Advanced User Interfaces
2015-2016

MindWatch

836897 Simone Graziussi

February 1st, 2016

Abstract

MindWatch is a smartwatch application for brain training. The idea is to submit to the user logic, math, general-knowledge or memory questions/games at random moments of the day directly on the wearable device. The proposed games are very easy to understand and quick to solve, in accordance to the wearable device usage paradigm. The application also exploits the user current position to show location-related games, for example asking to compute the interests on a given sum if the user is at a bank.

Contents

I Introduction 3

1 Extended Abstract 3

II User Requirements 3

2 Stakeholders 3

3 Needs 3

4 Goals 4

5 Constraints 4

6 Requirements 4

7 User Interface 5

III Use Cases 5

8 Smartphone: Profile 6

9 Smartphone: Settings 7

10 Smartwatch: Game 8

IV Scenarios 9

11 Problem scenarios 9

12 Interaction scenarios 10

V Design: Boundary-Control-Entity Diagrams 10

13 Handheld 12

13.1 User Interaction 12

13.2 Games Scheduling 13

14 Wearable 13

VI Implementation: Architecture **14**

15 Android Handheld **14**

16 Android Wearable **15**

17 External libraries **15**

VII Implementation: Persistent Data **15**

18 General Description **15**

19 SQLite Game Database **15**

VIII Implementation: Class Diagrams **16**

20 Common **17**

21 Handheld **18**
21.1 User Interaction 18
21.2 Games Scheduling 19

22 Wearable **21**

IX Conclusion **22**

23 Image Gallery **22**

24 Challenges and Critical Aspects **23**

25 Future Directions **23**

26 Bibliography **24**

Part I
Introduction

1 Extended Abstract

MindWatch is a smartwatch application for brain training. The idea is to submit to the user logic, math, general-knowledge or memory questions/games at random moments of the day directly on the wearable device. Computation of the random moments of the day is done according to the settings specified in the companion handheld application, where the user will also be able to set the games difficulty and to review his/her progress.
The proposed games, entirely shown and solved on the wearable device, are very easy to understand and quick to solve, in accordance to the wearable device usage paradigm: the user will read the text and select an answer in a few seconds.
The application also exploits the user current position to show location-related games: for example asking to compute the interests on a given sum if the user is at a bank or a football-related question if the he/she is at the stadium.
For this reasons, the main goals of the project are to provide an easy and fast way for the user to keep his/her mind sharp and to offer location-aware content to motivate the user in the effort.

Part II
User Requirements

2 Stakeholders

- user of the application: the main target is adults (general audience) who want to keep their mind sharp by "training" every day
- programmer
- teacher/tutors

3 Needs

- keep the brain active, exercising memory, quickness, etc.
- compete with other users: the point system may lead to competition between users, to see who is better/faster at solving random challenges

4 Goals

- solve different types of logic or general-knowledge questions/games
- receive challenges at random moments of the day
- if possible, receive challenges related to the user current position
- gain points if the solution is correct
- visualize the user progress for each brain training type, e.g. via a graph
- share progress with other users
- set difficulty level for the challenges
- set time slots in which to receive the challenges
- set frequency of challenges during a day

5 Constraints

The main constraint is the smartwatch technology. The user expects easy and fast interaction, so:

- questions/games should have a simple structure: the user must understand the problem in a few seconds (no long texts, no complicated rules, etc.) and answer in an easy way (maximum four options, with large buttons)
- interaction should be very quick: the game should be solvable in a few seconds, in order to avoid using the smartwatch for too long

6 Requirements

- smartphone application:
  - provide access to settings, like difficulty, time slots, etc.
  - show the user progress: gained points, percentage of solved challenges, etc.
  - allow the user to share his/her progress via email or social networks
- smartwatch application:
  - show challenges as notifications

- provide a way to answer the challenges: four options easy to select
- store the results and add points if necessary
- generate challenges at random moments taking into account selected time slots and difficulty
- show challenges related to the user's current position: for example if he/she is inside a mall the game could be related to a shopping activity
- avoid being too "annoying": challenges should not be submitted too close to each other or outside the selected time slots and frequency

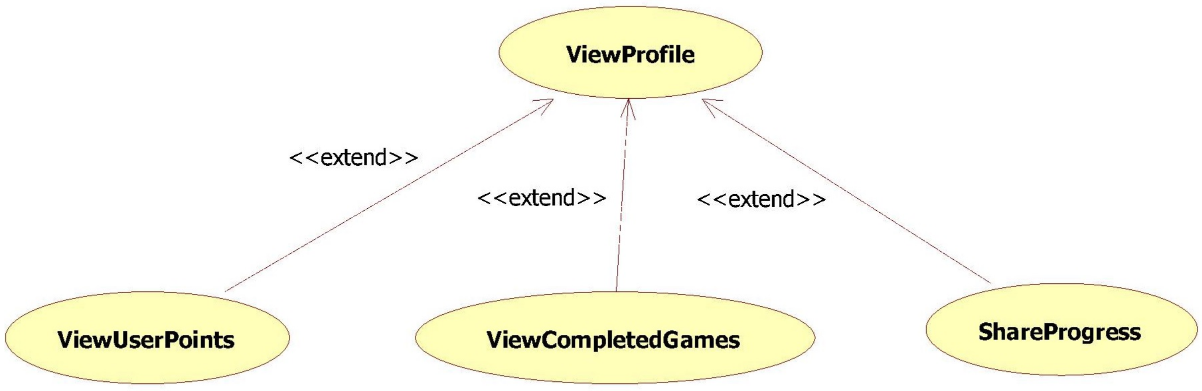
7 User Interface

As previously mentioned, the main concern is with the wearable application. Smartwatches are characterized by small screens, so the first problem is to create games that don't have too much text or too big images, otherwise it would be impossible for the user to understand them.
The second main problem is that wearable devices have different screen sizes and shapes, like round and square, and so the game content and options must be displayed properly in the device the user is currently using.
Moreover, care must be taken in how the user selects the game answer: options should be just a few (at most four) and displayed in easily selectable buttons.
Finally, the game timeouts (e.g. seconds remaining to select an option) should be displayed in a clear way that does not take too much screen space.

Part III
Use Cases

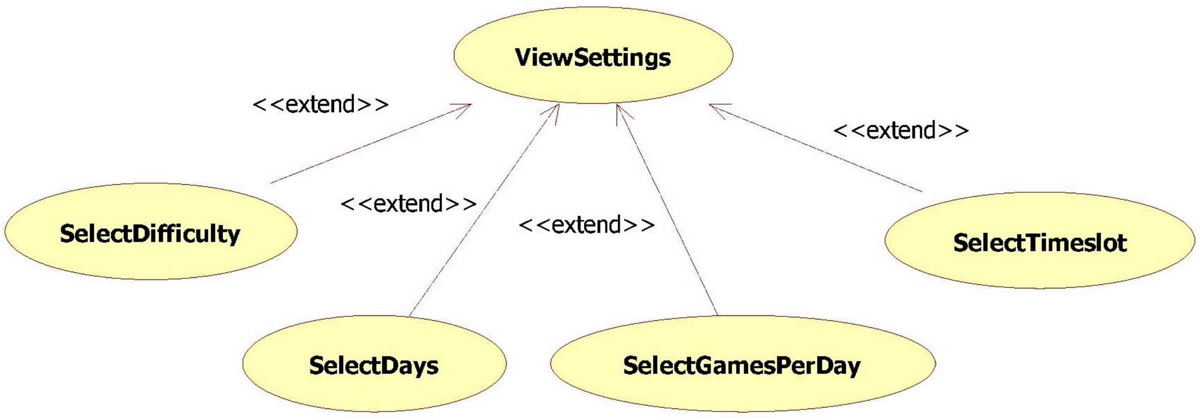
8 Smartphone: Profile

Actors	User (smartphone)
Entry Conditions	The smartphone app has been opened by the user
Flow of events	The user in the profile page can: • view the points gained so far in every game category, displayed as a radar chart on the main screen • view the percentage of completed games, displayed as a progress bar for each difficulty level on the main screen • share the progress via email/social networks, tapping on a button on the action bar • access settings, tapping on a button on the action bar
Exceptions	None



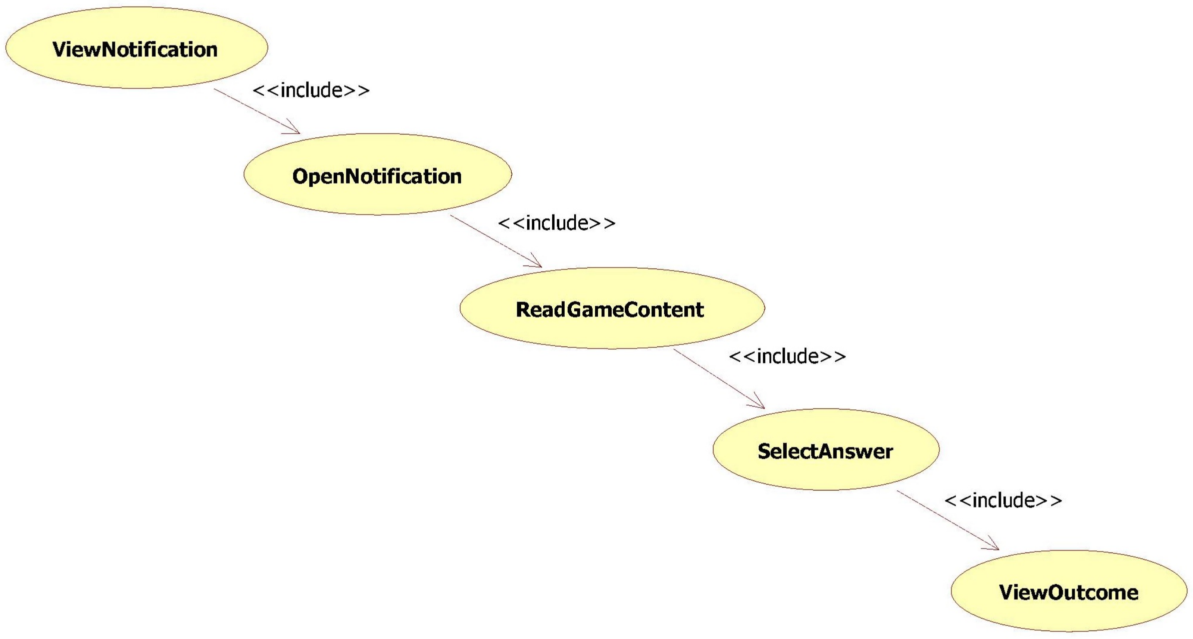
9 Smartphone: Settings

Actors	User (smartphone)
Entry Conditions	The user has clicked the Settings button on the profile page
Flow of events	The user in the settings page can: • select the games difficulty among different possibilities (e.g. Easy, Medium, Hard) • select the weekdays in which games will be proposed to the user • the timeslot in which games will be proposed to the user during the selected days • set how many games will be shown each day
Exceptions	If the timeslot is inconsistent (starting time after ending time) the system shows an error message and asks the user to select a valid configuration



10 Smartwatch: Game

Actors	User (smartwatch)
Entry Conditions	The smartwatch received a notification from the smartphone application
Flow of events	• user receives a notification on the smartwatch • user opens notification (by swiping left and clicking on the “Open” button) • user reads the game content for a given amount of seconds (depending on type and difficulty of the game) • after those seconds expire, the system automatically shows the game options • user has a given amount of seconds (again, depending on type and difficulty of the game) to select the answer among the provided options • when the user selects an answer or the timeout reaches zero, the game outcome is shown: “Correct”, “Wrong” or “Time’s Up” • user closes the page by swiping left and goes back to the smartwatch main screen
Exceptions	None



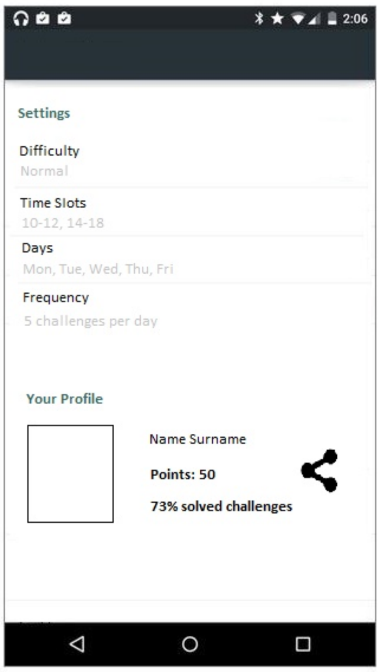
Part IV
Scenarios

11 Problem scenarios

- Tom is a factory worker and he'd like to keep his mind sharp by doing some exercises once in a while. He does not want anything too complicated, with strange rules to learn and hours of thinking to reach a solution, but something fast and easy to learn. He decides to download the application: he sets a timeslot in the afternoon when he is usually at home and an easy difficulty level. In the afternoon, he receives some games at random moments where he is asked to solve basic math equations or complete a sequence of shapes/numbers. One of them is related to the place Tom was in at the time, a bank.

- Harry, John and Jack are three students who like to compete in games and they decide to download the application: they set maximum difficulty and the whole day time slot. They challenge themselves to solve the greatest number of questions/games in the next days: doesn't matter if they are at school or if they are eating when they receive the problem, they must solve it in order to beat their friends. Every day they share their progress via social networks to see who's the best.

12 Interaction scenarios



Settings

Difficulty
Normal

Time Slots
10-12, 14-18

Days
Mon, Tue, Wed, Thu, Fri

Frequency
5 challenges per day

Your Profile

Name Surname

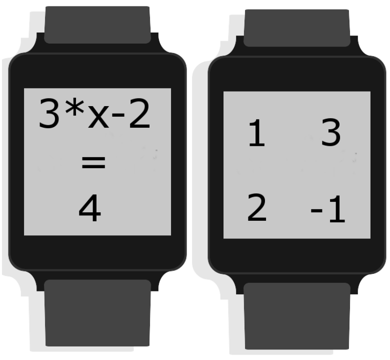
Points: 50

73% solved challenges

Bob open the application on the smartphone and goes through the settings:

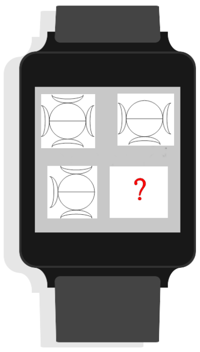
- selects “Normal” difficulty from the given options
- sets the timeslot, from 9 AM to 16 PM
- selects only the weekdays, during the weekend he does not want to be disturbed
- selects 5 challenges per day

He takes a look at his profile, empty for the moment

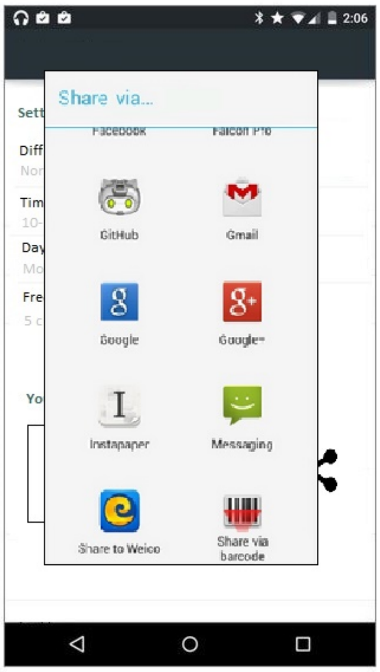


At 10:30 he receives a challenge: it’s a basic equation.

He has a few seconds to answer: he thinks fast while the content is show, then four options are showed. He selects 2 and the application tells him that the answer is correct.



At 13:45 he receives another challenge, this time it's a "complete the sequence" game. Again, four options are showed: he selects the wrong answer, so he gains no points.



In the evening, after receiving the 5 challenges of the day, he decides to share his progress with his friends.

Part V
Design: Boundary-Control-Entity Diagrams

Boundary-Control-Entity Diagrams provide some information about the application design choices and clues on how the system will actually work.

The boundary classes represent the objects that interface with the system actors. In our case, the only actor is the user.

The control classes implement the business logic of the application by providing some methods to perform them.

Entities represent the system data.

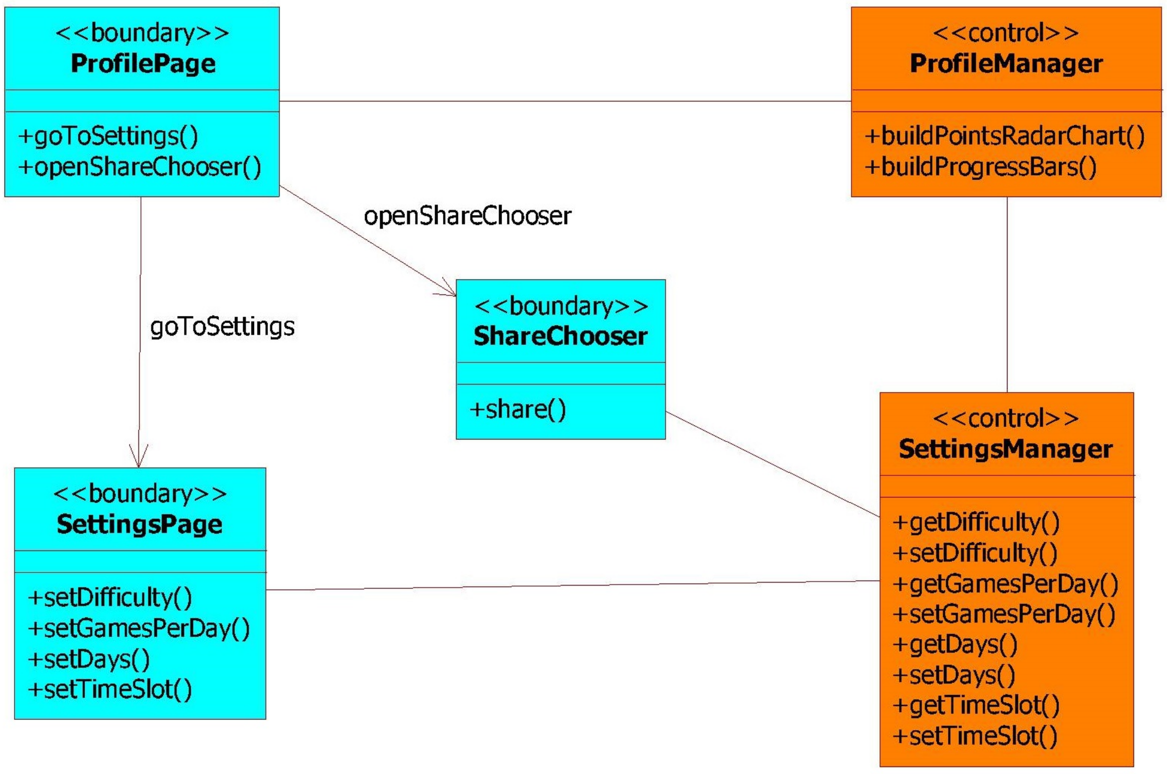
13 Handheld

13.1 User Interaction

The first part of the handheld functions is to provide user interaction with profile and settings. For this reason the system provides two main screens, Profile and Settings.

The Profile Page will show information about the user progress (points and completed games percentage) provided by the Profile Manager and allow the user to share his/her progress through the Share Chooser. Both pages will of course contact the Settings Manager to retrieve the game general preferences.

The Settings Page will allow the user to see and edit the application preferences, like difficulty, games per day, etc.



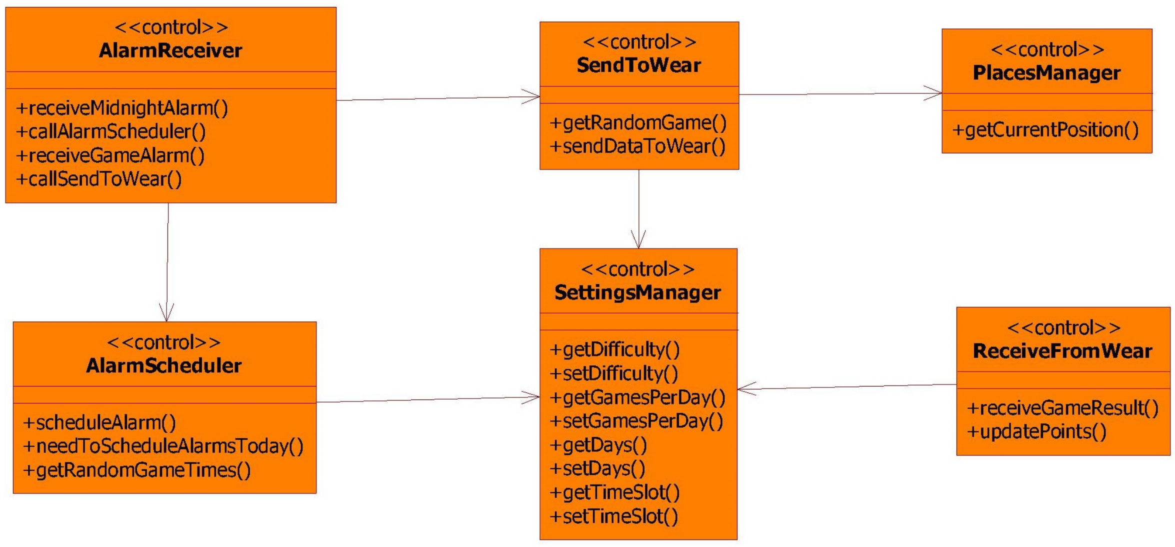
13.2 Games Scheduling

The second part of the handheld function is to schedule the games to send to the wearable application, which requires no user interaction.

At the first application launch, a repeating alarm will be scheduled at midnight of every day. When this alarm is received by the Alarm Receiver, it will call the Alarm Scheduler to setup the daily games. After obtaining the user settings (days, timeslot and games per day), the scheduler will setup the alarms for the games at random moments of the day.

When the Alarm Receiver receives a game alarm during the day, it will call the Send To Wear component. This component will take care of querying the current user position via the Places Manager, to select a proper game from the database (e.g. considering the difficulty provided by the Settings Manager) and to send the data (text and/or images) to the wearable device.

After sending the game, the Receive From Wear component will wait for a response from the smartwatch. Once this is received, it will take care (calling the Settings Manager) of updating the user points.

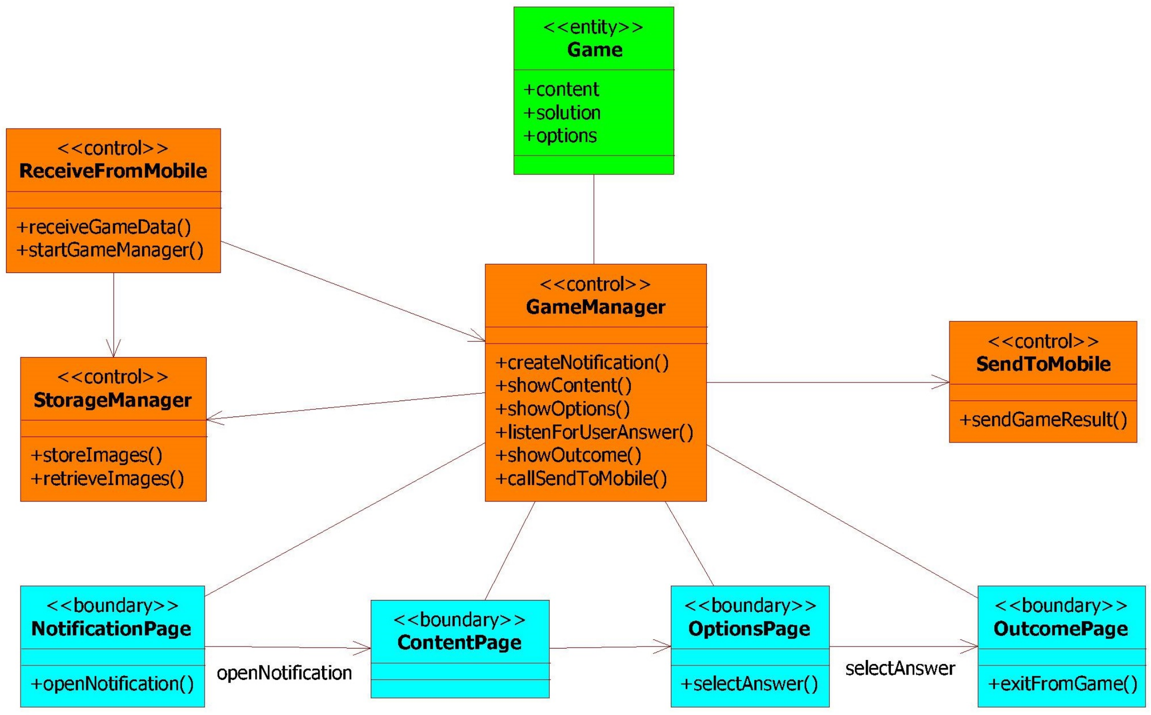


14 Wearable

In the smartwatch application, the Receive From Mobile component will receive the game data (text and/or images) from the handheld application. After receiving it, if there are some images it will call the Storage Manager to temporarily save them on the wearable memory. In any case it will call the Game Manager to show the game to the user.

The Game Manager component will show a notification to the user. Once he/she opens it, it will take care of showing the game content, the game options after that and finally the game outcome once the user selects the answer. The Game Manager will also take care of all timers (maximum time to read content and select an answer) and, if we have some images, to retrieve them calling the Storage Manager and at the end delete them.

At the same time as the game outcome is shown, the Game Manager will also call the Send To Mobile component to send the game result to the handheld application.



Part VI
Implementation: Architecture

15 Android Handheld

The handheld (smartphone/tablet) side of the application is built on top of the Android operative system, with support from API 19 (Android 4.4 KITKAT) to the current version, API 23 (Android 6.0 Marshmallow).

The “Android Wear” application, available on the Play Store, allows the user to connect the wearable to the handheld.

16 Android Wearable

The wearable (smartwatch) side of the application is built on top of the Android Wear operative system, with support to all versions available at the moment: from API 19 (Android 4.4 KITKAT Wear) to API 23 (Android 6.0 Marshmallow). The application look&feel adapts to the smartwatch shape (round or square).

17 External libraries

The application uses the following external libraries:

- **Android SQLiteAssetHelper** to easily ship the SQLite game database as a ".db" resource file
- **MPAndroidChart** to display the user points radar chart
- **Gson** to serialize information sent between the mobile and wearable devices

Part VII
Implementation: Persistent Data

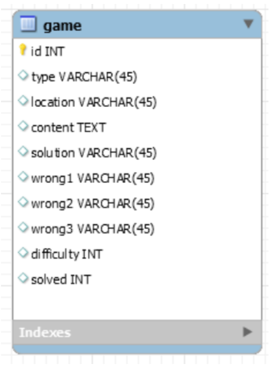
18 General Description

Persistent data is only stored by the handheld application. In particular:

- settings, such as games per day, difficulty, etc., are stored using the SharedPreferences function provided by Android via the SettingsFragment implementation
- the amounts of points gained by the user in every game category so far are also stored using SharedPreferences
- the available games are stored in a SQLite database

19 SQLite Game Database

The database is composed of only one table, "Game", that stores the information about the available games. The table is shipped with the application as-is and the only run-time edit to its content is the boolean flag "solved".



```
id INT
type VARCHAR(45)
location VARCHAR(45)
content TEXT
solution VARCHAR(45)
wrong1 VARCHAR(45)
wrong2 VARCHAR(45)
wrong3 VARCHAR(45)
difficulty INT
solved INT
```

- **id**: the unique id of the game
- **type**: the game category, i.e. the fully qualified name of the Java class representing it
- **location**: the location of the game, it can for example be "none" for games that have no location, "mall" for games to be shown when the user is inside a mall, etc.
- **content**: the game text, e.g. the equation the user needs to solve
- **solution, wrong1, wrong2, wrong3**: the answer options. Note that the current implementation is not very flexible (exactly four options) but the application manages the options array in a general way so a future update (e.g. move options in another table to offer a variable number of them for each game) would be quite easy to develop
- **difficulty**: the game difficulty
- **solved**: boolean telling if the user already solved the game or not

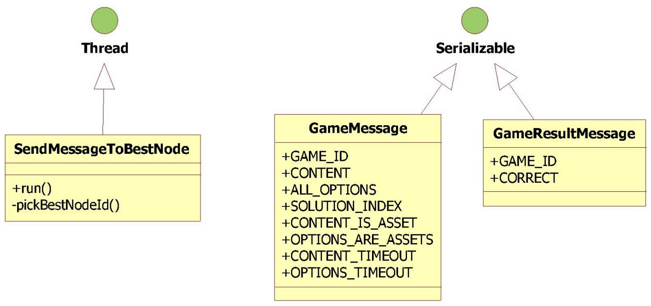
Part VIII
Implementation: Class Diagrams

The application is composed of three modules:

- **common**: shared by both the handheld and the wearable applications
- **mobile**: contains the functions for the handheld side
- **wear**: contains the functions for the wearable side

In the following sections there will be a detailed explanation of the implementation class diagrams. Note that green circles represent built-in Java/Android classes to better represent inheritance.

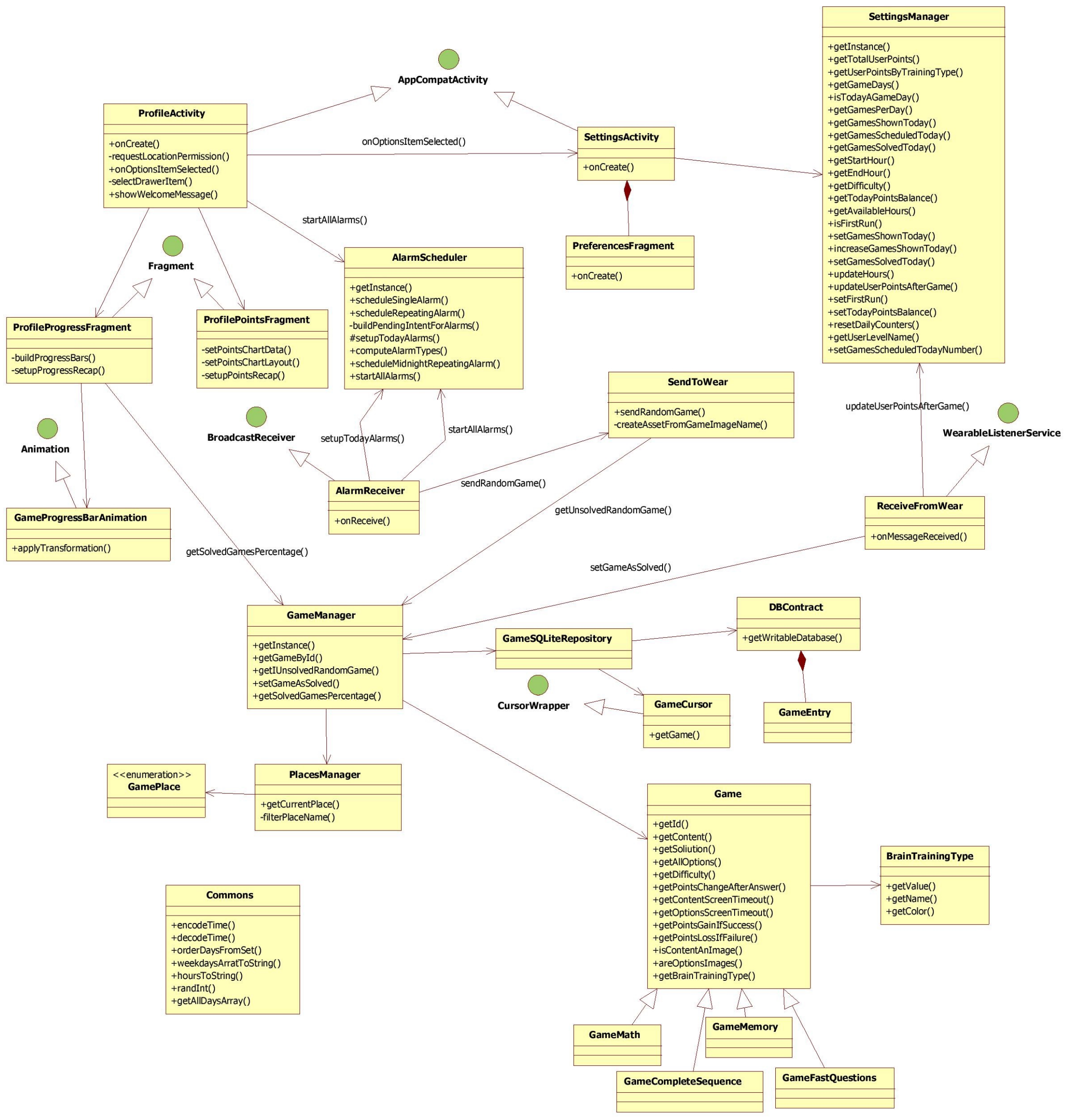
20 Common



The module just contains:

- **SendMessageToBestNode**, a Thread implementation that allows to send a message from the current device to the "best" connected node (handheld to wearable and vice-versa)
- two serializable classes that represent the messages sent between the handheld and the wearable:
  - **GameMessage** contains the game description (content, options, etc.)
  - **GameResultMessage** contains the game outcome description (game ID and a boolean to tell if the user answered correctly)

21 Handheld



21.1 User Interaction

The main activity of the application is **ProfileActivity**. It is in charge of showing the user points information (via a radar char and a textual recap) using the **ProfilePointsFragment** and his/her progress (a progress bar for each difficulty level and a textual recap) using the **ProfileProgressFrag-**

ment, offering navigation between the two views via the drawer pattern. At the first application run, the ProfileActivity welcomes the user with a brief introduction, asks for dynamic location permissions (on API >= 23) and starts the midnight alarm for game scheduling. The activity's "app bar" allows to open the Android share chooser and to reach the SettingsActivity.

The **SettingsActivity**, with its **PreferencesFragment**, allows to manage the user settings, such as game difficulty, games per day, etc. It interfaces with the **SettingsManager** to save and store the information.

21.2 Games Scheduling

The **AlarmScheduler** is in charge of scheduling the alarms. It is called on the first run of the application to start the repeating midnight alarm (used to trigger the computation of the daily games) and by the AlarmReceiver to schedule the daily game alarms (it queries the SettingsManager for the user preferences and schedules the games according to the selected time slot and frequency). Note that the scheduler makes sure that the daily alarms are evenly distributed in the given time interval (e.g. distance between two games is at least 20 minutes), while keeping their randomness.

The **AlarmReceiver** is a BroadcastReceiver that receives both the midnight alarm and the games alarms. When the former is received, it simply calls the AlarmScheduler to setup the daily alarms. When the latter is received, it calls the SendToWear component to send a random game to the wearable device.

The **SendToWear** class takes care of sending a random game (obtained querying the GameManager) to the wearable device. If the selected games contains some images (as content and/or options) it manages their transmission as well.

The **ReceiveFromWear** class listens for messages (game outcomes) coming from the wearable device. Once it receives one, it calls the SettingsManager to update the user points and, if the outcome is positive, the GameManager to set the game as solved.

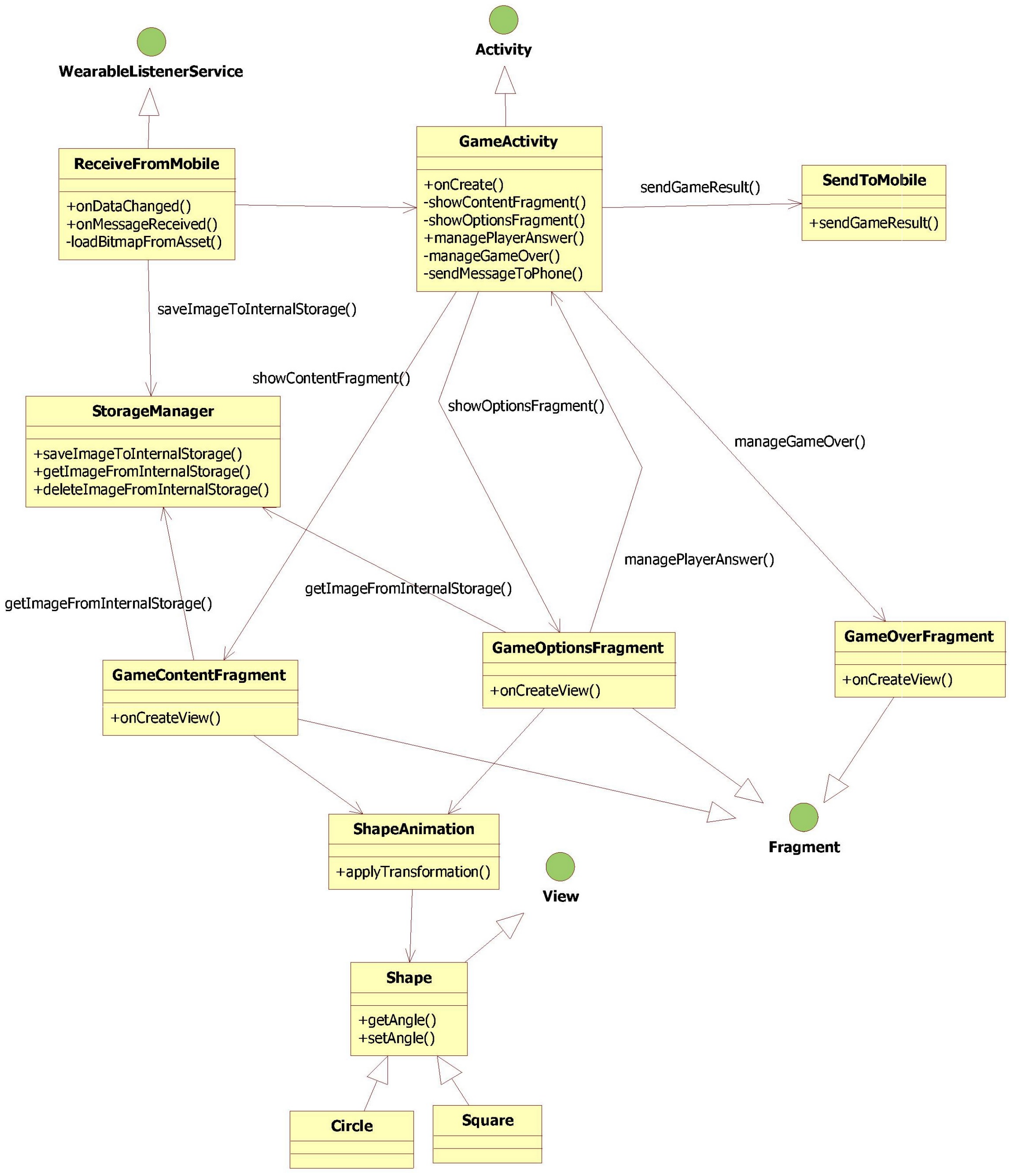
The **GameManager** manages the SQLite game database (via **GameSQLRepository**, **DBContract**, **GameEntry** and **GameCursor** classes), offering methods to get a random unsolved game, to set a game as solved and so on. It uses the abstract class Game to return the results. The GameManager also calls the PlacesManager to offer, if possible, a location-aware game.

The **PlacesManager** interfaces with the Google Places API to get the current user position (translated into a **GamePlace**, used by the GameManager to query for localized games in the database).

The abstract class **Game** and its implementations like **GameMemory**, **GameMath**, etc. represent a game entity. It offers methods to get the main properties like content and options, but also some helper methods to get the game points, timeouts and images (if any). Every game implementation has a specific **BrainTrainingType** (e.g. GameMath has Calculation).

Finally, **Commons** is a class that contains general methods used throughout the applications (arcs are omitted in the diagram for clarity).

22 Wearable



On the wearable everything starts when data (text and/or images) is received from the phone by the **ReceiveFromMobile** service. Once this happens, the service calls the **StorageManager** to temporarily store the images (if any) on the device memory and creates a notification setting as pending intent the only activity in the module, GameActivity.

The **GameActivity** receives the game from ReceiveFromMobile and starts by displaying the **GameContentFragment** for the amount of seconds specified in the game message. This fragment is in charge of displaying the game content (if textual by simply setting it in a TextView, if image by retrieving it from StorageManager and displaying it in an ImageView) and the timeout animation via the **ShapeAnimator** that controls a Shape, **Circle** or **Square** (based on the actual shape of the wearable).

After the timeout expires, GameActivity shows the **GameOptionsFragment**. This fragment displays the given answer options and starts the ShapeAnimator like in the previous case.

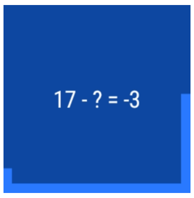
When the user selects an answer or the timeout expires, the GameActivity shows the **GameOverFragment** that will display the outcome ("Correct", "Wrong" or "Time's Up") to the user.

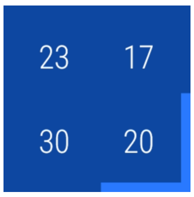
At the same time GameActivity calls the **SendToMobile** class in order to send the game result to the handheld device.

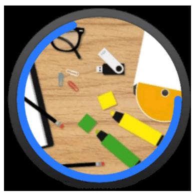
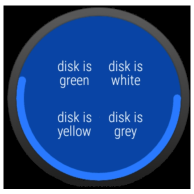
Part IX
Conclusion

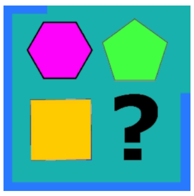
23 Image Gallery

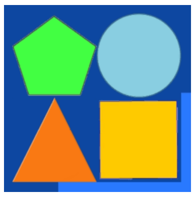
Follows a short image gallery of the final result (both wearable and handheld devices):

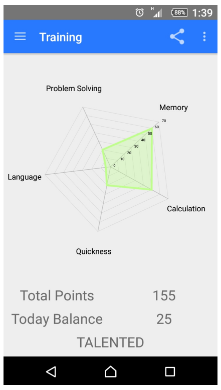




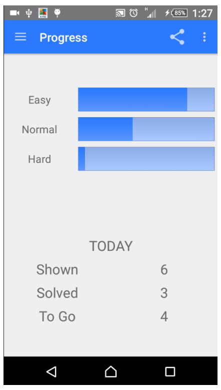







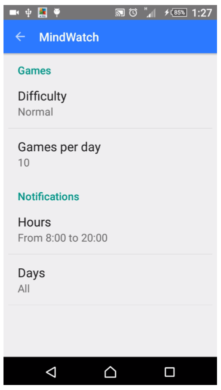

Training

Total Points 155
Today Balance 25
TALENTED


Progress

TODAY

Shown	6
Solved	3
To Go	4


MindWatch

Games

Difficulty
Normal

Games per day
10

Notifications

Hours
From 8:00 to 20:00

Days
All

24 Challenges and Critical Aspects

The main challenge and critical aspect of the project was to provide suitable means of displaying and controlling games on a smartwatch. Wearable devices are meant to display very small amounts of text and require as few interactions as possible, and this of course goes against the idea of games in general. For this reason many limitations on proposed games had to be imposed, in order to minimize text, required time and interactions. Nonetheless, the application still may show more text, take more time (especially on higher difficulty settings) and require more interactions than the average wearable application, but that is, unfortunately, unavoidable.

During the implementation phase, the main challenge was in the communication between handheld and wearable application. While for a normal application with smartwatch support displaying notifications on the wearable devices is almost trivial, the need to have a wearable-only notifications and to send images alongside the games requires to setup senders and receivers using the Wearable Data Layer API.

25 Future Directions

The wearable application may be improved adding more functions (like speech recognition to record the user answer) and providing more game types (e.g. more dynamic interactions).
The handheld application can be extended by providing a nicer look&feel and more information about the user progress (e.g. a history of performance for each day).

26 Bibliography

- Android Developer website for documentation and examples on Android programming
- Design and Implementation of Mobile Applications course website (prof. Baresi) for explanations and examples on Android development