University of Zurich
Department of Informatics

s.e.a.l.
software evolution & architecture lab

Programmierung für Mathematik HS10

Übung 5

1 Aufgabe: Eclipse IDE

1.1 Lernziele

1. Die Entwicklungsumgebung Eclipse kennen lernen.

1.2 Aufgabenstellung

1. Installieren Sie die IDE[1] Eclipse (wählen Sie Eclipse Classic).
2. Arbeiten Sie das zur Verfügung gestellte Eclipse-Tutorial durch.
3. Legen Sie für die nachfolgenden Aufgaben ein neues Eclipse Projekt an und geben Sie ihm einen sinnvollen Namen. Nutzen Sie Java Packages, um die einzelnen Teilaufgaben zu organisieren.

[1] *IDE* steht für *Integrated Development Environment*

2 Aufgabe: Rechnen mit Matrizen

2.1 Lernziele

1. Mit zweidimensionalen Arrays rechnen können.

2.2 Aufgabenstellung

Eine Matrix ist ein rechteckiges Schema

$$\begin{pmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m1} & \alpha_{m2} & \cdots & \alpha_{mn} \end{pmatrix}$$

(in dem Fall m Zeilen und n Spalten, man sagt auch, das sei eine $m \times n$ Matrix). Dabei ist $\alpha_{ij} \in \mathbb{R} \quad \forall i \in \{1, ..., m\}, j \in \{1, ..., n\}$. In Java kann eine Matrix durch ein zweidimensionales Array dargestellt werden.

1. Schreiben Sie eine Methode `printMatrix` welche als Übergabeparameter eine Matrix A erwartet (also ein zweidimensionales Array) und diese auf der Kommandozeile ausgibt.
2. Schreiben Sie eine Methode `calculateTranspose` welche als Übergabeparameter eine Matrix A erwartet und die transponierte Matrix A^T zurückgibt.
   Bemerkung: Sei
   $$A = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m1} & \alpha_{m2} & \cdots & \alpha_{mn} \end{pmatrix}$$
   dann ist die transponierte Matrix A^T wie folgt definiert:
   $$A^T = \begin{pmatrix} \alpha_{11} & \alpha_{21} & \cdots & \alpha_{m1} \\ \alpha_{12} & \alpha_{22} & \cdots & \alpha_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{1n} & \alpha_{2n} & \cdots & \alpha_{mn} \end{pmatrix}$$
3. Schreiben Sie eine Methode `add` welche die Summe von zwei Matrizen berechnen soll, falls diese definiert ist. Andernfalls soll eine Fehlermeldung ausgegeben werden.
   Bemerkung: die Summe von zwei $m \times n$ Matrizen A, B ist komponentenweise definiert, das heisst:
   $$A + B = \begin{pmatrix} \alpha_{11} & \cdots & \alpha_{1n} \\ \vdots & \ddots & \vdots \\ \alpha_{m1} & \cdots & \alpha_{mn} \end{pmatrix} + \begin{pmatrix} \beta_{11} & \cdots & \beta_{1n} \\ \vdots & \ddots & \vdots \\ \beta_{m1} & \cdots & \beta_{mn} \end{pmatrix} = \begin{pmatrix} \alpha_{11} + \beta_{11} & \cdots & \alpha_{1n} + \beta_{1n} \\ \vdots & \ddots & \vdots \\ \alpha_{m1} + \beta_{m1} & \cdots & \alpha_{mn} + \beta_{mn} \end{pmatrix}$$
   (falls die Dimensionen von A und B nicht übereinstimmen, ist die Summe nicht definiert.)
4. Schreiben Sie eine Methode `multiplicate`, welche das Produkt von zwei Matrizen berechnen soll, falls es definiert ist und sonst eine entsprechende Fehlermeldung ausgibt.

Bemerkung: Die Matrixmultiplikation $A \cdot B$ ist für eine $n \times k$ Matrix A und eine $k \times m$ Matrix B wie folgt definiert:

$$A \cdot B = \begin{pmatrix} \alpha_{11} & \cdots & \alpha_{1k} \\ \vdots & \ddots & \vdots \\ \alpha_{n1} & \cdots & \alpha_{nk} \end{pmatrix} \cdot \begin{pmatrix} \beta_{11} & \cdots & \beta_{1m} \\ \vdots & \ddots & \vdots \\ \beta_{k1} & \cdots & \beta_{km} \end{pmatrix} = \begin{pmatrix} \gamma_{11} & \cdots & \gamma_{1m} \\ \vdots & \ddots & \vdots \\ \gamma_{n1} & \cdots & \gamma_{nm} \end{pmatrix}$$

wobei

$$\gamma_{ij} = \sum_{l=1}^{k} (\alpha_{il} \cdot \beta_{lj}) \quad \forall i \in \{1, ..., n\}, j \in \{1, ..., m\}$$

(falls die Anzahl Spalten von A nicht mit der Anzahl Zeilen von B übereinstimmt, ist $A \cdot B$ nicht definiert)

3 Aufgabe: OOP und Information Hiding

3.1 Lernziele

1. Ein Problem von natürlicher Sprache in Java Code übersetzen können.
2. Eine bestehende Klasse intern abändern und dabei die Schnittstellen beibehalten können.

3.2 Aufgabenstellung

a) Student

1. Ein Student hat folgende Eigenschaften:
   - *Name*
   - *Alter* (um an der Universität zu studieren muss man mindestens 18 sein)
   - *Matrikelnummer*

   Schreiben Sie eine Klasse `Student`, welche diese Ausgangslage in Java abbildet. Es soll möglich sein den Namen und das Alter direkt bei der Initialisierung oder auch noch nachträglich zu setzen (Tipp: dazu muss ein Konstruktor geschrieben werden). Allerdings soll der Name nur einmal gesetzt werden können und bevor das Alter gesetzt wird, soll überprüft werden, ob das Alter gültig ist. Wird ein ungültiges Alter übergeben, soll eine Meldung ausgegeben und das Alter auf -1 gesetzt werden. Die Matrikelnummer soll automatisch bei jeder Initialisierung eines `Student` Objektes vergeben werden und zwar so, dass zwei Studenten nie dieselbe Nummer haben. Der erste Student bekommt die Nummer 1, der zweite die Nummer 2 etc. (Tipp: verwenden Sie eine statische Variable).
   Implementieren Sie die Methode `toString()`, welche einen sinnvollen `String` mit Name, Alter und Matrikelnummer zurückgeben soll. (Tipp: Lesen Sie in der Java-API nach, wie die Methode `toString` von der `Integer` Klasse verwendet werden kann.)

2. Schreiben Sie eine Klasse `University`, welche die Studenten der Universität speichern soll. Benutzen Sie dazu als Instanzvariable ein Array vom Typ `Student[]`. Die Sekretärin soll die Möglichkeit haben einen Studenten zu dieser Liste hinzuzufügen und zwar so, dass die Matrikelnummern der Studenten in dem Array automatisch aufsteigend geordnet sind. Ausserdem soll sie auch die Möglichkeit haben das Alter aller Studenten um ein Jahr zu erhöhen (dies tut sie einfachheitshalber nur einmal pro Jahr, für alle Studenten gleichzeitig).

3. Schreiben Sie einen `TestDriver`, in welchem Sie die beiden Klassen testen. (Um die Klassen richtig testen zu können, ist es sinnvoll, die `University` Klasse um die Methode `printStudents` zu ergänzen, welche alle Studenten auf der Kommandozeile ausgibt.)

b) Anpassen der `Student`-Klasse

Die Sekretärin beklagt sich bei den Informatikdiensten über dieses System, weil sie jedes Jahr das Alter der Studenten anpassen muss. Passen Sie nun die Klasse `Student` so an, dass fortan anstelle des Alters das Geburtsjahr der Studenten gespeichert wird. Implementieren Sie die passenden Accessor-Methoden `getYearOfBirth()` und `setYearOfBirth(...)`. Da die Universität aber noch viele alte Programme am Laufen hat, die die Klasse `Student` verwenden, müssen die Zugänge über die alten Accessor-Methoden (`setAge(...)` und `getAge()`) unbedingt gewährleistet bleiben.

Fügen Sie Ihrer Klasse folgende Hilfsmethode hinzu. Danach erhalten Sie innerhalb der Klasse mit `getCurrentYear()` das jeweils aktuelle Jahr als Integer.

```
1   private int getCurrentYear() {
2       return (new java.util.GregorianCalendar()).get(java.util.
            GregorianCalendar.YEAR);
3   }
```

Testen Sie Ihre abgeänderte Klasse mit dem `TestDriver` aus der ersten Teilaufgabe und überprüfen Sie damit, ob die Schnittstellen gleich geblieben sind.

4 Aufgabe: GUI – First Window

4.1 Lernziele

1. GUI-Elemente kennenlernen.
2. Bestehenden Code verwenden und anpassen können.

4.2 Aufgabenstellung

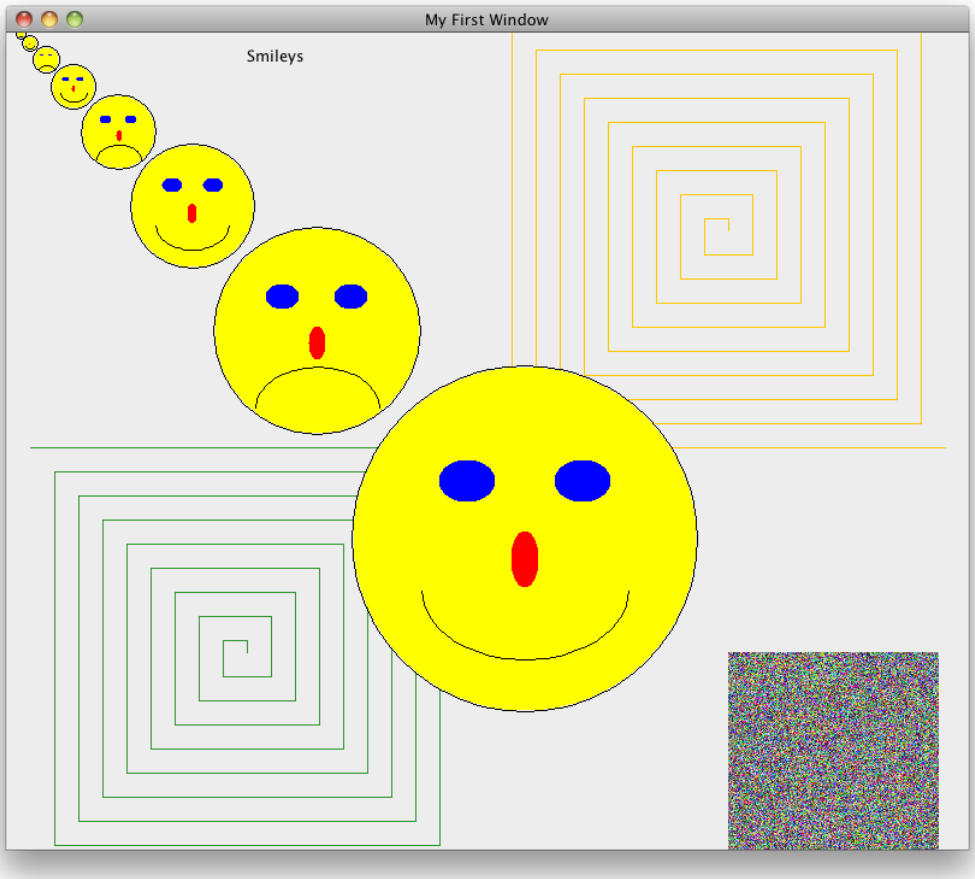
Verwenden Sie für diese Aufgabe die zur Verfügung gestellten Klassen-Files `Gui.java` und `GuiTest.java`.[2]

1. Importieren Sie die beiden Files in Eclipse und führen Sie den TestDriver aus.
2. Schauen Sie sich die Klasse `Gui` und deren `show()`-Methode an. Darin werden u.a. drei Hilfsmethoden aufgerufen, um Muster zu erzeugen: `drawRandomColorSquarePattern`, `drawGreekPattern` und `drawSmiley`. Das Ziel ist, diese so zu vervollständigen, dass in etwa dasselbe Bild(siehe unten) herauskommt.
3. In der `drawRandomColorSquarePattern`-Methode geht es darum, im durch die Übergabewerte vorgegebenen Quadrat für jedes Pixel eine zufällige Farbe zu setzen. Ein einzelnes Pixel (x, y) in einem `BufferedImage image` verändern Sie farblich wie folgt – r, g und b stehen dabei für die Rot-, Grün- und Blauanteile und können Werte von 0 bis 255 annehmen.

```
1   Color color = new Color(r, g, b);
2   image.setRGB(x, y, color.getRGB() );
```

4. Schauen Sie sich für die beiden anderen Methoden die `Graphics`-Klasse – insbesondere die im Methodenrumpf angegeben Methoden – in der Java-API an und vervollständigen Sie sie ebenfalls, indem Sie sich an existierendem Code orientieren.
5. Probieren Sie weitere Dinge aus; Verändern Sie die `show`-Methode, um andere Muster zu erzeugen.

[2] Diese finden Sie entweder direkt auf der Übungsseite oder im OLAT.


My First Window
Smileys