**EVALUATION FORM (Object-Oriented programming with Java)**

Evaluator name and student number: Your name and student number here

Evaluation type: Self-evaluation / Peer-evaluation

Project’s repository name in Github: Name of the project repository at Github

Total evaluated points: Sum of points from the table below

|  |  |  |  |
| --- | --- | --- | --- |
| **Evaluation targets** | **Points** | **Evaluation** | **Notes and justifications from the evaluator** |
| **Design specification** | **5** |  |  |
| Purpose, goals, users | 1 |  |  |
| Use-cases, user stories (at least one functionality per CRUD operation – view data, create data, update data, delete data) | 1 |  |  |
| Data model (entities, attributes) | 1 |  |  |
| UML Class diagram describing the classes that will be implemented in the program | 2 |  |  |
| **Database implementation** | **5** |  |  |
| Diagram/tables describing the relational model | 1 |  |  |
| SQL script that creates the tables | 2 |  |  |
| Primary and foreign keys are defined in the script | 1 |  |  |
| SQL script that inserts some test data into the database | 1 |  |  |
| **Code management and version control (public/private repository at Github)** | **10** |  |  |
| Readme file in Github with a description of the application | 2 |  |  |
| Design specifications and screen-shots of the GUI are available | 2 |  |  |
| Installation instructions are available | 2 |  |  |
| Source code and SQL scripts | 2 |  |  |
| Wiki pages are used to provide a description of the project/application and important information about it (ex: installation instructions) | 2 |  |  |
| **GUI implementation** | **20** |  |  |
| Control(s) (UI) to view data from some table in the database | 5 |  |  |
| UI to update data | 5 |  |  |
| UI to delete data | 5 |  |  |
| UI to add data | 5 |  |  |
| **Java OO implementation** | **50** |  |  |
| Amount of implemented Classes  1 Class: 2 points  2 Classes: 8 points  3-4 Classes: 12 points | 2,8,12 |  | List here the name of the classes |
| The code contains comments allowing other developers to understand the logic/thoughts under Classes and methods  Some comments can be found in the code: 1 point  All Classes and methods are commented with useful info: 4 points | 1,4 |  |  |
| Access validators properly used at Class (attributes, methods) ensuring encapsulation and information hiding | 2 |  |  |
| Setters, and Getters are used to ensure encapsulation and information hiding in all Classes | 4 |  |  |
| Setters methods contain data validation logic to ensure the program logic will work as expected | 4 |  |  |
| JDBC calls used for CRUD operations  Creating/Adding data: 4 points  Reading data: 4 points  Updating data: 4 points  Deleting data: 4 points | 4-16 |  | Indicate the method names where the operations are being handled, and which operation is performed in the method. Ex:  AddCarDb (adding data to db)  DeleteCarDb (delete a car from the db) |
| Exception handling (try/catch blocks) is performed for Database operations | 4 |  | Indicate the method names where the exceptions are being handled |
| Exception handling (try/catch blocks) is performed for other types of exceptions (ex: arithmetic, input mismatch).  2 points per type of exception (max two exception types) | 2,4 |  | Indicate the method name where each exception type is being handled |
| **Inspecting and evaluating project deliverables and code** | **10** |  |  |
| Self-evaluation form is delivered with **valid/correct** evaluations | 4 |  |  |
| Evaluation of other student’s project is delivered with **valid/correct** evaluations | 6 |  | Name of students being evaluated |

**Conversion table (points 🡪 Grade in the course)**

0-50 points: 0, Not passing the course

51-60 points: 1

61-70 points: 2

71-80 points: 3

81-90 points: 4

91-100 points: 5

**Deliverables specification:**

|  |  |
| --- | --- |
| **Evaluation target** | **Details** |
| **Design Specifications** | File(s) at your github project repository AND/OR descriptions at your repository’s Wiki  The specification should have a format/content according to the guidelines provided in class (ppt slides) |
| **Database implementation** |  |
| Diagram/tables describing the relational model | According to the guidelines provided in class (ppt slides) |
| SQL script that creates the tables  Primary/Foreign keys | .sql file containing SQL sentences that can be imported into your database server. Primary and foreign keys (if applicable) need to be defined in the script |
| SQL script that insert test data | .sql file containing INSERT sentences that will add some initial test data into the database, so that someone can start testing the app |
| **Code management and version control (public/private repository at Github)** |  |
| Readme file | Readme file having similar content as explained at:  <https://guides.github.com/features/wikis/> |
| Wiki pages | Feel free to use the Wiki pages in a way that you think will best describe your project. Keep in mind that a potential employer will be reading the Wiki pages |
| Design specifications | Same content you prepared for the evaluation criteria above published at your repository |
| Screenshots | This is important as someone (potential employer?) may not have the time to install the app and see how it works. Publishing the screenshots of your app will give a good idea of how the application works |
| Installation instructions | Instructions can be written for example in the Readme file. Make sure the instructions are clear and complete, in such a way that anyone wanting to test your app will be able to setup the environment (ex: Database Server, creating the database) and start the app |
| GUI Implementation, Java OO implementation | Implement your app following the guidelines provided in class and information provided in other sources (ex: books) |