

Deminder



Project for the course

SOFTWARE ENGINEERING

at

Duale Hochschule Karlsruhe

by

Natalie Busam

Tillmann Nünninghoff

Lea Wegner

Thomas Malina

12. Juny 2019

Docent:	Ms Berkling
Course:	TINF17B4
Academic year:	2018/2019

Table of Contents

W1 - Creating the idea of Deminder	3
W2 - Team roles, tools and issue list.....	4
W3: SRS and Use-Case Diagram	5
W4: promised Use Cases and Mock ups.....	6
W5: Features	8
W6: Classdiagram.....	9
W7: Scrum.....	10
W8: Retrospective.....	12
W9: MVC	12
HW 11: Midterm exam	13
Semester II - Week 3	15
Function Points	15
Calculation	15
Result.....	16
SEMESTER II – WEEK 4.....	17
Our third test form.....	18
S2W5: Refactoring	18
Retrospective 2.....	19
S2W6: Design patterns.....	20
S2W8: Metrics	22
Installation (and user tests).....	23
Final.....	24
Promotion video	26

W1 - Creating the idea of Deminder

Do, 04 Okt 2018 12:38:44

Every student knows the stress exams can cause you, but it is not just the learning that makes them such a struggle. There are so many deadlines you have to keep track of, the exam dates, the papers you need to finish and not to forget the things you have to do outside of university such as appointments you have to attend or emails you have to write.

This is exactly what our App **Deminder** is about. Take away that stress of having to keep in mind all your deadlines, allowing you to focus on the important things. It will give you an overview over your deadlines, allowing you to arrange and sort them to your liking and will remind you about upcoming deadlines. Additionally you can add tasks to each deadline. These can then be checked to let you keep track of your progress for each deadline. This allows you to not only use **Deminder** as a tool helping you not to forget any important upcoming dates but also to organize and track your progress for each of their tasks, making it an ideal helper not only for students but anyone that wants to stop to worry about their schedules.

Future vision:

We have some ideas that are probably out of scope for our current semester and will either be worked on if there is extra time left or tackled in the next semester.

1. Implementing a server:

We want to build a server that allows users to interact with each other effortlessly. You will be able to create groups in which you share deadlines, create deadlines for others (for example tasks for your family or room mates) and all deadlines will be stored on the server instead of your phone.

2. More Widgets:

We want to allow users to place more, unique widgets to their phones. For example a widget always showing the top priority deadline or a widget showing a day/week/month/ overview.

3. More notifications:

We do not want Deminder to turn into an "alarm app", but notifications help you to keep track of what you need to do. This is more of a design task as we want to make sure to keep these criterias in mind. Things we thought about are for example snoozing, alarms right before the deadline.

W2 - Team roles, tools and issue list

Fr, 12 Okt 2018 16:13:04

Our team roles and responsibilities are the following (RUP):

- Requirements Specifier: Natalie
- Analyser & Designer: Tillmann & Lea
- Implementer: Lea, Natalie, Thomas, Tillmann (all of us)
- Tester:
 - Test Manager & Analyst: Natalie
 - Test Designer: Tillmann
- Deployment Manager: Natalie
- Project Manager: Thomas
- Tool Specialist: Natalie & Lea
- Configuration and Change Management:
 - Configuration Manager: Thomas
 - Change Control Manager: Lea

We will use *Android Studio* as IDE.

Technologies

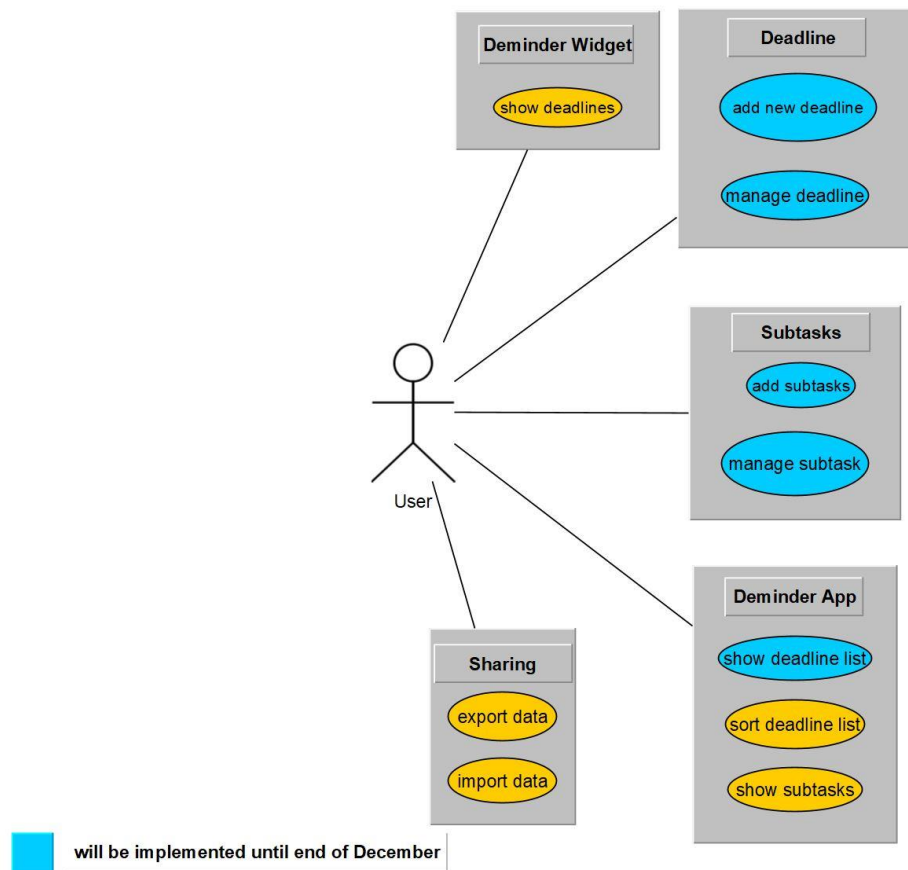
We will use Java to program the android app. We will most likely save the deadlines in an ics format. We currently don't plan to also support iOS. Also we will use Git and GitHub for software versioning, the link to our repository: <https://github.com/Kalkihe/Deminder>.

We are using *YouTrack* as project management tool. Here is the link to your current issues-list: <https://deminder.myjetbrains.com/youtrack/issues>.

W3: SRS and Use-Case Diagram

Fr, 19 Okt 2018 14:57:37

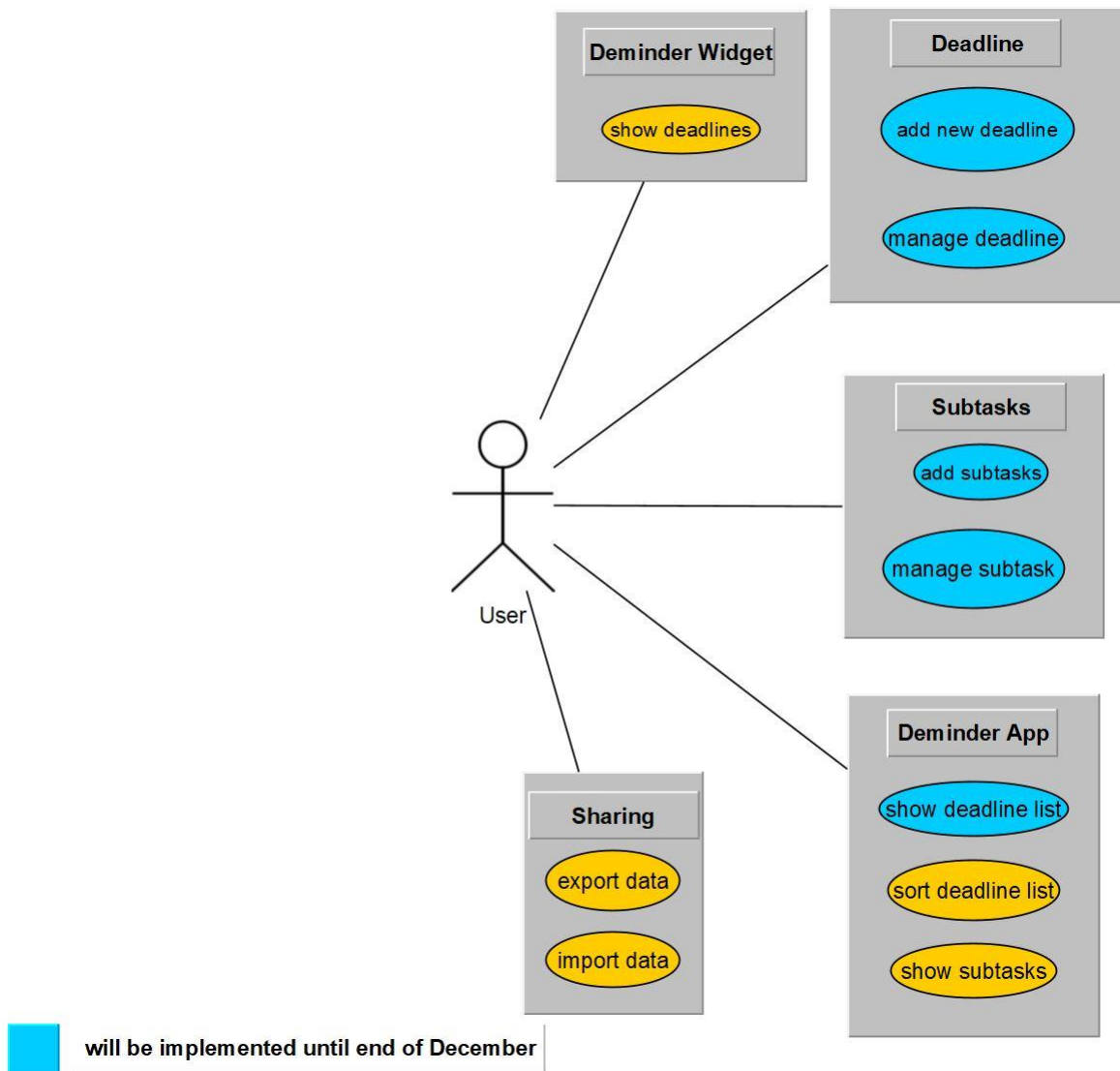
Today, we'd like to present you our Use-Case Diagram and our [Software Requirements Specification](#).



W4: promised Use Cases and Mock ups

Mo, 22 Okt 2018 16:12:20

Today, we'd like to present you our updated Use-Case Diagram with the Use-cases promised for the end of december.



And we want to show you our Mock ups :)

We specified two Use cases, [here](#) you can see our "add Deadline" Use case :

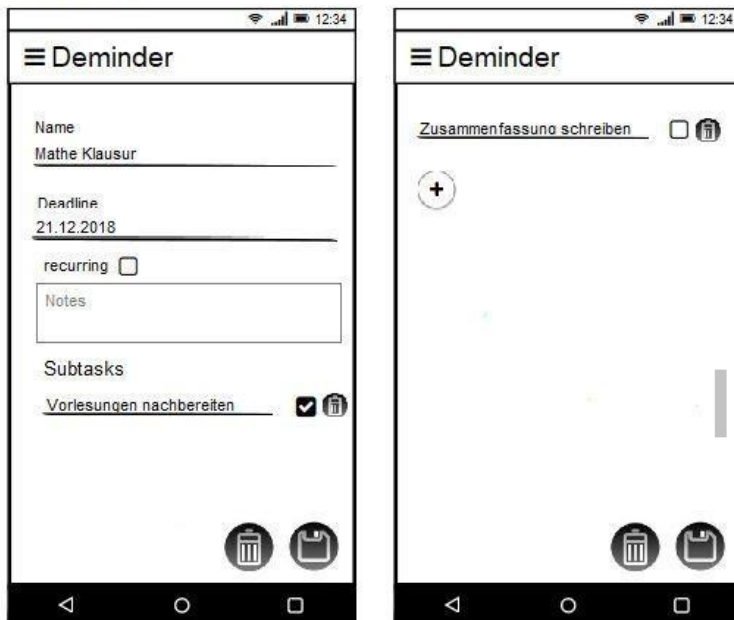
Phone Screen



The screenshot shows the 'Deminder' app interface. At the top, there's a header with a hamburger menu icon and the title 'Deminder'. Below the header, there's a form with the following fields: 'Name' (text input), 'Deadline' (text input), 'recurring' (checkbox), 'Notes' (text area), and 'Subtasks' (a list with a '+' button to add new subtasks). At the bottom right, there are two circular icons: one with a trash can and one with a document. The Android navigation bar is visible at the very bottom.

and [here](#) you can see our "manage Deadline" Use case.

Phone Screen



Two screenshots of the 'Deminder' app are shown side-by-side. The left screenshot shows the 'add Deadline' form with the following data: Name: 'Mathe Klausur', Deadline: '21.12.2018', recurring: unchecked, Notes: empty, and Subtasks: 'Vorlesungen nachbereiten' (checked). The right screenshot shows the 'manage Deadline' screen, which displays the same deadline as a list item: 'Zusammenfassung schreiben' (unchecked). Below the list item is a '+' button to add new subtasks. The same trash and document icons are at the bottom right of each screen.

Thanks for looking :)

Best regards

Deminder(Lea)

W5: Features

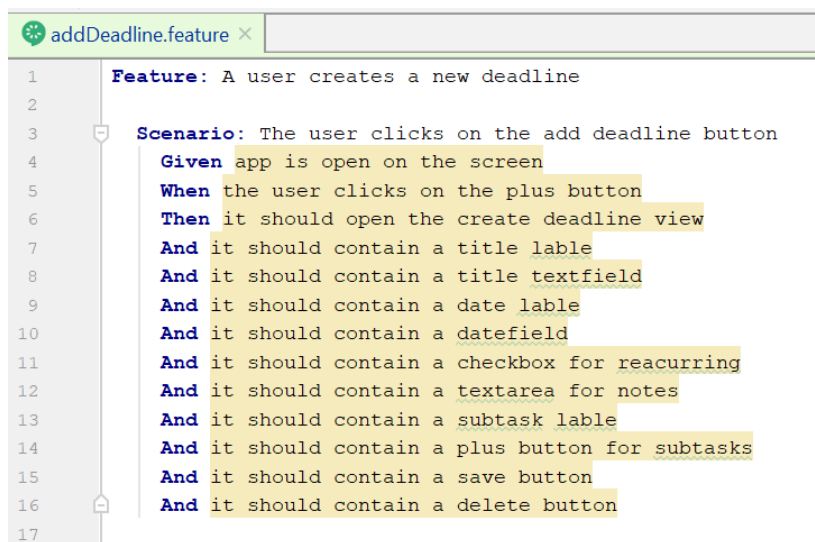
Di, 30 Okt 2018 14:40:39

This week we created two feature files for the two use cases we defined in [week 4](#). You can look at them in git or in the corresponded use case file (see last weeks post).

Git:

- [add deadline](#)
- [manage deadline](#)

And here is a screenshot showing that we have syntax highlighting running in our IDE:



The screenshot shows a code editor window titled 'addDeadline.feature'. The code is written in Gherkin syntax and is highlighted with colors: blue for keywords (Feature, Scenario, Given, When, Then, And), yellow for text, and green for strings. The code is as follows:

```
1 Feature: A user creates a new deadline
2
3 Scenario: The user clicks on the add deadline button
4   Given app is open on the screen
5   When the user clicks on the plus button
6   Then it should open the create deadline view
7   And it should contain a title lable
8   And it should contain a title textfield
9   And it should contain a date lable
10  And it should contain a datefield
11  And it should contain a checkbox for reacurring
12  And it should contain a textarea for notes
13  And it should contain a subtask lable
14  And it should contain a plus button for subtasks
15  And it should contain a save button
16  And it should contain a delete button
17
```

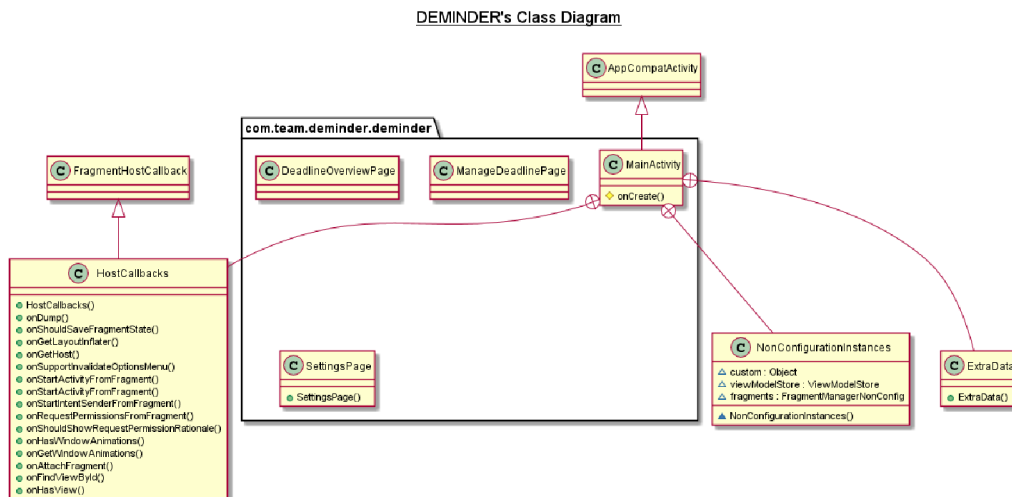
Unfortunately, there is no possibility to have auto completion for predefined sentences within our IDE (Android Studio). We did a lot research on that but weren't able to find any solution.

W6: Classdiagram

Mo, 12 Nov 2018 04:04:27

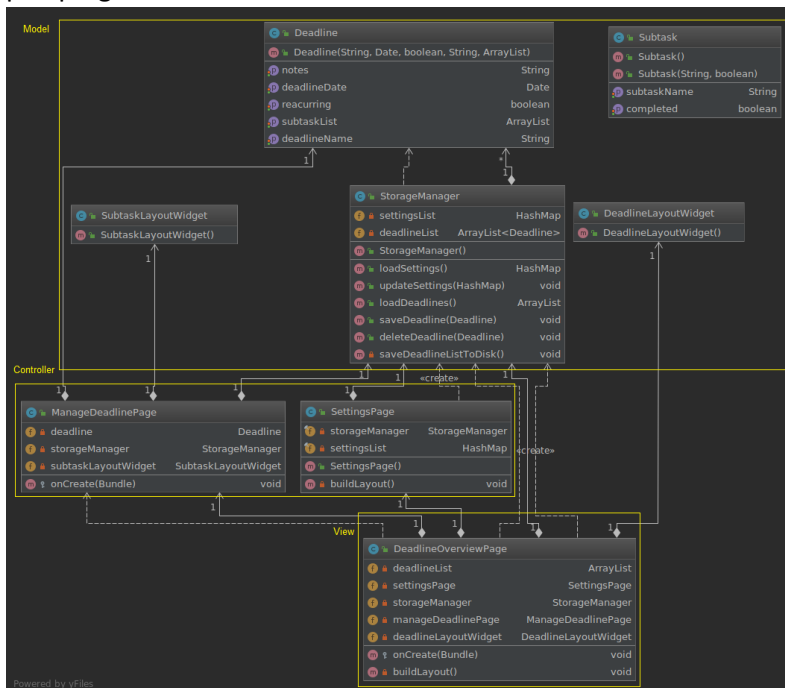
This week we created our class diagram. We had to download the plugins "Sketch it" and "PlantUML" and install them in Android Studio. I also had to install "Graphviz" on my laptop (Windows).

This is our result:



UPDATE 29.11.2018:

While using the tools, we had some issues because we created sub-packages within our main-package. Unfortunately, our tools were unable to create a class diagram containing all sub-packages. Because of this, we opened our project in IntelliJ and created the following class diagram with the pre-plugin in IntelliJ.



Please note that we already edited the picture to show which classes are part of Model, View or Controller.

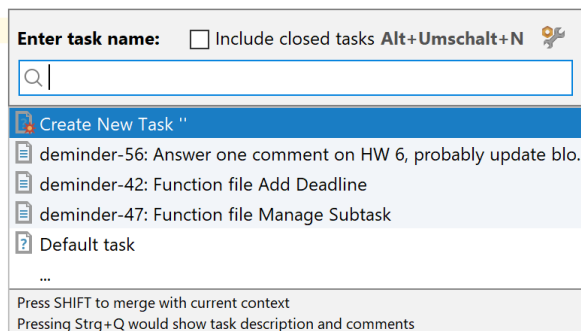
W7: Scrum

Do, 15 Nov 2018 17:24:15

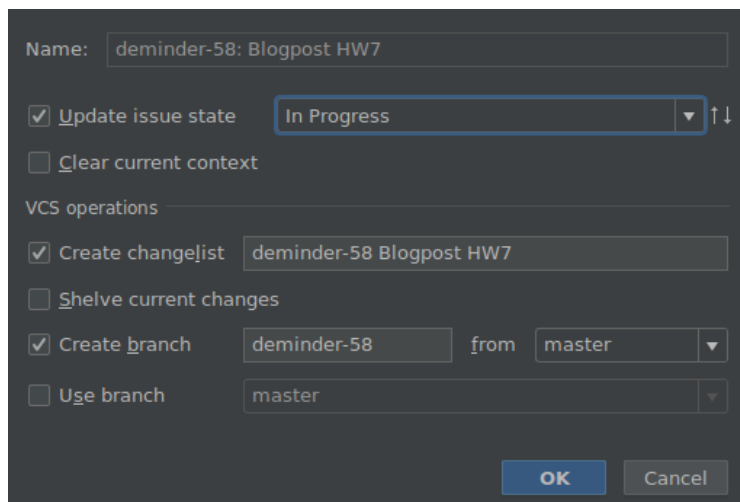
Today, we want to introduce our project management environment to you. First of all, we are using [YouTrack](#) as project management tool.

Of course, we can access our task list in our IDE, Android Studio, as seen in the following screenshot:

```
public SubtaskLayoutListElement() {  
}
```

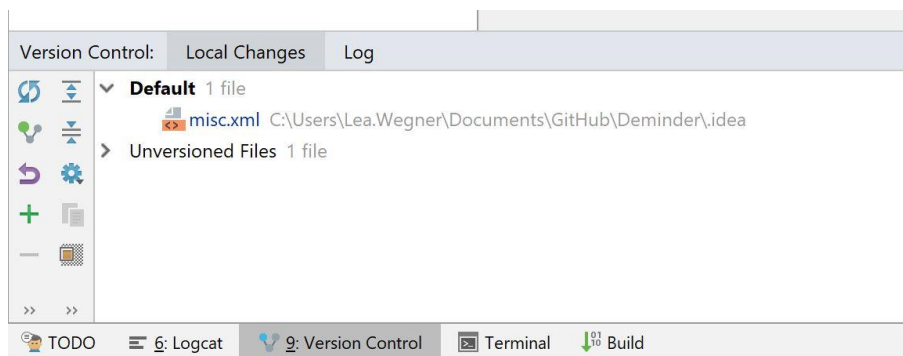


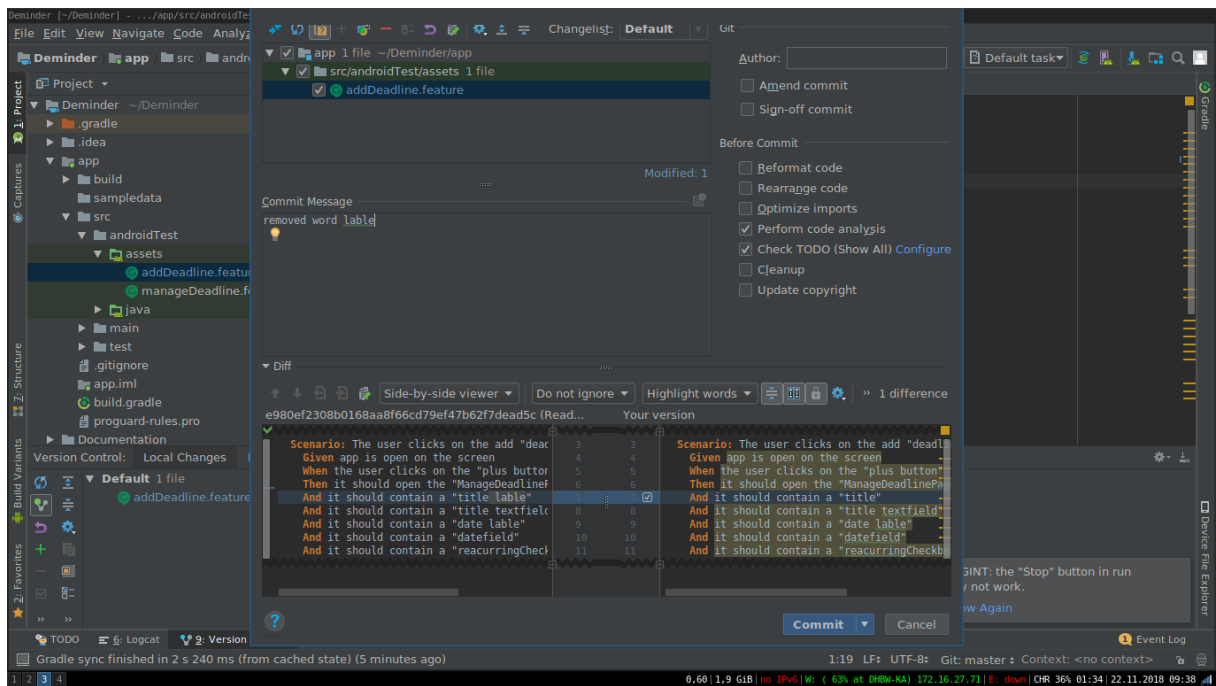
Furthermore, we can track the time one of our team is working on a specific task. For this, we double-click on a task in the above shown list and set the status to "In Progress".



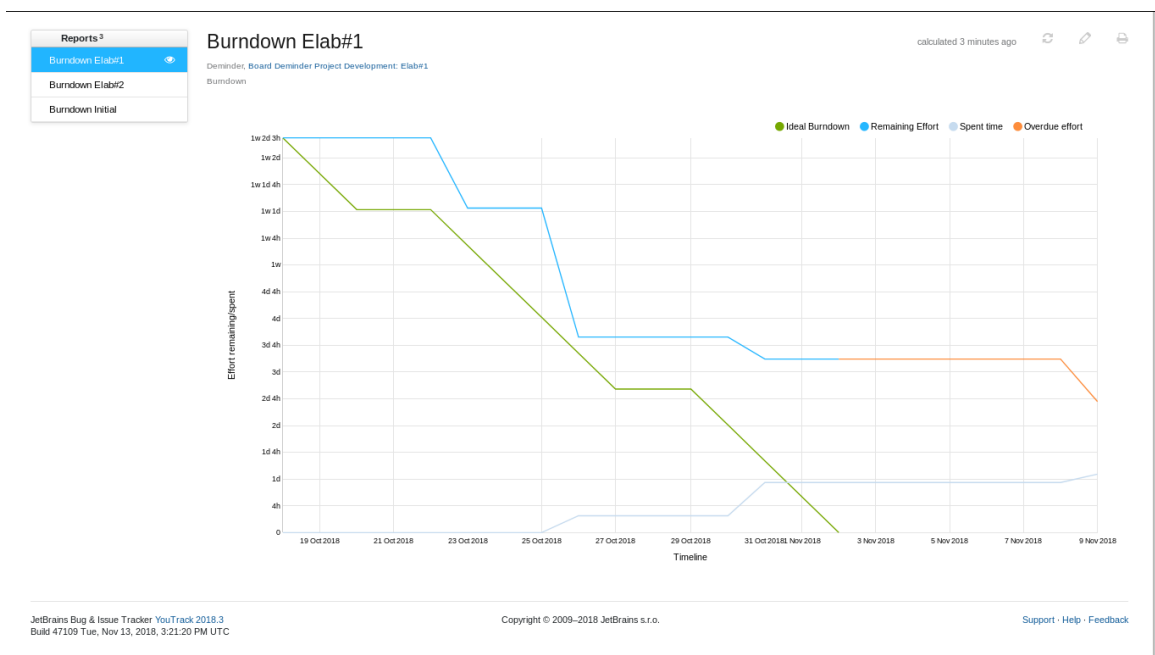
We have configured Youtrack that it automatically tracks the time of a Issue being in Progress. To stop time tracking, we just have to set the issue to a different state than "In Progress".

We can contribute to our git repository via Android Studio directly, too.





Here is a screenshot of one of our sprints. You can click on the picture to get to the [report](#).



As you can see in the list of our [issues](#), our tasks are assigned to one of our team members, have estimated time and are tagged according to RUP. Also, tasks that are associated with a use case, are tagged with it.

W8: Retrospective

Do, 22 Nov 2018 12:21:22

This week, we had a retrospective about our teamwork. We thought about things that went well in the past, what can be done better in the future and what our goals for our improvement are. Now, we want to share those things with you.

What went well:

- Communication within our team, everyone knows what our current status is
- Diversity within our team, everyone does what he can do best

What can be done better

- Criticism vs. letting people do their job
- Motivation

Goals for improvement in the future

- Considerate criticism and giving feedback to criticism
- Respect responsibility assignments according to RUP
- Use the time given in the software engineering lecture
- Having pizza at scheduled working meetings

W9: MVC

Do, 29 Nov 2018 14:52:42

This week, we had to choose our MVC-Framework. Due to the fact that we're creating an Android App, we are using MVC inherent.

Plus, we wrote our SAD. You can find it [here](#). You can find our current class diagram with marked MVC regions in this document.

W10: Cucumber running

Sa, 08 Dez 2018 20:33:23

This week we have launched the Cucumber tool for testing our app.

Here is the video:

https://www.youtube.com/watch?time_continue=2&v=jgk37rlxd30

HW 11: Midterm exam

Di, 11 Dez 2018 14:49:27

Today, we want to sum up the work we did so far.

Requirements

Use Cases	Use-Case-Diagram Add Deadline Manage Deadline Add Subtask Manage Subtask Show Deadline list
Software Requirement Specification	SRS
Test Cases	Add Deadline Manage Deadline Add Subtask Manage Subtask Show Deadline list
Test log	Screenshot
Functional test	Cucumber Test running

Project management

RUP gantt chart	Gantt chart Team member hours
Burndown charts (* see note)	Initial Elab#1 Elab#2 Const#1 Const#2

Ability to execute

Code	Package with all source files
------	---

Quality

Architecture	MVC
SAD	SAD

Configuration Management / Environmental Setup

Automated Testing	Cucumber tests running
-------------------	--

Other

Presentations	Midterm presentation
---------------	--------------------------------------

**Note to burndown charts: We know that our burndown charts don't look like they should. We had this issue going on in the semester and finally managed to find out what the problem is (that issues were in the wrong sprint). We dug deep into every issue on our board and checked when it was due, when it was marked as done etc. and placed it into the correct sprint according to the dates of the sprint. Unfortunately, the reports are still being created incorrect. For example, in Elab#1 it seems that there is one issue with 25 minutes estimated time. But we checked all issues in that sprint three times and couldn't find any matching issue. For us, it seems to be Youtracks fault.*

Semester 2: Welcome back

Mi, 03 Apr 2019 12:52:43, deminder18, [category:uncategorized]

The new semester has started and with that we are back to work. Last semester we finished all use cases promised for december as specified [here](#).

This semester we will tackle the rest of the use cases as specified [here](#). We have not made any changes to the team or general vision of the projekt however we made some minor changes to the planned feature set of deminder. We do not want to support recurring deadlines and and scraped the automatic calculation of a deadline priority. This was done after analysing our burndown chart from last semester. We realized that we had far less time to spend on implementation than originally imagined so we cut these features to ensure we have time for the most important features of our project.

You can find all use cases [here](#) and our updated SRS here.

S2 W2: Risk

Mi, 10 Apr 2019 10:55:08, deminder18, [category:uncategorized]

This week, we thought about the risks regarding our project. You can have a look at them [here](#).

Furthermore, for UC-planning, we created some statistics about the use cases we implemented last semester.

[Overall time spent per use case](#)

[Time spent per use case for implementation](#)

[Time spent per use case for documentation](#)

[Time spent per use case for testing](#)

Semester II - Week 3

Mi, 17 Apr 2019 12:59:59

Hi everyone!

This week we worked on the topic function points in order to be able to estimate the time that is needed in future task.

First of all our done Use-Cases:

- [Add new deadline](#)
- [Manage deadline](#)
- [Add subtasks](#)
- [Manage subtasks](#)
- [Show deadline list](#)

And upcoming Use-Cases:

- [Show subtasks](#)
- [Sort deadlines](#)
- [Import Deadline](#)
- [Export deadline](#)
- [Widget](#)

Function Points

Function points are assigned to an issue in order to estimate the time for future tasks. This is done using experience from previous tasks. In real projects, this is particularly important or even necessary for cost estimation.

Calculation

For this we used the following tool: [Tiny Tools](#)

The Complexity Adjustment Table was filled out specifically for our project.

Complexity Adjustment Table

ITEM	COMPLEXITY ADJUSTMENT QUESTIONS	SCALE					
		No Influence 0	1	2	3	4	Essential 5
1	Does the system require reliable backup and recovery?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
2	Are data communications required?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
3	Are there distributed processing functions?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
4	Is performance critical?	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
5	Will the system run in an existing, heavily utilized operational environment?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
6	Does the system require on-line data entry?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
7	Does the on-line data entry require the input transaction to be built over multiple screens or operations?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
8	Are the master files updated on-line?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
9	Are the inputs, outputs, files or inquiries complex?	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
10	Is the internal processing complex?	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
11	Is the code to be designed reusable?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
12	Are conversion and installation included in the design?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
13	Is the system designed for multiple installations in different organizations?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
14	Is the application designed to facilitate change and ease of use by the user?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Domain Characteristics Table 1 ED Calculation

Result

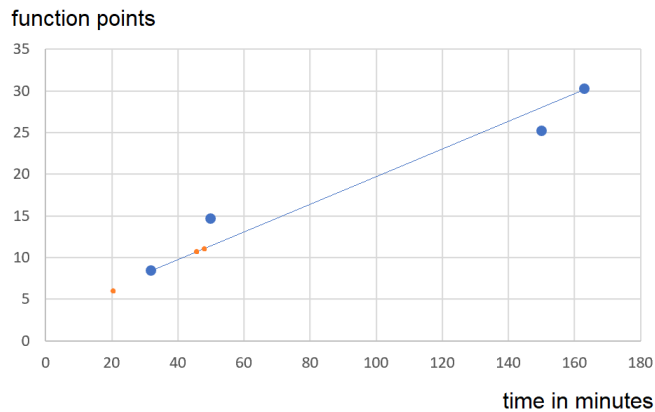
[Open PDF to see our fp-calculation](#)

Here is our table for spendend and estimated time:

Use cases first semester	Minutes spent on implementation	Function points
Add deadline	150	25,2
Add Subtask	32	8,4
Manage deadline	163	30,2
Manage subtask	50	14,64

Use cases second semester	Minutes estimated for implementation	Function points
Export Data	50	11,76
Import data	50	11,76
Show subtasks	20	5,88
Sort deadline list	45	10,9

Depending on this, the line chart:



Thank you for reading and feel free to comment!

SEMESTER II – WEEK 4

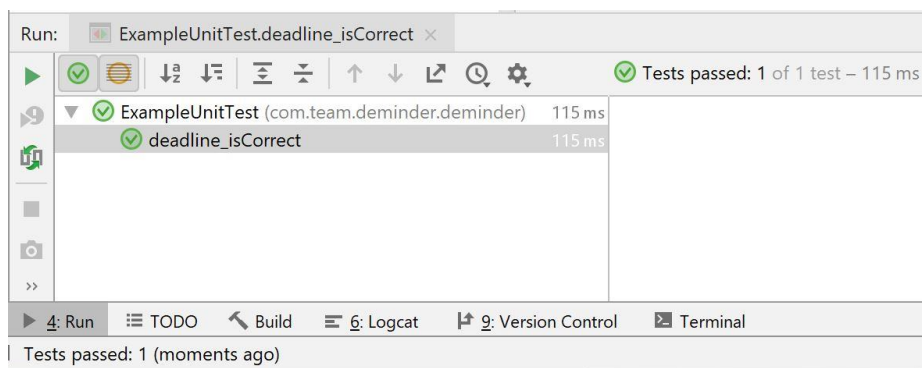
Do, 25 Apr 2019 14:08:48

Hi guys,

this week was all about testing.

We use JUnit for our testing, you can find our tests [here](#) and our gradle file that adds the dependency here. Since we are developing an Android application we have many UI tests (feature tests) and very little functionality to test (unit tests). Besides little unit tests this also means we don't worry about real data and objects in our tests so we chose not to use fakes or Mockito.

The following JUnit test checks whether the name, date, notes and list of subtasks are saved correctly. It follows the "addDeadline" activity diagram. The purpose of this is to ensure that the user is provided with the requested information.



Our Test-Plan can be found [here](#).

We also set up Travis Ci with Coveralls this week. Travis Ci is pretty much a plug and play tool for CI. It runs all Tests and builds our project and informs us when something went wrong. While Travis Ci is usually, as I said already, super easy to set up it has only partial Android support. The Problem is that you need to have a running Android emulator or device to be able to run the UI based tests (feature tests).

Here is our [travis.yml](#)

And our master [build.gradle](#)

And our [build.gradle](#)

You need these 3 files for the setup. Jacoco is the plugin to generate the coveralls statistics by the way.

Some tips for when set up Travis Ci for Android:

- The emulator Version is crucial. Some Versions are officially supported by Travis but still don't work
- Correct indentation is crucial when using .yml files!!!

- Some emulator versions just get stuck starting up (see next point) use comments to see if the problem is your code or the emulator version
- Ui tests are **very** sensitive to your emulators screen size. Be sure to set a fixed screen size in your travis.yml or Travis chooses randomly and you will have failing tests for no reason.
 - `emulator -avd test -no-window -skin 768x1280 &`
- Be sure to list the matching android sdk Version for whatever version your emulator uses in the components section of your Travis.yml if it isnt the same as your build version.

```
components:
// Emulator Setup
- android-22
- sys-img-armeabi-v7a-android-22
// Build Setup
- android-28
- build-tools-28.0.3
```

Our third test form

Do, 25 Apr 2019 14:22:57

Our third test form are user tests. These are needed to test the functions and usability of the app. For this we need randomly chosen, voluntary Android users who familiarize themselves with the app and give us detailed feedback on how intuitively they find our app "Deminder".

For our test-coverage you can have a look at our git-repository: [GitHub](#)

S2W5: Refactoring

Do, 25 Apr 2019 14:37:10

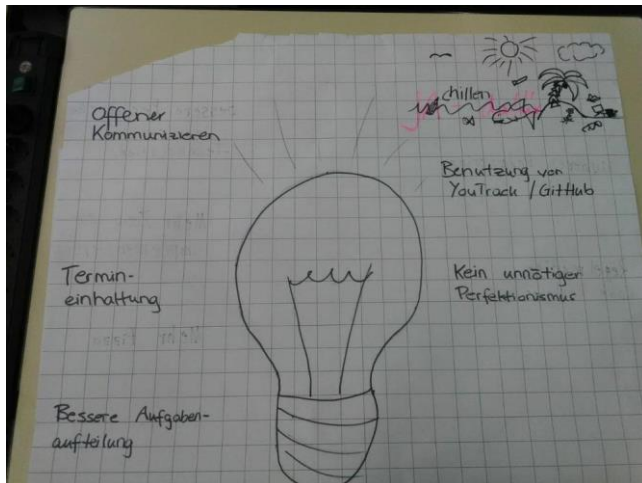
This week, we also did an exercise on refactoring code. Here are the links to the repositories of our team members on GitHub:

- Lea: <https://github.com/Luminca/SE-Refactoring>
- Natalie: <https://github.com/mc263/se-refactoring>
- Thomas: <https://github.com/Kalkihe/SE-Refactoring>
- Tillmann: <https://github.com/DrNuenninger/SE-Refactoring>

Retrospective 2

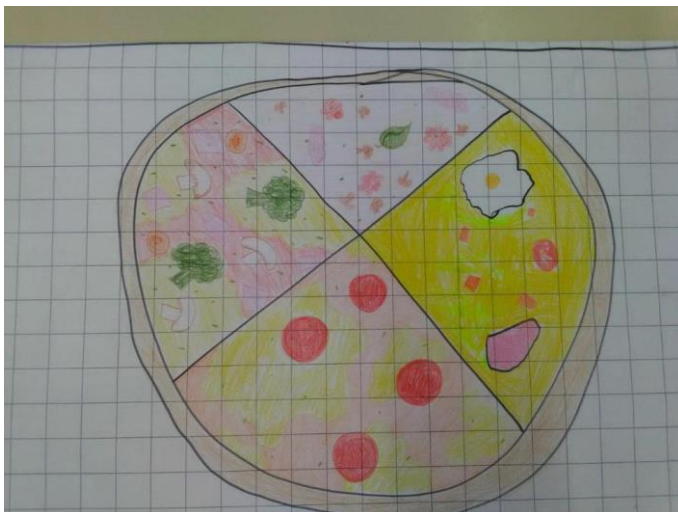
Mi, 08 Mai 2019 11:32:00

What have we learned since the last retro?



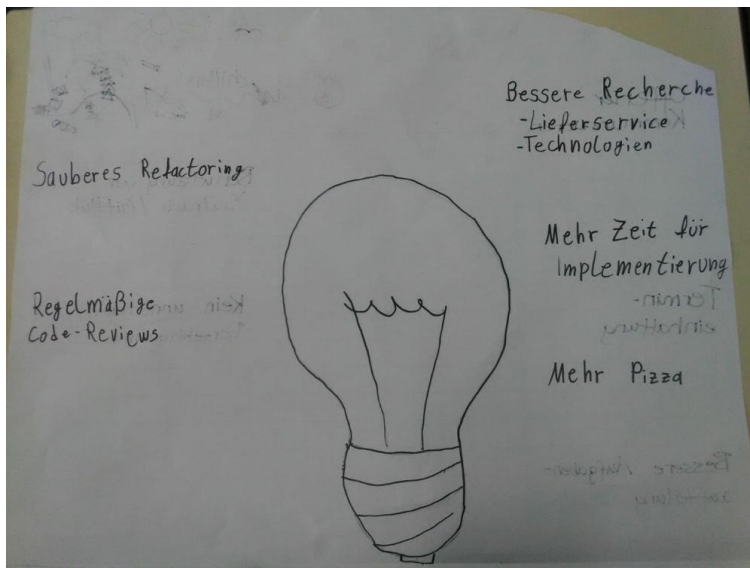
- Offener kommunizieren
- Termin-einhaltung
- Bessere Aufgabenaufteilung
- Chillen
- Benutzung von Youtrack/Github
- Kein unnötiger Perfektionismus

What characterizes our project group?



The four pizza pieces symbolize our team members. Each of us has designed his piece for himself - each of us is individual and has his own strengths. This is how we complete ourselves and function perfectly, as a team, together.

What else can we improve?



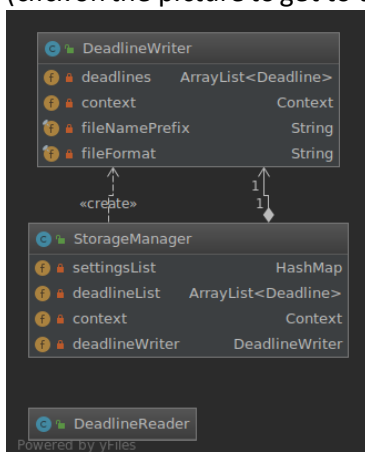
- Sauberes Refactoring
- Regelmäßige Code-Reviews
- Bessere Recherche
- Mehr Zeit für Implementierung
- Mehr Pizza

S2W6: Design patterns

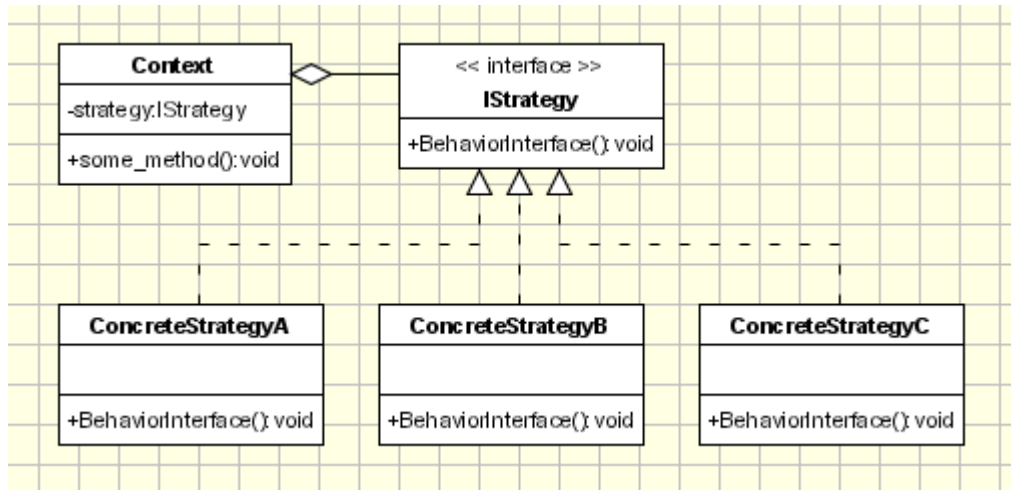
Fr, 17 Mai 2019 12:38:36

This week, we had a lesson about design patterns. We looked for a place in our architecture to implement a design pattern there.

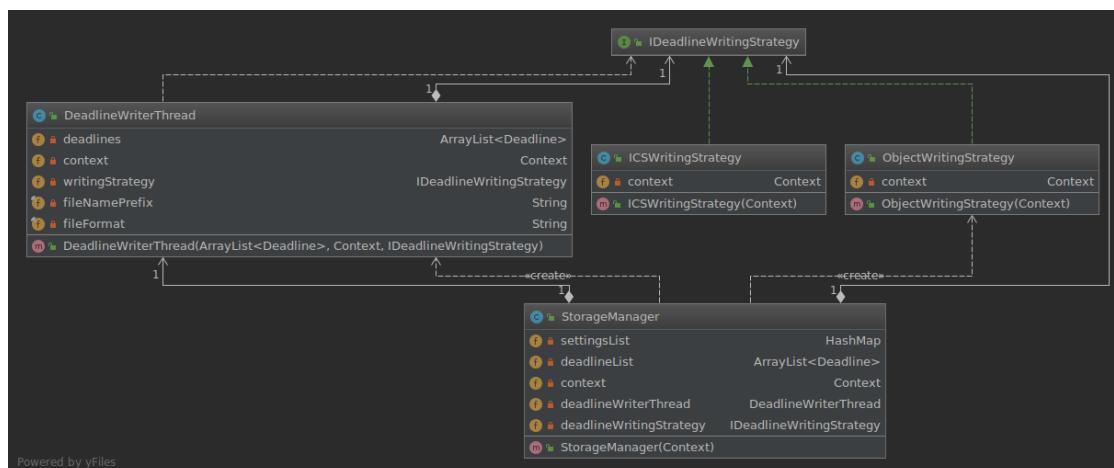
We choose the package "StorageManager". This part of our app takes care of saving the deadlines to the memory of your phone. When we started editing this package, the architecture looked like this (click on the picture to get to the source code of that state):



DeadlineWriter is a Thread, which takes care of writing files of deadlines to the memory. It currently uses a simple ObjectOutputStream. Later on, we may want to use a ICS library instead to write the deadlines in a format that can be read by other task apps, too. To implement or even switch between this ways of writing, we would have to change the class DeadlineWriter. We therefore decided to use the Strategy-Pattern:



This allows us to implement two or more methods of writing deadlines to memory and switch between them by simply injecting the right dependency to "DeadlineWriter". To state the use of "DeadlineWriter" more clearly, we renamed it into "DeadlineWriterClass". Take a look at the new class diagram of the package "StorageManger" (and take a look at the source code via clicking on the picture):



The strategy pattern allows us to switch between the different writing strategies easily and implement new writing strategies in the future without having to change the code of "DeadlineWriterThread".

Finally, you can have a look at our [overall class diagram](#) with the use of the pattern marked.

That's all folks!

S2W8: Metrics

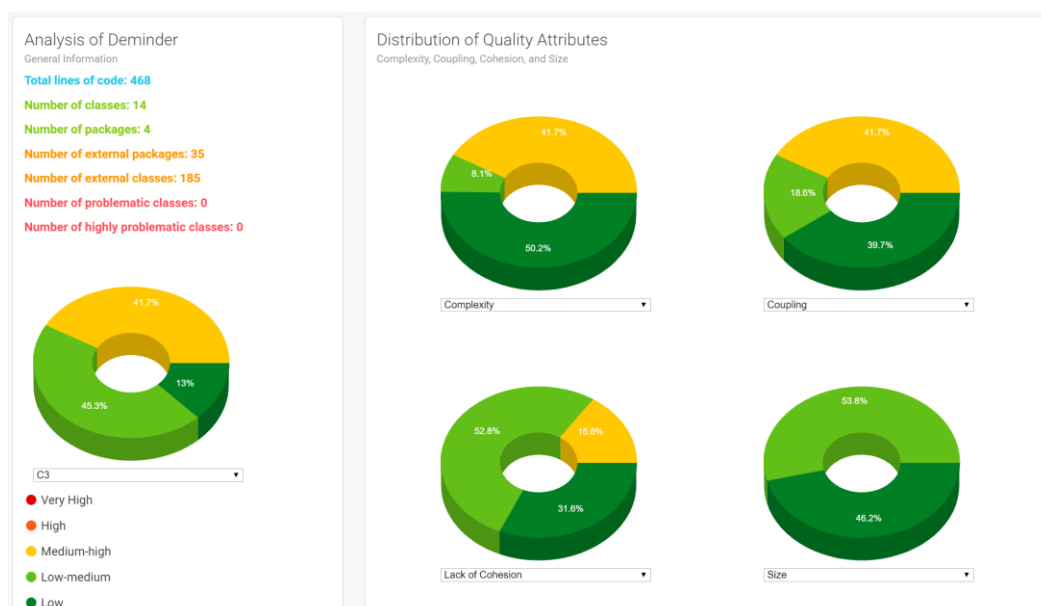
Mi, 22 Mai 2019 13:27:09

This week, we did some metrics on our code. We use the tool [CodeMR](#) for this. JetBrains offers a plugin for AndroidStudio. Through this, we are able to create metrics with one click.

We have run our metrics and they show this result ([click on the screenshot to get to the code in that state](#)):



We refactored our code to better the complexity and coupling. We like to mention that our 'worst' level is "medium-high" as you can see in the screenshot. This shows that our code was pretty decent in the first place. You can see our improvement here ([click on the screenshot to get to the code in that state](#)):



We achieved this improvements by putting some (mostly layout-) functions outside of the big classes into smaller classes. You may see that Lack of Cohesion now increased. The reason for this is that, while improving on our metrics, there was further work done regarding the functionality of our app. This caused the creation of some methods, which are called by android itself and not by us. This is the reason that CodeMR highlights that methods.

Now we like to explain the metrics we used:

Complexity: Implies how difficult to understand a class is and the number of interactions between its entities.

Coupling: Describes the relationship between two classes (A and B). It has a high amount for example if A calls on services of an object B or A has a method that references B (for example as a return type).

List of all classes (#13)

ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	COMPLEXITY	COUPLING	LACK OF COHESION	SIZE
1	ManageDeadlinePage					122	medium-high	medium-high	low-medium	low-medium
2	DeadlineOverviewPage					91	medium-high	medium-high	low-medium	low-medium
3	DeminderWidget					13	low-medium	low-medium	low	low
4	DatePickerFragment					11	low-medium	low-medium	low	low

As you can see, the class with the id 4 "DatePickerFragment" has a low-medium complexity and coupling. We are not going to change this because this is a library we use and didn't implement by ourselves.

Installation (and user tests)

Di, 11 Jun 2019 16:24:09

It's on! This week, we'd like to give you the opportunity to install the first version of our app! Simply follow the instructions given here: <https://github.com/Kalkihe/Deminder/releases/tag/1.1>

Furthermore, we'd appreciate it if you take a look at our app and test it! If you like to help us in bettering the app in the future, please make sure to download [this document](#), fill in the blanks and send it to us, best would be via [mail](#). Thank you!

Final

Mi, 12 Jun 2019 11:23:55

Requirements

Use Cases	Use-Case-Diagram Add Deadline Manage Deadline Add Subtask Manage Subtask Show Deadline list Widget Sort deadline list Import Data Export Data Show subtasks
Software Requirement Specification	SRS
Test Cases	Add Deadline Manage Deadline Add Subtask Manage Subtask Show Deadline list
Test log	Screenshot
Functional test	Cucumber Test running

Project management

RUP gantt chart	Gantt chart Team member hours Future long term planing (use cases second semester)
Burndown charts (* see note)	Initial Elab#1 Elab#2 Const#1 Const#2 Const#3 Const#4 Const#5 Tran#1 Tran#2 Tran#3

Function point calculation	Blog post Comparson
----------------------------	--

Ability to execute

Code	Package with all source files
Installation	Blog post

Quality

Architecture	MVC
SAD	SAD

Configuration Management / Environmental Setup

Metrics	Blog post
Risk management	Link to document
Automated Testing	Cucumber tests running Travis CI deploys and runs unit tests automatically with every push to our git repository
Used technologies	For used technologies and tools, see SRS
Patterns	See blogpost

Other

Presentations	Final presentation
---------------	------------------------------------

**Note to burndown charts: We know that our burndown charts don't look like they should. We had this issue going on in the semester and finally managed to find out what the problem is (that issues were in the wrong sprint). We dugged deep into every issue on our board and checked when it was due, when it was marked as done etc. and placed it into the correct sprint according to the dates of the sprint. Unfortunately, the reports are still being created incorrect. For example, in Elab#1 it seems that there is one issue with 25 minutes estimated time. But we checked all issues in that sprint three times and couldn't find any matching issue. For us, it seems to be Youtrack's fault. Unfortunately, besides knowing that, the problem continued to exist. We looked in to every issue again and the burndowns are still displayed wrong. It seems like Youtrack does something wrong here. Plus, Youtrack does not track the time for an issue for its assignee. Even if the assignee puts the issue in "In Progress", the time gets added for that user, that puts that issue in "To Verify" or "Done". What would a company that works with scrum masters do in that case? After this experience, we would definitely NOT recommend to use Youtrack if the time tracking has to be correct without much effort.*

*** We did all commit in a even amount, even if github tells you different. This comes from the fact, that you have a local git account on your machine. If you do not have the same username and mail*

adresse in that local account and on github, the commits you do on your local machine are not assigned to your github account. [Click here to view an example.](#) As you can see, Tillmann did this commit, but not with his github account DrNuenninger. Tillmann und Lea have this setting due to the fact that they use laptops from work with internal git settings. Thomas simply forgot to change his settings in first semester.

Promotion video

Mi, 12 Jun 2019 12:21:06, deminder18

Here we have a short promotion video of our app.

https://www.youtube.com/watch?v=wUqE_g_Brjg