Home exam (hjemmeeksamen)

PGR112 Object-oriented Programming

June 2023

Home exam, individual

Notice:

- The main topics in this exam are Java, JDBC, and diverse Object-oriented programming theory as given through the course.
- All code that is not from a course related git repository or from any of the lecture slides, must be coded by you, and you alone. If you do use existing code, you must clearly show (in comments) what the source was.
- You should put all the files that the delivery contains in one folder and zip before uploading to WISEFlow.
- The solution should be able to run properly in IntelliJ (i.e. not have to run with any other special technology/IDE)
- If you are unsure about something regarding the exam, re-read the instructions and try to make a best assumption. Clearly express what assumptions you made and why in the report (see below).

The delivery will contain three things:

1. The project folder with Java, folders, txt or json files if any, etc.
2. The database schema files you have created (see SQL files in repository for examples).
3. A small report (pdf or word document) that shortly describes the major OOP concepts and how you have implemented them through your design.

Case – Event management

You are given the task of implementing a management system for an event. For example, the graduation ceremony for a school/university.

Imagine a school that has at least 4 study programs. Each study program has a **Program Responsible** and a number of teachers associated with it. In each study program, there are a number of students. When the graduation ceremony approaches, there needs to be a means of registering for the event and managing the students that will graduate from each program.

Your task is to design a system for managing such an even.

A user can

- Register to attend the event.

  - The user that registers must be a **Student** (that means that you should have a separate database that stores information about each **Student**, and the **Study program** they are a part of).
  - When a student registers, an entry is created in a database to keep track of this event.
  - A student can only register once. If a student attempts to register, but an entry already exists, the user should get a notification of the fact.

- Each student may invite up to 4 people to the ceremony. Each additional **Person** is part of the registration event, as a **Guest** of the Student that registered.

- A user may adjust their registration event. For example, if they have 2 guests, but realized that one more person can actually attend, the student can edit the register event to include their extra guest.

- A user may delete a register event. For example, a student realized that they cannot attend, so they will delete their event. Their Guests are also removed.

- A user may get from the system a list of all participants (all **Program Responsibles**, all **Teachers**, and all **Students** and their **guests**). If several students invite the same **Guest**, the guest will only appear in this list once.

- A user may get from the database a list of all students that will participate (that is, **students** that have registered).

- A user may get from the database a list of all **students** from a particular **program** that have registered for the ceremony.

- A user may search for a particular student (for example by name) and see if that student has registered to attend the ceremony.

- A user may print to screen a program of the ceremony. This will be displayed in console (System.out.) and consist of:

  - Introduction: 30 minutes.
  - For **each** study program – Study program name, Program responsible name – total duration computed as follows:
    - – Program Responsible speech – 1 minute
    - - for each 5 students registered to participate – add 1 minute.
  - A 5-minute break between study programs
  - Closing remarks – 15 minutes.
  - If the user is a Student, the program they are a student in should be marked differently.

What are you to make?

Databases and tables

- You should create (and populate with relevant data) a database for Students, Staff, and Study Programs, called **universityDB**.
    - This is meant to store all the student and study program data, and will not be editable from your program.
    - Only Students can register to attend the graduation ceremony (Program Responsibles, Teachers, and any other Staff are invited by default, and Guests are only invited when a Student registers).
    - You will also have to include scripts to create and populate this database.

- You should create a database that stores the register events, called **eventDB**. Each student can only register once, with their guests.
    - The guests can be stored in the register event database or in a separate database
    - When a list of all attendants is created, each Person (Program Responsible, Teacher, Staff member, Student or Guest) should only appear once.
    - This database is fully editable from your program, as described above.

- Any additional tables, databases, or items that you need to implement the task.

Object-Oriented Program

- You should create an object-oriented program that handles the interaction with the user (see below **Interactive Component**) and with the database (see above **Databases and tables**)

- The program will have different types of objects that house the appropriate data and functionality.

Interactive Component

- Interaction with the users will be handled through Java.util.Scanner, as presented in class.
- A user can sign in when starting.
- A user that is a student can create a new register event. The new event will automatically contain the student's information (name, id, program, and any other relevant information) without requiring the user to input their own data again.
- A user can sign out (taking the program back to the main menu).
- A user can request the information described above
    - lists of all participants,
    - list of student participants,
    - list of student participants from a given program – the input to be read from Scanner,
    - information about whether or not a particular person will attend (either **Student** or **Guest**)
    - The program of the ceremony.

Examples:

```
Welcome! Here are your options:
    1.  Sign in
    2.  See overall program
    3.  Exit

1

Name:
Bogdan M

Welcome, Bogdan! Here are your options:
    1.  Register for the event
    2.  See all participants
    3.  See participants from your program
    4.  Search for participant
    5.  See overall program
    6.  Exit

1

Excellent! How many guests do you wish to invite?
2

Guest 1 name:
EnGuestsson

Guest 2 name:
ToGuestsson

Created entry:
Id: 1, Student: Bogdan M, Program: Programming, Guests: EnGuestsson, ToGuestsson

Welcome, Bogdan! Here are your options:
    1.  Edit registration for the event
    2.  See all participants
    3.  See participants from your program
    4.  Search for participant
    5.  See overall program
    6.  Exit

5

Start time: 13.00
Introduction: 30 minutes.
```

```
Programming – BMR – 1 minute
== Your registered as part of this program ==
    -    1 student  - 1 minute
Break – 5 minutes
DataScience – Resp1 – 1 minute
Break – 5 minutes
FrontendDevelopment – Resp2 – 1 minute
Break – 5 minutes
ArtificialIntelligence – Resp3 – 1 minute
Closing remarks – 15 minutes.

Welcome, Bogdan! Here are your options:
    1.  Edit registration for the event
    2.  See all participants
    3.  See participants from your program
    4.  Search for participant
    5.  See overall program
    6.  Exit

4

Search participant name:
ToGuestsson

Participant ToGuestsson will attend, as a Guest of Bogdan M, from the Programming
program
```

NOTE: This example is just for reference. Feel free to adjust how the program shows and collects information, how much detail you want to include, and how complex you want the interaction to be.

Other details, hints, advice

- Start small, with the most fundamental functionality you think you will need.
- Add more functionality as needed and relevant, once existing functionality works.
- Look for opportunities to use Object-Oriented concepts, where relevant.
- Start with the fundamental requirements (reading and writing information to/from database, interacting with the user via console, using basic object-oriented concepts) before moving to the more advanced ones.
- Clarify (for example, in a readme.MD file) any assumptions that you make, how to run your project, how to navigate the menu system, and any other information you want to provide to users/evaluators

Assessment criteria, for students and assessors

The following will be the main points to assessed:

- Java, Object-oriented programming, and JDBC is what is assessed in this exam.
- The basic OOP concepts such as Inheritance, Polymorphism, Abstraction and Encapsulation.
- Data types such as ArrayList, HashMap, etc.
- Use of advanced Java techniques such as Iterators, Optional and Lambda expressions.
- Exception handler to handle exceptions, and input validation to validate user input.
- Amount of (good) code and complexity are of importance
- Folder structure, tidy code, naming convention

Guideline for how much each part counts

The percentage numbers below present approximate numbers on how much each part will count

- User Interface logic: ca. 20%
- Use of OOP concepts: ca. 20%
- Use of JDBC: ca 20%
- Java techniques (data type, advanced techniques, exception handler etc): ca. 15%
- Report: ca. 10%
- Code structure, tidy code, good names on variables and functions, etc.: ca. 10%
- *Amount of code and complexity: ca. 5%

*All points affect each other. The assessment will require a holistic view of techniques applied in the exam delivery.

--End of exam text--