Description of the project work

Last modified 2021-02-26 – Erno Vanhala

General guidelines

Briefly

- Project is done in the group of two or three people, but one can also make the whole work by herself, but it has to still has all the features
- One must have the learnt features in the project – such as classes, inheritance, methods etc.
- The task is to implement an Android application
- One can get 0 to 40 points from work. 13 points is the minimum accepted and one gets it by doing the mandatory part

General requirements

- Application must work with Android 6.0 or newer
- Code and its comments must be in English (documentation can be in English or Finnish)
    - Before method short description
        - E.g. /* This method takes the input value, translates it to a sum of ASCII values and returns it */
    - No need to comment method if its name is getName() and it return a name
    - When there is doubt, comment
    - Write code that some random dude can understand, not just you
        - Within group, read each other's code

Returning

- Deadline is available in Moodle
    - Returning is in two phases
        - 1st you return your plan with class diagram and ideas what kind of an app you are creating

        - Final return includes complete Android application and documentation
- Returning the project is handled through Moodle (no email returnings)
- Things you need to return:
    - Source code in some **code repository** selected by you (e.g. GitHub, BitBucket etc.)
    - Short (max 4 minutes, less than 100MB) video (in streaming service or in repository) demonstrating the application
    - Documentation
    - **The best solution is to create documentation directory in code repository and put all the documentation there**
- All the materials (code, documentation) should be available until you have received the final grade

Documentation:

- Short description of the application you have created
- Who did what (not just names, but larger descriptions)
- How was the application designed and implemented
- Class diagram with most of the classes and their most important attributes and methods
- List of implemented features and final points based on these
- Some kind of an estimate on used hours (e.g. 20 days 5 hours per day == 100 hours)
- Every group members answers to the question “What I learnt from the project?” (at least 3 sentences)
- Group can tell what parts of the project were hart, what easy
- Documentation has to be in **PDF-format**
- Link to the document template

Description of the project work

The following table list features that have to be implemented and then there are optional features to gain more points. So the project is composed from

- Basic features that give you 13 points (the minimum requirements)
- Optional features that can give you more points (13-40)

The maximum number of points is 40. If you develop your own features, please mark them and justify their points. Remember to use the object-oriented way.

The project topic is to build health and wellness application to improve users' life. It can mean many things, such as weight loss, decrease of CO2 footprint or bookkeeping drunk soda cans. Group can decide what features the application has; the more features the more points.

Object-oriented way with at least 5 classes used is mandatory. Also utilization of at least one public open data API (such as the climate diet to calculate CO2 footprint, used similarly as in the Finnkino exercise) is mandatory. Based on user and API input/output the

application makes a log that the user can check. These are the basic features that give 13 points.

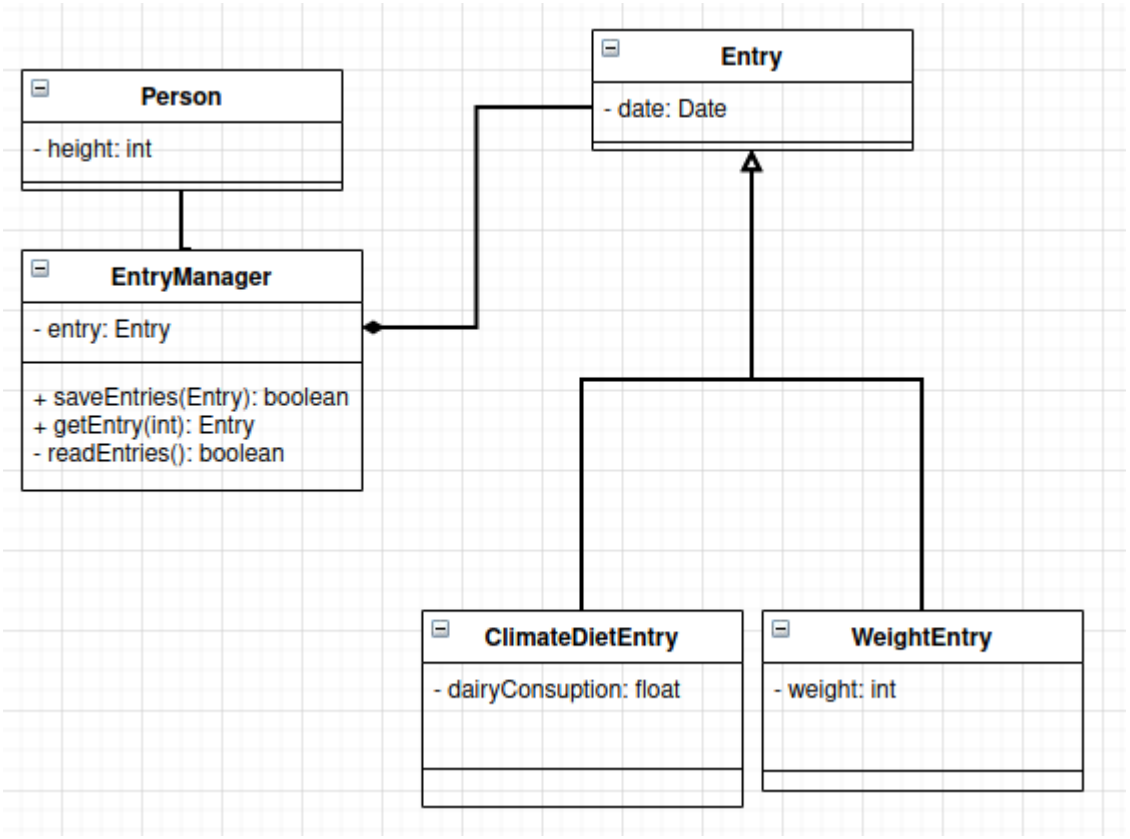

Figure 1: Ideas on what class diagram could include

In the following is a list of features that project work must and could have:

Feature	Points
Utilizes object-oriented programming	Mandatory
At least 5 classes (GUI classes are not counted)	Mandatory
At least one API must be used, e.g. Climate diet: https://ilmastodieetti.ymparisto.fi/ilmastodieetti/swagger/ui/index	Mandatory
Application saves user inputs to log (JSON, XML etc.)	Mandatory
One has a way to view the log (plain text, with graphs etc.), so user has an option to check her inputs (e.g. own weight) development over time	Mandatory
Application is build from well designed UI-components	1 – 5 points

Application has a login function	3 points
Application can has several users (and one can create more), data is saved somewhere	3 points
Strong (at least one number, one special character, bot upper- and lowercase letters and is at least 12 chars long) login password is required	2 points
Password is hashed and salted	2 points
Another open data source is used (e.g. https://www.avoindata.fi/, https://www.europeandataportal.eu/fi tai https://thl.fi/fi/tilastot-ja-data/aineistot-ja-palvelut/avoin-data/avoimet-rajapinnat), so that there are at least two data sources now implemented	2 – 5 points
User can input basic information of herself (length, weight, age, picture, city etc.) and these values are used in the application	2 points
Application collects data from user's weight change (she inputs the data) ja shows graphical presentation of change over time	3 points
Application shows graphically how the output of climate diet api has changed over time (e.g. how meat consumption has decreased and CO2 footprint has changed)	3 points
Application tells risk factors in city, age group, etc. user is living (e.g. the percentage of smokers) based on THL data and users own inputs	2 points
Application combines data from at least two data sources and produces new information	3 points
Application remembers where user was before closing the app	2 points
Usage of asynchronous HTTP-calls when getting data	2 points
Use of fragments instead of activities when building GUI	2 points
Utilization of scoped storage when saving data (app does not require user permission to access mass media, but works in its own sandbox)	2 points
Responsive UI (It works well with different screen sizes)	2 points
Documentations is not comprehensive	-1 – -5 points
Application has problems that disturb its use	-1 – -5 points
Program code is not written/commented in English, but some other language (UI texts can be in Finnish)	-3 points

Application does not follow object-orientation	-1 – -10 points
Application and/or documentation has racism, hate-speech or something other impropriate	-20 points
Some own fancy feature or several of those	Max 5 points per feature
Together (if features generate more than 40 points, the extra points can help to compensate weaker features, but no more than 40 points can be collected in total)	Max 40 points.

Important notes:

- Remember to use object-oriented programming and use at least 5 classes
    - At least five classes are mandatory, because object-oriented programming is hard to do without classes
    - UI classes should be separated from data classes
        - Just like Bottle and BottleDispenser in exercises
    - This is MVC-pattern (model-view-controller). Android Studio uses also MVVM pattern (model-view-viewmodel), but also there data is separated from UI

Own project topic specification

- Create a plan for an application and basic class diagram
- Describe to how many points are you aiming at
- Justify how this point score should be suitable for your work
- Notice how your own project needs to have similar features that the specified project (e.g. at least 5 classes, data collecting from external API, documentation, etc.)
- Send the plans to the lecturer via email. The plan is then checked and accepted or more information is asked