Universität Augsburg
Institut für Informatik

Deckblatt: Übung zur Vorlesung Informatik 2 SS 2018
Fakultät für Angewandte Informatik
Lehrprofessur für Informatik
Prof. Dr. Lorenz, Marius Brendle, Johannes Metzger

Hinweis: Es sind alle Felder auszufüllen! Abgabe der Übungsblätter immer **mittwochs** (Ausnahme wenn Feiertag: donnerstags) bis **spätestens 12:00 Uhr** in die entsprechend gekennzeichneten Briefkästen der Veranstaltung im Erdgeschoss des Instituts für Informatik (Gebäude N). Zuwiderhandlung wird mit Strafe geahndet! (Punktabzug)

Übungsblatt	

(hier die Nummer des bearbeiteten **Übungsblatts eintragen**)

	Übung 01 (1057 N) Montag 08:15 - 09:45 Uhr (Isabell Rücker)
	Übung 02 (1056 N) Montag 14:00 - 15:30 Uhr (Henning Cui)
	Übung 03 (1057 N) Montag 15:45 - 17:15 Uhr (Josef Kircher)
	Übung 04 (1054 N) Montag 17:30 - 19:00 Uhr (Mosaab Slimani)
	Übung 05 (1057 N) Montag 17:30 - 19:00 Uhr (David Hacker)
	Übung 06 (1055 N) Dienstag 12:15 - 13:45 Uhr (André Schweiger)
X	Übung 07 (1054 N) Dienstag 17:30 - 19:00 Uhr (Benjamin Sertolli)
	Übung 08 (1057 N) Dienstag 17:30 - 19:00 Uhr (Dat Le Thanh)
	Übung 09 (1054 N) Mittwoch 08:15 - 09:45 Uhr (Erik Pallas)
	Übung 10 (1055 N) Mittwoch 08:15 - 09:45 Uhr (Moritz Feldmann)
	Übung 11 (1054 N) Mittwoch 10:00 - 11:30 Uhr (Denise Böhm)
	Übung 12 (1056 N) Donnerstag 08:15 - 09:45 Uhr (Florian Magg)
	Übung 13 (1054 N) Donnerstag 15:45 - 17:15 Uhr (Marvin Drexelius)
	Übung 14 (1054 N) Donnerstag 17:30 - 19:00 Uhr (Patrick Eckert)
	Übung 15 (1057 N) Donnerstag 17:30 - 19:00 Uhr (Alexander Szöke)
	Übung 16 (1057 N) Freitag 08:15 - 09:45 Uhr (Philipp Braml)
	Übung 17 (1054 N) Freitag 10:00 - 11:30 Uhr (Elisabeth Korndörfer)
	Übung 18 (1054 N) Freitag 12:15 - 13:45 Uhr (Philipp Häusele)
	Übung 19 (1056 N) Freitag 12:15 - 13:45 Uhr (Maximilian Demmler)
	Übung 20 (1054 N) Freitag 14:00 - 15:30 Uhr (Florian Straßer)

(hier die eingeteilte **Übungsgruppe ankreuzen**)

Teamnummer	6

(hier die Nummer des eingeteilten **Teams eintragen**)

Tarik Selimovic
Anton Lydike
Dominic Cesnak

(hier die **Vor- und Nachnamen** aller Teammitglieder **eintragen**)

Aufgabe		
Aufgabe		
Aufgabe		
Aufgabe		
Gesamt		(vom Tutor auszufüllen)

Übungsblatt 7

25)

History)

```
package aufgabe25;

import java.awt.*;
import java.awt.event.*;

public class History extends Dialog {
    private static final long serialVersionUID = 1L;

    public History (Frame f, java.util.ArrayList<Double> list) {
        super(f, "History", true);

        FlowLayout fl = new FlowLayout();
        List l = new List(10);

        for (int i = list.size(); i > 0; i--) {
            l.add(String.valueOf(list.get(i-1)));
        }

        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                dispose();
            }
        });

        this.setLayout(fl);
        this.add(l);
        this.pack();
        this.setVisible(true);
    }
}
```

Window)

```
package aufgabe25;

import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class Window extends Frame implements ActionListener {
    private static final long serialVersionUID = 1L;
    private java.util.ArrayList<Double> history = new ArrayList<Double>();
    private Button btn_rn = new Button("random");
    private Button btn_hs = new Button("history");
    private TextField tf  = new TextField();

    public Window() {
        super("Aufgabe25");
        FlowLayout fl = new FlowLayout();
        this.add(btn_rn);
        this.add(tf);
        this.add(btn_hs);
        btn_rn.addActionListener(this);
        btn_hs.addActionListener(this);

        tf.setPreferredSize(new Dimension(150, 16));

        this.setLayout(fl);

        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                dispose();
            }
        });

        this.pack();
        this.setVisible(true);
    }

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

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource().equals(this.btn_rn)) {
            generateRandom();
        } else if (e.getSource().equals(this.btn_hs)) {
            showHistory();
        }
    }

    private void generateRandom() {
        double r = Math.random();
        this.history.add(r);
        this.tf.setText(String.valueOf(r));
    }

    private void showHistory() {
        new aufgabe25.History(this, this.history);
    }
}
```

26)

a)

```
CREATE TABLE Patient (
  id INTEGER NOT NULL,
  PRIMARY KEY(id)
);

CREATE TABLE Arzt (
  id INTEGER NOT NULL,
  PRIMARY KEY(id)
);

CREATE TABLE Behandlung (
  id INTEGER NOT NULL,
  diagnose VARCHAR(200) NOT NULL,
  arztID INTEGER NOT NULL,
  patientID INTEGER NOT NULL,
  FOREIGN KEY (arztID) REFERENCES Arzt(id),
  FOREIGN KEY (patientID) REFERENCES Patient(id),
  PRIMARY KEY (id)
);
```

b)

```
CREATE TABLE Arzt (
  isbn CHAR(10) NOT NULL UNIQUE,
  jahr INT,
  autor VARCHAR(20),
  PRIMARY KEY(isbn)
);
```

27)

a)

```
CREATE TABLE Point (
  x FLOAT NOT NULL,
  y FLOAT NOT NULL,
  id INTEGER NOT NULL AUTO_INCREMENT,
  PRIMARY KEY(id)
);

CREATE TABLE GeometricObject (
  position INTEGER NOT NULL,
  type ENUM('circle', 'rectangle') NOT NULL,
  -- circle
  radius FLOAT,
  check (
    type = 'circle' AND
    radius IS NOT NULL AND
    radius > 0
  ),
  -- rectangle
  length FLOAT,
  width FLOAT,
  check (
    type = 'rectangle' AND
    length IS NOT NULL AND
    width IS NOT NULL AND
    length > 0 AND
    width > 0
  ),
  FOREIGN KEY (position) REFERENCES Point(id)
);

-- PRO: Erweitern der Oberklasse GeometricObject ist einfach

-- CON: Unterscheidung zwischen objekttypen ist schwer, besonders wenn
--      eine andere Klasse nur referenzen auf Kreise zulassen möchte

-- CON: Wenn eine weitere Klasse von GeometricObject erbt, können backups nicht
--      mehr eingelesen werden, da GeometricObjects struktur geändert wurde

-- CON: Die Constraints werden schnell unübersichtlich
```

b)

```
CREATE TABLE Point (
  x FLOAT NOT NULL,
  y FLOAT NOT NULL
)

CREATE TABLE Circle (
  position Point not null,
  radius FLOAT NOT NULL,
  check (radius > 0)
)

CREATE TABLE Rectangle (
  position Point not null,
  length FLOAT NOT NULL,
  width FLOAT NOT NULL,
  CHECK ( length > 0 ),
  CHECK ( width > 0 )
)

-- PRO: Operationen auf der Datenbank sind recht einfach

-- CON: Erweitern der oberklasse GeometricObject wird komplex wenn mehr
--      unterklassen hinzugefügt werden.
```

28)

a)

```
INSERT INTO Point VALUES (5, 0, 0);

UPDATE Circle SET position = 5 WHERE id = 7;
```

b)

```
-- insert placeholder customer (do this only once)
INSERT INTO Kunde (id) VALUES (-1);

-- make sure the cutsomer isn't mentioned in a foreign key
UPDATE Ausleihe SET kundeID = -1 WHERE kundeID = 123;

-- delete customer
DELETE FROM Kunde WHERE id = 123;

---
SELECT DISTINCT kundeID FROM Ausleihe WHERE begin = '2017-12-20';
```