SDD

System design document for Learn Java

Version: 1

Date: 9/5-17

Author: Tobias Lindgren, Madeleine Lexén, Hanna Carlsson och Sara Kitzing

This version overrides all previous versions.

1 Introduction General info.

What is this? What does it describe?

1.1 Design goals

The application is designed for android, and must therefore be able to run on android. It should be able to run on Lollipop or higher.

For Learn Java to be able to withstand changes, the code should be easily extendible. This means that the classes need to encapsulate behavior and be loosely coupled. These changes could be to the project scope and platform requirements.

The application also depends on some external libraries, and must therefore also be loosely coupled with them to support adaptability to future changes.

1.2 Definitions, acronyms and abbreviation

Some definitions etc. probably same as in RAD
Key: The correct answer
Level: A subcategory within a main category. Contains information about the specific topic and a question on the topic.
Map: The start page, a map over the different main categories
Read more: A page which shows information about basic programming in text form that the user will learn throughout the different worlds. It can be accessed from the menu in all views.
Boss: The last level in a main category which summarizes what the previous levels has taught the user.
Fill in the blanks: A question where the user will get a number of sentences with blanks that they will have to fill in themselves.
Write code: A question where the user will get a task where they need to write code

2 System architecture

2.1 Overview

There are two machines involved in this system; the server, which is here represented by the computer running the program, and the phone the app its run on, either a virtual machine or a physical phone.

2.1.1 Server

The program needs to be able to compile code written as user input in the boss-querys. Since the program is an android app this is not possible to do on the actual device, why

there server is implemented. The server is based on a ServerSocket. A String with the user input is sent from the app to the server. Ther server compiles the user code and sends back the result as a string. The server class also handles necessary operations such as setting up and closing the streams.The server project is not on an actual server since we did not have access to one, therefore you need to run the app on the same computer as you are running the server.

2.1.2 Compiling code with BeanShell

BeanShell is the library which in this application is used when compiling the code from the user input. The only class used in BeanShell is the Interpreter. Interpreter-methods is used to compile a String that comes from the servers inputstream and the interpreter-output is sent to the file compiledCode. It is then read and shortened to so only the latest result is sent back.

2.1.3 User and statistic

The app has a login function and therefore a user. Each user is an object of the class user, and is saved has a bin file each time the app is closed. When a user logs in the users object is loaded from the bin file through the AccountManager singleton class, which handles the login and add new user methods. Every user object holds a reference to a statistics object where the statistics is saved. The statistics model works as a database but instead of saving the data to a real database, the data is stored in lists.

2.1.4 Query

The app has a few different varieties of questions, and these have a common superclass Query. This is so polymorphism can be utilized when delegating the checking of the answer to the right type of question.

2.1.5 Category and levels

The game is divided into four categories, *Primitive types, boolean-logic and if-statements, for- and while-loops* and *arrays.* Each of these categories contains five levels and the levels contain a query, info for the query and a hint to help solve the query. When accessing a category for the first time, only the first level is unlocked. To unlock the next level you have to complete the previous level first. When all levels in a category has been completed, the next category is unlocked.
The Level is an object containing a *Query*-object, a *hint*-string and an *info*-string. The *Query*-object contains a *question*-string and an *answer*-string. All *Level*-objects are created when accessing the game and stored in a *HashMap* so they are easily accessible when needed. When creating the *Level*-objects, all required strings are retrieved from .txt-files. There are three different kinds of queries, *Fill in the blanks, Multi choice* and *Write code.* The *Fill in the blanks*-query gives the user a text with missing words that the user are supposed to fill in. The *Multi choice*-query asks a question and gives the user four different choices to choose between when answering the question. The *Write code*-query has the user writing it’s own code to get a certain output.

The most overall, top level, description of the system. Which (how many) machines are involved? What software on each (which versions). Which responsibility for each software? Dependency analysis. If more machines: How do they communicate? Describe the high level overall flow of some use case. How to start/stop system. An UML deployment diagram, possibly drawings and other explanations. Possibly UML sequence diagrams for flow. (Persistence and Access control further down) Any general principles in application? Flow, creations, ...

3. Subsystem decomposition

Learn Java is decomposed into four packages: model, controller, service and layout. Model, controller and layout is using the MVC-pattern, service handles reading and loading files and connection to the server. All services is communicating to the model through the controller-classes.

3.1 Model

The model packages handles the logic behind the application. It does not have any connections of its own, the controller communicates with it.

UML for model-package

3.2 Controller

The controller package handles all the communication in the program. It has listeners on buttons in the layout and sends the information to model. It also handles the communication between the service module and the model.

UML for controller-package

3.3 Layout

The layout package contains all the XML files. Each component in the layout has an id which is coupled in the controller package.

UML for layout-package

3.4 Service

The service package holds all the classes that is not connected to MVC. It handles tasks such as reading a file, getting images and set up and connect to the server.

3.4.1 Server

3.4.2 File reader

3.4.3 Image handler

3.5 Dependency analysis

Insert STAN-figure here
Comments on STAN

Recap: What is this doing (more detailed) Divide it into top level subsystems. An UML package diagram for the top level. Describe responsibilities for each package (subsystem). Describe interface. Describe the flow of some use case inside this software. Try to identify abstraction layers. Dependency analysis Concurrency issues. If a standalone application - Here you describe how MVC is implemented - Here you describe your design model (which should be in one package and build on the domain model) - A class diagram for the design model. else - MVC and domain model described at System Architecture Diagrams - Dependencies (STAN or similar) - UML sequence diagrams for flow. Quality - List of tests (or description where to find the test) - Quality tool reports, like PMD (known issues listed here) NOTE: Each Java, XML, etc. file should have a header comment: Author, responsibility, used by.., uses ...

4. Persistent data management

The application data is mainly stored as a bin file. This file contains an AccountManager with a hashmap which holds each of the applications usernames as the key and the corresponding User-objects as the value. Each User-object holds the information associated to the specific user as well as a Statistic-object containing an arraylist with statistics for each level. However, the pictures is saved in a directory where each picture is related to its User through the user’s name.

The text for all questions is saved in text files, which is read when creating all the levels objects when the program is started.

5. Access control and security

NA

6. References

Andras SDD:
https://github.com/Lajsyl/Falling/blob/documents/SDD.pdf
https://github.com/nahojjjen/OOP-Project-TowerDefence/blob/master/document/SDD.pdf
https://github.com/Tejpbit/TDA367/blob/master/Documents/RAD%26SDD/SDD.pdf