, 40, 20, 10);
   draw(rect3);
   Rectangle rect4 = new MyRectangle(startX+30, 50, 10, 20);
   draw(rect4);
   Rectangle rect5 = new MyRectangle(startX+10, 60, 20, 10);
   draw(rect5);
   Rectangle rect6 = new MyRectangle(startX+10, 20, 10, 40);
   draw(rect6);
   Rectangle rect7 = new MyRectangle(startX+20, 20, 60, 10);
   draw(rect7);
   Rectangle rect8 = new MyRectangle(startX+70, 30, 10, 60);
   draw(rect8);

   drawRecursive(startX+80, depth+1);
}

*  if(depth > "[redacted] length()) {
      return;
   }
```

Name: _____

```
private static void draw(Node node){
    root.getChildren.add(node);
}
```

```
public class Person {
    Person(String Name, String Vorname) {
    this.name = name;
    this.vorname = vorname;
    }
private String Name;
private String Vorname;
private String eMail = null;
private String Telefon = null //als String um Leerzeichen und +49etc zuzulassen.
    public String getName( ){
      return name; }
    public String getVorname( ){
      return vorname; }
    public String getEMail ( ) {
      return eMail; }
    public String getTelefon( ) {
      return Telefon; }
    public void setEMail (String eMail) {
      this.eMail = eMail; }
    public void setTelefon ( String telefon) {
      this.telefon = telefon; }
...}
```

Name: _____

```
public class Adressbuch {
  Arraylist<Person> adressbuch = new Arraylist<>();
  public void add (Person person) {
    adressbuch.add (person); }

  public String getString ( ) {
  String result = "";
    for (Person person : adressbuch) {
      result += person.getVorname();
      result += "\t";
      result += person.getNachname( );
      result += "\t";
      result += person.getTelefon( );
      result += "\t";
      result += person.getEMail ( );
      result += "\n";

    }
    return result;
  }
public class Main {
  public static void main (String[ ] args) {
    Person ich = new Person ("[redacted]");
    ich.setEmail ("[redacted]");
    ich.setTelefon ("[redacted]");
    Adressbuch buch = new Adressbuch();
    buch.add (ich);
    Buch
    System.out.println(buch.getString());
    }
}
```