**Overall Schedule**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Session | Week Autumn 2017 | Class content | Homework | Milestones |
| 1 | 36 | Course overview (Info in Optima)  Object-orientation key concepts (1)  Car example – 1  Creating Viope account / Accessing the course  Viope 2-1, 2-2 | Study key concepts from your learning resources (check details in the workbook)  Install Eclipse in your own computer. Start bringing your own computer to the classes  Create a GitHub account and email your GitHub username to the teacher (antonius.camara@laurea.fi)  Complete Viope exercises: 2-1, 2-2, 2-3 (Post and check solutions at: Solutions for Viope exercises) |  |
| 2 | 37 | Programming exercises:   * Viope 2-1, 2-2, 2-3 * Own new exercises (using an ArrayList?) |  |  |
| 3 | 38 | Object-orientation key concepts (2)  Car example – 2  Viope 2-4, 2-5, 2-6 | Study key concepts (2)  Install WindowBuilder plugin in Eclipse. Bring own computer to next class  Viope exercises: 3-1, 3-2 |  |
| 4 | 39 | Programming exercises:   * Viope 3-1, 3-2 * Own new exercises |  |  |
| 5 | 40 | UI Programming with Swing  Handling UI events  Project evaluation criteria, self-evaluation, peer-evaluation  Implementing a simple UI and using it in a simple program (MyFirstGUI and MyFirstEvent) | Study Swing and Event Handling (Add more details here) |  |
| 6 | 41 | Creating programs with multiple windows and different components  Ideating the project app | Study Swing and event handling (some example from a book?)  Complete a mini-project until next contact session (upload to Optima, points…)  Ideate your project app (purpose, screens, operations, dialogs) |  |
|  | 42 | Vapaajakso |  |  |
| 7 | 43 | Solution for the mini-project  Designing an OOP app (Functionality, UI, Classes, Relational Database)  Show example of my own app (I only need here to have the specs. I don’t need to have the code). “Job placement tracking” (Student, Placement, Laurea Supervisor, Company)  Designing own app (project) | Create a first dratf of your project's design documentation |  |
| 8 | 44 | Review of own designs (Sharing among people?)  Designing and implementing a relational database. Review concepts  Own exercise (DB design)  Designing and implementing a relational database (Own project) |  |  |
| 9 | 45 | Using JDBC to connect to a database  Own exercises |  |  |
| 10 | 46 | Mini JDBC app in class |  |  |
| 11 | 47 | Error handling |  |  |
| 12 | 48 | Project check-point (workshop activity?) |  |  |
| 13 | 49 | TBD |  |  |
| 14 | 50 | Deadline for project delivery |  |  |
| 15 | 51 | Project demos, deadline for self- and peer- evaluations |  |  |

**Course description**

This course is targeted to students that want to extend the basic programming skills learned during the first year of studies or want to pursue careers in web/mobile application development. In the web/mobile app development job market, knowledge of Object-Oriented programming techniques is a necessary pre-requisite as many commonly used development frameworks are based on Object-Oriented principles (ex: Android SDK, PHP Symfony, PHP Laravel, Java Spring).

In this course students will extend their java programming skills with object-oriented programming techniques. The learning process will be based on a personal project to implement a java desktop application. Students can freely decide the purpose and topic of the application. Besides applying object-oriented programming principles, students will also learn methods to design object-oriented applications, how to access relational databases with JDBC, and how to use the Swing toolkit to implement User Interfaces.

The java applications developed in this course are expected to be simple desktop applications consisting of few screens where the user can read data from the database, insert data, update data, and delete data. The application's user interface should contain standard UI elements such as text boxes, drop-down lists, and command-buttons.

**Pre-requisites**

First year Tiko/BIT ICT courses: Fundamentals of programming (any language), Basics of relational databases and SQL.

Students should have an own computer to install the development environment and do the development work.

**Course contents**

Application of Object-Oriented programming principles in the implementation of an own application

Database access with JDBC

Eclipse IDE

GUI implementation with Swing and Eclipse WindowBuilder plugin

**Course material**

Besides the material provided by the teacher during the contact sessions, students can make use of a diverse range of resources. For example:

Beginning Java Programming: The Object-Oriented Approach, Baesens Bart, Wiley, 2015 (Link to Ebook Central: http://ebookcentral.proquest.com/lib/laurea/reader.action?docID=1895071)

Java 2 - Ohjelmoinnin peruskirja, Kosonen Pekka, Docendo, 2008

**Guidance and communication**

Optima

Lectures and course material are in English. Finnish students (Tiko) can return material in Finnish and can communicate with the teacher in Finnish

**Evaluation**

Evaluation will be based on the outcomes of the students personal project (java desktop application). There will be no exams.

Students will implement a personal project (Java desktop application) based on the targets and criteria specified by the teacher. During the project work, students will collect points from completing different project targets according to the set criteria and deadlines. The final grade will be based on the total points collected by the student. There will be no exams.

**Other arrangements**

Attendance to the first contact session is compulsory. Students that do not attend the first contact session and do not inform the teacher in advance will be removed from the course. The unused study-places will then be offered to students that are on the enrolments waiting list.

Attendance to other contact sessions is not compulsory and it is optional. Contact sessions will be offered to support students learning process (theory, workshops, development environment, personal project). The course will not have contact sessions every week. Students should check in ASIO the reserved dates, and in Optima the content of each contact session.

If a student misses contact sessions it is still possible to complete the course, provided that the student is able to independently learn the topics and comply with the requirements and deadlines of the personal project work.

Evaluation target and criteria

Excel file containing all the items and points each one can get.

Application design

* Class definition
* User interface wireframes
* Relationships between classes
* Classes with inheritance

Database

* Relational schema with primary keys and foreign keys
* SQL script that creates the database
* SQL script that inserts test data in the database

Github

* Public repository with the following structure: … (package structure from Eclipse)
* Local repository under Eclipse mapped to Github repository

Application

* Main screen
* Screen listing data
* Delete
* Update
* Insert
* Classes implemented (3-7) (points vary)

Milestone 1

* Certain things uploaded to Github

Milestone 2

* Certain things uploaded to Github

Milestone 3 (Final)

* Certain things uploaded to Github

Project case

Same type of app used in Deitel’s book ATM (chapter 12 and 13)

Classes for applications objects

Database access with JDBC, database access occurs under methods of different classes

Optima and Github

Optima: info about course, schedule, evaluation, link to github repositories

Github repositories:

* Example source code used in classes
* My project application (private, not visible to students)
* Students application (student’s own public repository)

Github workflow – Course code examples

|  |  |
| --- | --- |
| **Teacher** | **Student** |
| Create e new private repository to store the courses’ example codes |  |
| Create a team under Laurea’s organization |  |
| Add the team to the repository (Give only Read permission) |  |
|  | Create a personal Github account (<https://github.com/join>) |
|  | Send to teacher your Github account name |
| Add student’s Github account to the team |  |
|  |  |
|  |  |
|  |  |

Github workflow – Setting up student’s public project repository

|  |  |
| --- | --- |
| **Student** | **Teacher** |
| Install Eclipse in own computer |  |
| Create the project folder structure in Eclipse (Project, Package, etc…) | Teacher provides guidelines |
| Install the GitHub Desktop app on your own computer |  |
| Using the Github app, create a local repo on the Eclipse project’s root folder (src). NOTE: Select “Add” repo (instead of “Create” or “Clone”), then when informed that the location seems to be empty, click on the “create repo” link |  |
| Via the Github app: 1) Commit the changes to master; 2) “Publish” the repository to Github.com as a public repository; 3) Sync the repository |  |
| At Github.com: Add your teacher’s account as a collaborator in your repository |  |
| Do development work on your own computer |  |
| Document your work as instructed by your teacher |  |
| Sync Milestone - 1 | Including self-evaluation |
| Sync Milestone - 2 | Including self-evaluation |
| Sync final work | Including final self-evaluation |
|  |  |
|  |  |

**Creating the list of peer reviews**

Student submitting the project x Evaluator

Form soleops get the list of all students with student numbers

Make one list for active students another for inactive

First column, students sorted by alphabetical order x Second column students sorted by student number

Make the same for inactive students

Merge the lists

Sort all again by alphabetical order