

# Gruppe 08



## Sitzung 10

18.05.2020

# Begrüßung



# Programm für die 10. Sitzung

Begrüßung

Organisation und Fragen

Review der Diagramme von Gruppe 04

Präsentation

JavaFX - Tabellen und Listen

Aufgaben für die nächste Sitzung

Sonstiges?

# Organisation und Fragen



# JavaDoc + Implementierung

- Bis Freitag abgeschlossen
- Prüft, ob ihr den Autor bei der Klassenbeschreibung habt

# Zeitplan

Datum	Uhrzeit	Präsentation bzw. Abgabe
Do. 30. April 2020		Abgabe des <u>Anforderungsmodells</u>
Mi. 6. Mai 2020	10:00 - 12:00	Präsentation des <u>Anforderungsmodells</u>
Mo 11. Mai 2020		Abgabe der <u>Produktbeschreibung</u>
Di. 19. Mai 2020		Abgabe der <u>Sequenzdiagramme</u> und des <u>Strukturmodells</u>
Mi. 20. Mai 2020	10:00 - 12:00	Präsentation des <u>Analysemodells</u>
Mo. 25. Mai 2020		Abgabe des <u>Designmodells</u>
Fr. 29. Mai 2020		Abgabe der <u>Klassentests</u>
Mi. 3. Juni 2020	10:00 - 12:00	Abschlusspräsentation des fertigen Produkts

# Zeitplan der Präsentation

- 10:00 Uhr: Treffen in unserem BBB-Raum, Review der Diagramme von Gruppe 04
- 10:45 Uhr: Präsentation von Gruppe 04 und 08
  - Strukturmodell
  - 2-3 Sequenzdiagramme
  - 5 Minuten für Fragen
- 11:30 Uhr: Besprechung der Rückmeldung von Gruppe 04 und Planung des weiteren Vorgehens
- Frage: Wer möchte mit Felix präsentieren?

# Review der Diagramme von Gruppe 04





# Fragen

1. Welches Diagramm von welcher Gruppe wurde begutachtet?
2. Ist das Diagramm informativ? Enthält es alle Details, die Sie erwartet hätten? Wenn es nicht informativ ist, was fehlt Ihrer Meinung nach?
3. Ist das Diagramm übersichtlich gestaltet bzw. strukturiert? Enthält das Diagramm unnötige Kreuzungen? Sind Komplexität und Größe angemessen?
4. Sind alle in der Aufgabenstellung aufgeführten Details im Diagramm umgesetzt?
5. Gibt es syntaktische/semantische Fehler im Diagramm? Wenn ja, welche?
6. Was ist besonders gelungen (nicht gelungen)?
7. Gibt es sonstige Anmerkungen und Hinweise, die Sie der Gruppe zu dem Diagramm geben möchten?

# Präsentation

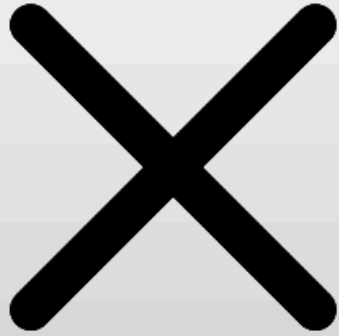
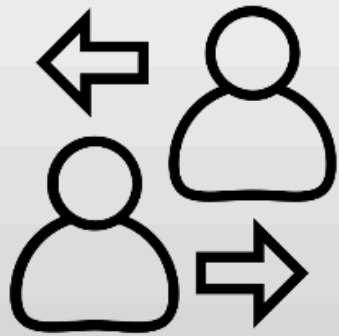




# JavaFX - Tabellen und Listen



# JavaFX - Tabellen und Listen

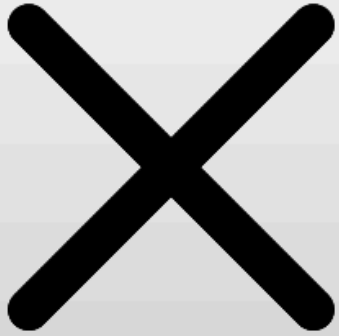
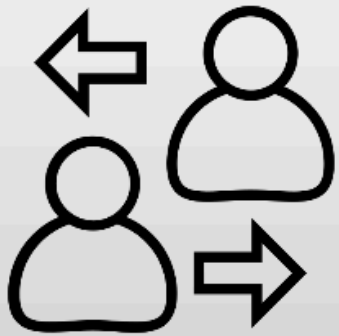


## SGO - Student-Gruppen Organisator



Vorname	Nachname	Matrikelnummer
M.	Muster	123456

**Student**  
Vorname   
Nachname   
Matrikelnummer

## SGO - Student-Gruppen Organisator

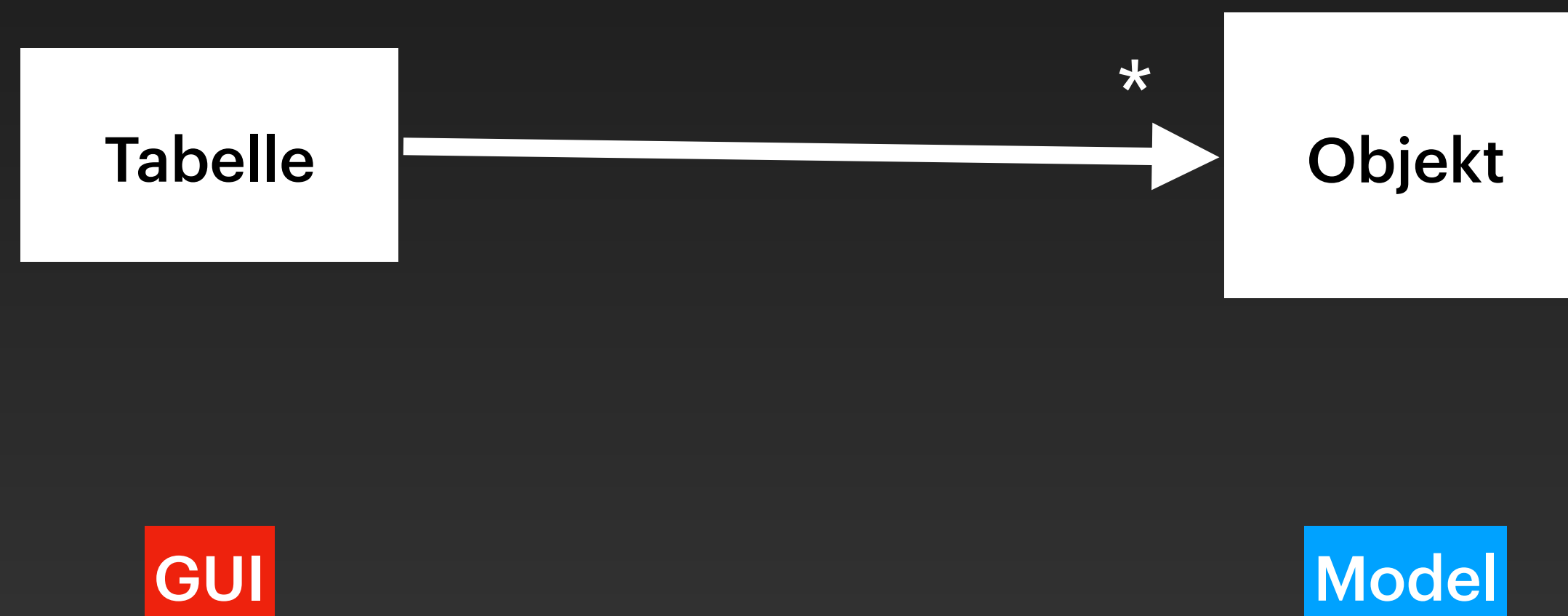


1  
2  
3

Gruppennummer

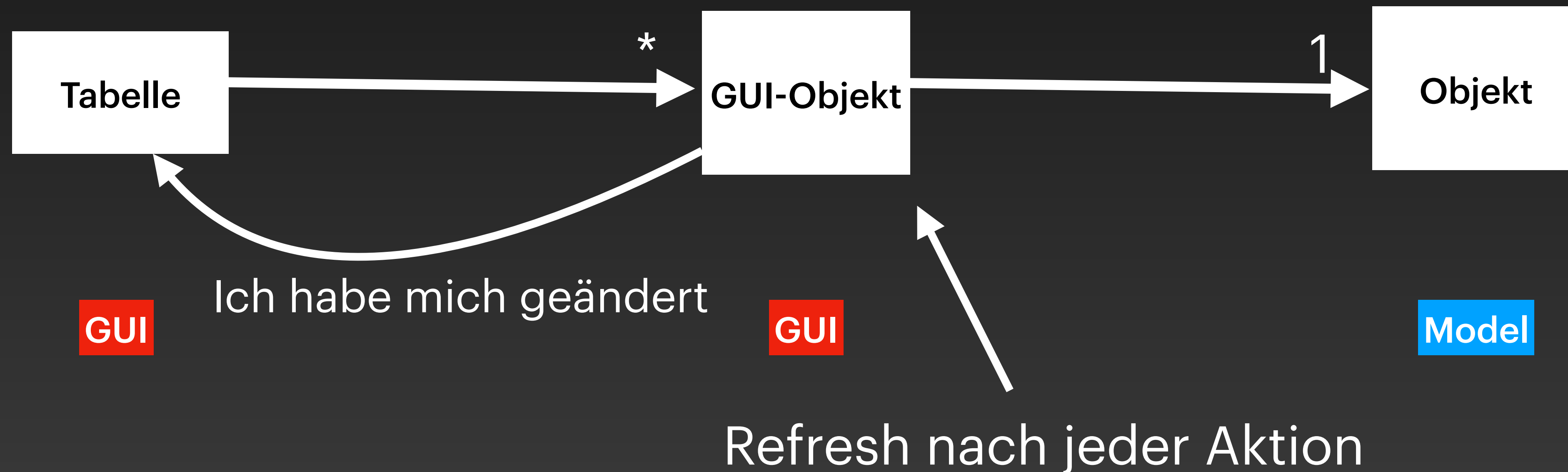
# Tabellen/Listen mit Daten füllen

- Grundsätzlich mehrere Möglichkeiten



# Tabellen/Listen mit Daten füllen

- Grundsätzlich mehrere Möglichkeiten



# Model vs. ViewModel

```
public class Student implements Serializable {  
  
    private String firstName;  
  
    private String lastName;  
  
    private int studentId;  
  
    public Student(String firstName, String lastName, int studentId) {  
        this.firstName = firstName;  
        this.lastName = lastName;  
        this.studentId = studentId;  
    }  
  
    //Getter und Setter für alle Attribute  
    public String getFirstName() { return firstName; }  
  
    public void setFirstName(String firstName) { this.firstName = firstName; }  
  
    public String getLastName() { return lastName; }  
  
    public void setLastName(String lastName) { this.lastName = lastName; }  
  
    public int getStudentId() { return studentId; }  
}
```

```
public class StudentViewModel {  
  
    private Student student;  
  
    private StringProperty firstName;  
    private StringProperty lastName;  
    private StringProperty studentId;  
  
    public StudentViewModel(Student student){  
  
        this.student = student;  
        this.firstName = new SimpleStringProperty();  
        this.lastName = new SimpleStringProperty();  
        this.studentId = new SimpleStringProperty();  
        refresh();  
    }  
  
    public void refresh(){  
        this.firstName.set(student.getFirstName());  
        this.lastName.set(student.getLastName());  
        this.studentId.set(student.getStudentId()+"");  
    }  
  
    public String getFirstName() { return firstName.get(); }  
    public void setFirstName(String fName) { firstName.set(fName); }  
    public StringProperty firstNameProperty() { return firstName; }  
}
```

# ViewModel

- Property-Klassen sind beobachtbar
- Gibt String, Integer, Double, Object etc.
- Ändert sich der Wert, wird die Tabelle/Liste automatisch aktualisiert
- Brauchen für alle Attribute get/set/Property (Achtung bei Parameter-/Rückgabetypen)

```
public class StudentViewModel {  
  
    private Student student;  
  
    private StringProperty firstName;  
    private StringProperty lastName;  
    private StringProperty studentId;  
  
    public StudentViewModel(Student student){  
  
        this.student = student;  
        this.firstName = new SimpleStringProperty();  
        this.lastName = new SimpleStringProperty();  
        this.studentId = new SimpleStringProperty();  
        refresh();  
    }  
  
    public void refresh(){  
        this.firstName.set(student.getFirstName());  
        this.lastName.set(student.getLastName());  
        this.studentId.set(student.getStudentId()+"");  
    }  
  
    public String getFirstName() { return firstName.get(); }  
    public void setFirstName(String fName) { firstName.set(fName); }  
    public StringProperty firstNameProperty() { return firstName; }  
}
```



# ViewModel in der Tabelle

Spalten mit entsprechenden Parametern des ViewModels verknüpfen

```
@FXML
public void initialize() {
    tableColumnStudentsFirstName.setCellValueFactory(new PropertyValueFactory<StudentViewModel, String>("firstName"));
    tableColumnStudentsLastName.setCellValueFactory(new PropertyValueFactory<StudentViewModel, String>("lastName"));
    tableColumnStudentsStudentId.setCellValueFactory(new PropertyValueFactory<StudentViewModel, String>("studentId"));
}
```

# ViewModel in der Tabelle

Neues Objekt in Model und in Tabelle einfügen

Zugriff auf Tabellenliste mit `getItems():ObservableList<>`

```
@FXML
void addStudent(ActionEvent event) {

    StudentController studentController = sgoController.getStudentController();
    try {
        String firstName = textFieldFirstName.getText();
        String lastName = textFieldLastName.getText();
        int studentId = Integer.parseInt(textFieldStudentId.getText());
        Student newStudent = studentController.createStudent(firstName, lastName, studentId);
        StudentViewModel svw = new StudentViewModel(newStudent);
        tableViewStudents.getItems().add(svw);
        svw.refresh();
        clear();
    } catch (StudentAlreadyExistsException | NumberFormatException e) {
        e.printStackTrace();
    }
}
```

# ViewModel in der Liste

Liste mit entsprechendem Parameter des ViewModels verknüpfen

```
@FXML
public void initialize() {
    buttonBarUpdate.setVisible(false);
    listViewGroups.setCellFactory(listview -> {
        return new ListCell<GroupViewModel>() {
            @Override
            public void updateItem(GroupViewModel item, boolean empty) {
                super.updateItem(item, empty);
                textProperty().unbind();
                if (item != null)
                    textProperty().bind(item.groupNumberProperty());
                else
                    setText(null);
            }
        };
    });
}
```

# Aufgabe für die nächste Sitzung



# Aufgaben zu Freitag

- JavaDoc + Implementierung fortsetzen
- FXML-Dateien in Projekt einfügen und Codeskeleton erzeugen
- Erste Funktionalität hinzufügen

# Sonstiges/Fragen?



# Bis Freitag



Held der Programmierung