weCharade: Developers Manual

DAT255, Software engineering project
Chalmers, VT 2013
Latest revision: 2013-05-26

An overview of the Android application weCharade and a guide how to get started.

Overview

Dependencies

- Java 6 SE development environment
- Android SDK
- A physical or virtual Android device

Android SDK targets

Minimum SDK:	10
Target SDK	17

Build procedure

1. Start Eclipse
2. Install EGit-plugin and Android SDK API 10 and 17 if needed.
3. Go to File -> Import -> Git -> Projects from Git
4. Add URI: git@github.com:alestep/faaa_project.git with SSH
5. Choose master branch and import into existing projects
6. Choose faaaproject and press finish
7. The application can now be built and run on an emulator or an Android device.

The application uses Android 2.3.3 API level 10 and has been tested with emulators with API level 10 (2.3.3) and 17 (4.2.2). Current android devices has been used to test the application:

- Sony Ericsson Xperia Pro (Model number: MK16i), with Android 4.0.4
- Sony Ericsson Xperia Active (Model number: ST17i), with Android 2.3.3
- Samsung Galaxy SII LTE (Model number: GT-I9210), with Android 4.0.4
- Samsung Galaxy SIII (Model number: SGH-i747), with Android 4.1.2

Tests

Automatic tests are included in a separate and is called weCharadeAndroidTest. The tests are done in two different ways. The main part contains JUnit tests to primary test the *com.example.wecharades.mode*l package.

The second part is to test the GUI. This has been done with base in the projects user stories and activities and is done by the test classes using Robotium (version 4.0.1) in the *com.example.wecharades.views.test* package.

Architecture

The application code resides in packages organized by a MVP pattern, which stands for Model-View-Presenter. The MVP pattern separates the UI concerns between the data of the UI (the Model), the display of the UI (the View), and the logic of the UI (the Presenter). In the case of android this means that the View is the Activity, which handles user input and updates the display. The Presenter is a class that handles the communication between the Model and the View, and the Model handles the persisting and retrieving data, along with any logic that the data must adhere to.

Package – Model

The package consists of classes that controls the connection with the database and a the local data storage.

Package – Presenter

The presenter package contains the presenter classes that connects the activities with the model.

Package – View

The view package consist of the activities and which handles the display of the UI.