Developer Manual

For "Alarmed+"

Installation and setup:

1. Clone the git repository from https://github.com/rekoil/DAT255-EpiClock
2. Make sure you have the Android SDK (API level 15 and support library), an Android (Virtual) Device and Java 6 SE development environment.
3. From eclipse import the project as an android project. This will import the main project as well as the test project into your eclipse workspace.

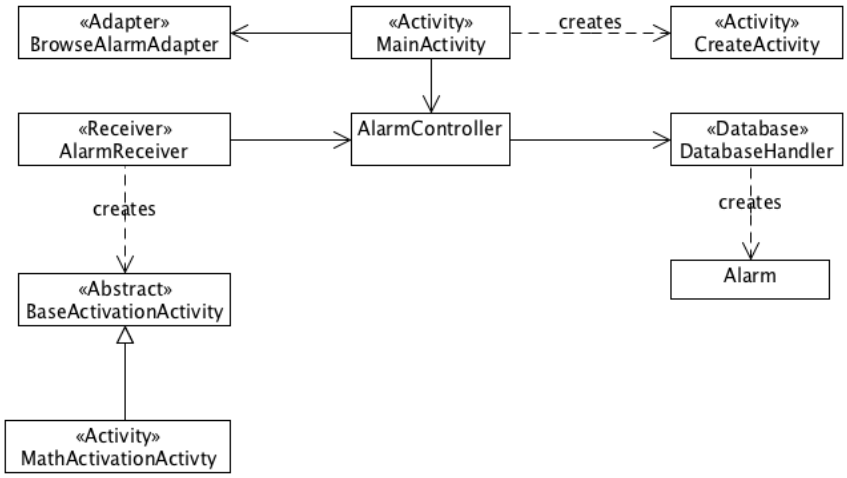
Tests:

All the tests for the project are located in a seperate test project, AlarmedTestTest. There are two ways to run the test suite.

- Eclipse
  - Right click on the test project and choose *Run As -> Android JUnit Test*
- Run the tests via ant (this requires the emulator)
  - Navigate to the test directory
  - Then run: **ant clean debug install test** (if the first time)
  - After that you can simply use **ant run**
- Via ant can you generate an emma test report via: **ant clean emma debug install test**
- Then open the AlarmedTestTest/bin/coverage/coverage.html in your web browser

Architecture:

Overall:
See figure below for an overall overview of the system design. See Modules for more information about the BaseActivation and module system.



Controllers:

The system is based around the AlarmController to set and acquire alarms from the AlarmHandler. The current implementation of the AlarmHandler is using an SQLite database to store the Alarms. The AlarmController is also responsible for scheduling the alarms via AlarmManager in Android.

Package Structure:

The package structure follows class types, so activities pertaining to the main application are part of the .activity package, controllers in the .controller package and alarm activation modules in the .modules package, with their own module-specific .activity, .model, etc. packages as required.

Modules:

Modules are stored in a separate package named .modules. Each module is only required to include an activity that can be launched by the application when an alarm goes off. Modules extend a 'BasicActivationActivity' which supplies them with methods for controlling the volume of the alarm and of course the ability to stop the alarm. All an implementing ModuleActivity has to do is extend the the abstract activity and then do a **super.oncreate()** in the oncreate method and then when the user has completed the task **super.stopAlarm().**

MathModule

The MathModule currently has five different implementations of MathProblemType and is randomly selected and presented to the user.
They are:

- FibonacciProblem
- PrimeProblem
- AdditionProblem
- MultiplicationProblem
- FactorialProblem

The MathController is responsible for generating new problems for the MathActivity, which is reading the input from the user and checking the answers with the controller, which in turn (based on the number of correct answers in a row) raises or lowers the difficulty.
For UML overview see the image below: