



TO DO APP

Modul Project 226a

Exposee

In this document our working process, as well as the structure of this
of this project will be documented

Privat

Inform

This is a documentation about the structure and design of my application, so that you can get an insight and understanding of how it works.

Work environment

What	Tool
Class diagram	draw.io
Program and Coding	IntelliJ and Notepad++
Sequence Diagram	https://online.visual-paradigm.com/
JUnit Testing	IntelliJ
Version Control	GitHub

Plan

GANTT

		KW40	KW41	KW42	KW43	KW44
Inform						
	Projektidee	*				
	Entwicklungsumgebung					
Plan						
	UML	*				
	Use Cases		*			
	Classes and methods		*			
Implements						
	All Classes			*		
	All methods			*		
	Sequence Diagram				*	*
	UML updated					*
	Junit					*
	UI *maybe				*	
Documentation						
Planned	Extra features	Where I am *				

Tasks

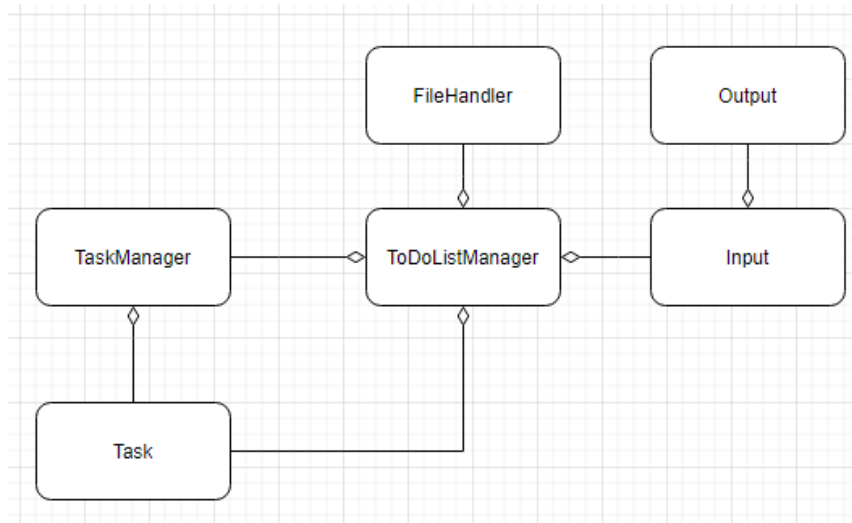
What	Time in min	Planned Date	Finished Date
GitHub development environment	15'	05/10/2021	05/10/2021
Create UML and show it to the teacher	25'	05/10/2021	05/10/2021
Create classes and methods	15'	05/10/2021	05/10/2021
Task			
Variables	5'	05/10/2021	21/10/2021
Getter & setters	1'	05/10/2021	21/10/2021
ToDoList methods			
addTask	10'	26/10/2021	21/10/2021
editTask	20'	26/10/2021	21/10/2021
checkId	5'	26/10/2021	21/10/2021

deleteTask	2'	26/10/2021	21/10/2021
markTaskAsDone	2'	26/10/2021	21/10/2021
sortByDate	15'	26/10/2021	21/10/2021
sortByProject	20'	26/10/2021	21/10/2021
isDateValid	10'	26/10/2021	21/10/2021
convertDateToString	15'	26/10/2021	21/10/2021
parseDate	5'	26/10/2021	21/10/2021
isToDoListEmpty	2'	26/10/2021	21/10/2021
ToDoListManager methods			
startProgramm	10'	26/10/2021	21/10/2021
processTask	120'	26/10/2021	21/10/2021
*executeRemoveTask	15'	"	21/10/2021
*checkTask	35'	"	21/10/2021
*executemarkTaskAsDone	20'	"	21/10/2021
*executeTaskFromFile	10'	"	21/10/2021
FileHandler methods			
readFile	10'	26/10/2021	21/10/2021
saveToFile	10'	26/10/2021	21/10/2021
IO			
printAvailableActions	5'	27/10/2021	21/10/2021
readInput	5'	27/10/2021	21/10/2021
printMenu	5'	27/10/2021	21/10/2021
printInstructionTask	5'	27/10/2021	21/10/2021
printTask	5'	27/10/2021	21/10/2021
printErrorMsg	5'	27/10/2021	21/10/2021
printTaskStatus	5'	27/10/2021	21/10/2021
printGoodBye	5'	27/10/2021	21/10/2021
Validation			
isIdAvailable	5'	03/11/2021	03/11/2021
isTaskComplete	10'	03/11/2021	03/11/2021
isDateValid	5'	03/11/2021	03/11/2021
isEditedTaskValid	10'	03/11/2021	03/11/2021
isPathValid	2'	03/11/2021	03/11/2021

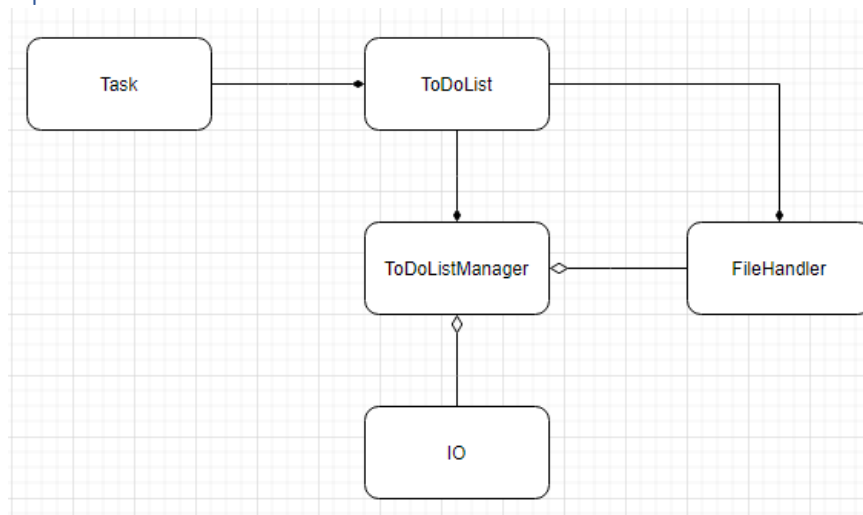
UML

Rough design of the classes

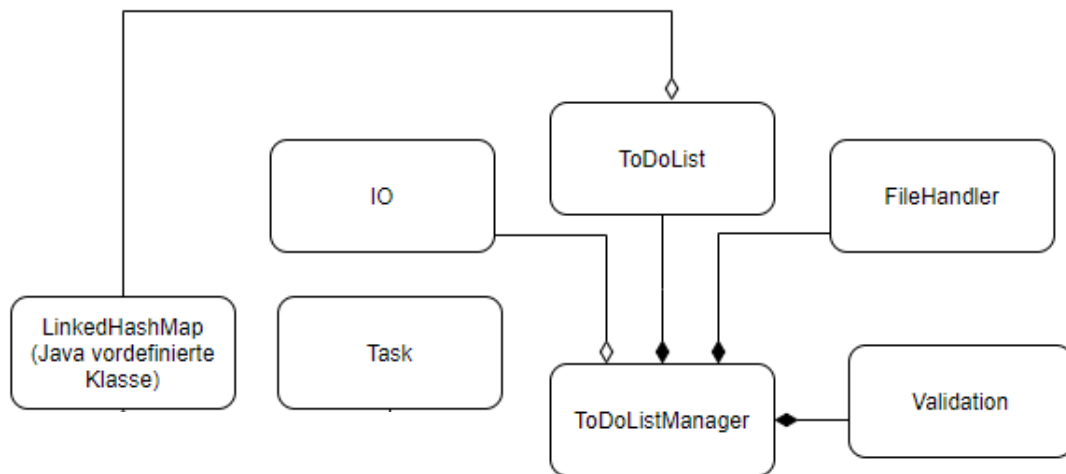
version 1.0



Updated version 1.1



Updated version 1.2



Use Cases

Starting the program:

Pre-Condition: The program is started

Description: IntelliJ has just started, and you start to compile the program.

Post-Condition: Menu with options of what you can do, will show. You don't have any tasks; your to-do list is empty. You can choose one of the showed options. Because of your empty to-do list, you can only add tasks or read tasks from file, all the other options are not allowed.

Add a task:

Pre-Condition: You must enter your task and follow the instruction.

Description: The menu with options was already displayed and you have selected to add a new task to your to-do list. At first, we get an instruction and example how do you add a task and after that you can write your task down.

Post-Condition: If the task was successfully added, you will get an output with a green colour and you will get the menu with options again otherwise you will get an error message because you did something wrong and didn't follow the instruction. Because of that you must write your task again or press 0 to break the process and go back to the menu.

Remove a task:

Pre-Condition: You must enter the id of the task, that you want to delete/remove.

Description: The menu with options was already displayed and you have selected to remove a task from your to-do list. At first, we get an instruction, after that you can enter an id.

Post-Condition: After entering the id of the task, that you want to remove/delete. The task with the following id will be removed, you will get a message that will tell you that otherwise if the id doesn't exist, the input will be demand again until you enter a valid id or enter 0.

Edit a task:

Pre-Condition: An instruction of how you edit a task will show, after that you must enter the task that you want to edit.

Description: The menu with options was already displayed and you have selected to edit/update a task from your to-do list. At first, we get an instruction, after that you can enter your task.

Post-Condition: You must enter the whole task gain, things that you don't want to be edited/updated you write a "-" "down. If you followed instruction, you would get a green output, who tells you that the task was successfully edited otherwise it will demand your input again until you entered the task right or entered 0 to break the process and go back to the menu.

Display all task:

Pre-Condition: The menu was displayed

Description: The menu with options was already displayed and you have selected to display all tasks.

Post-Condition: All tasks will show in boxes with his id, title, due date, status, and project name. All tasks that have the status done will show at the end.

Sort tasks by date:

Pre-Condition: The menu was displayed

Description: The menu with options was already displayed and you have selected to sort all tasks by date.

Post-Condition: You will get a message that the task was successfully sorted, and all the tasks will be displayed sorted by date.

Sort tasks by project:

Pre-Condition: The menu was displayed

Description: The menu with options was already displayed and you have selected to sort all tasks by project.

Post-Condition: You will get a message that the task was successfully sorted, and all the tasks will be displayed sorted by project.

Save tasks to file:

Pre-Condition: An instruction of how you save all your tasks will show, after that you must enter the path of where you want to save it.

Description: The menu with options was already displayed and you have selected to save all your tasks (to-do list) in a txt file (it must be a txt file). At first, we get an instruction, after that you can enter your path.

Post-Condition: After a valid path you will get a green output, that will tell you that the tasks were successfully saved, and it will show the path again otherwise an error message will show until you enter a valid path or 0.

Read from file:

Pre-Condition: An instruction of how you read from your file all your tasks will show, after that you must enter the path of your file.

Description: The menu with options was already displayed and you have selected to read all your tasks from your txt file. At first, we get an instruction, after that you can enter your path.

Post-Condition: If the path is valid all tasks will be read and added to the to-do list otherwise an error message will show until you enter a valid path or 0.

Exit:

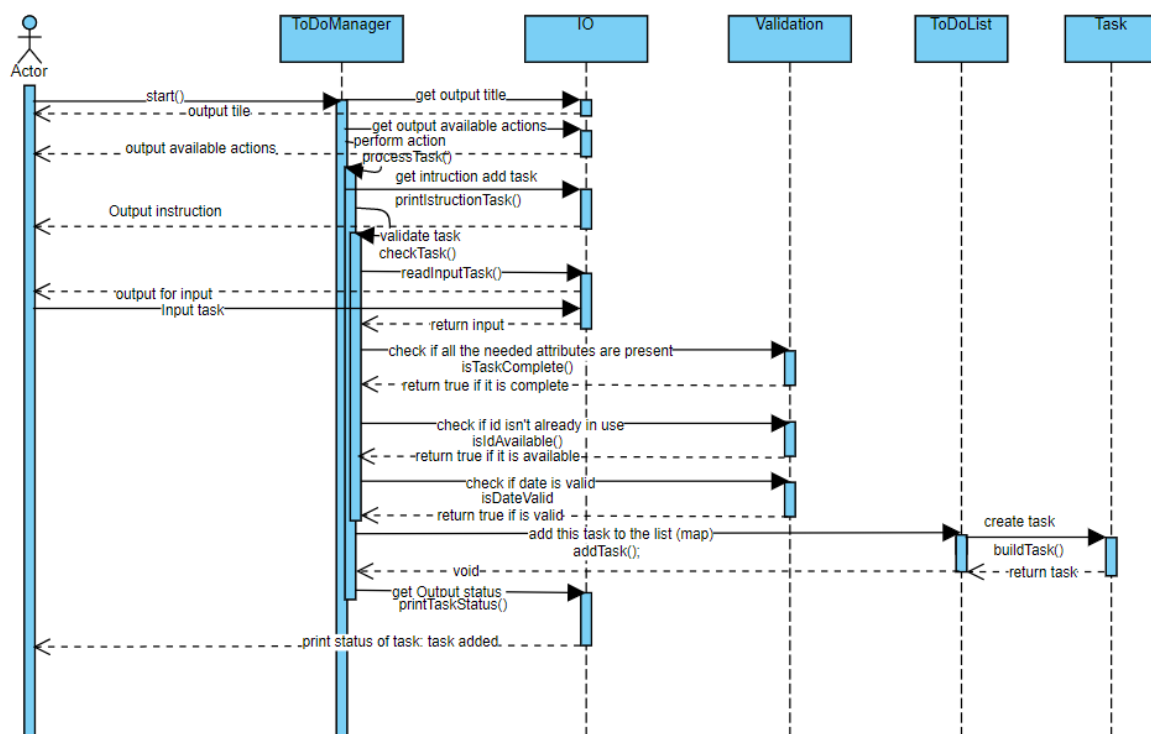
Pre-Condition: The menu was displayed

Description: The menu with options was already displayed and you have selected to exit. This is the only way to end the program.

Post-Condition: Goodbye message will show and the program is finished

Sequence Diagrams

Add a Task, following the instructions of how to add a task correctly:



→the application keeps running, until the user decides to end the program.

Descriptions -> aggregation etc.

Shortcuts github and intellij


Where I did what he what

Implementations

Delegation

ToDoListManager:

```
case ADD_TASK -> {
    io.printInstructionsTask(instructionType: "add");
    task = checkNewTask();
    if (!task.equals(GET_BACK_TO_MENU)) {
        toDoList.addTask(task);
        io.printTaskStatus(msgType: "task", information: "added");
    }
}
```



And more...

Composition

Validation Class:

```
public Validation(Map<Integer, Task> tasks) { this.tasks = tasks; }
```

FileHandler Class:

```
public FileHandler(ToDoList toDoList) { this.toDoList = toDoList; }
```

And more...

Aggregation

ToDoListManager Class:

```
private final ToDoList toDoList = new ToDoList();
private final Validation validation = new Validation(toDoList.getTasks());
private final FileHandler fileHandler = new FileHandler(toDoList);
private final IO io = new IO();
```

FileHandler Class:

```
private final ToDoList toDoList;
```

IO Class:

```
private final Scanner scan = new Scanner(System.in);
```

And more...

Encapsulation

ToDoListManager:

```
private static final int ADD_TASK = 1;
private static final int MARK_AS_DONE = 2;
private static final int REMOVE_TASK = 3;
private static final int EDIT_TASK = 4;
private static final int DISPLAY_ALL_TASKS = 5;
private static final int SORT_TASKS_BY_DATE = 6;
private static final int SORT_TASKS_PROJECT = 7;
private static final int SAVE_TASKS_TO_FILE = 8;
private static final int READ_FROM_FILE = 9;
private static final int EXIT = 10;
private static final String GET_BACK_TO_MENU = "0";

public static boolean applicationRunning = true;
private final ToDoList toDoList = new ToDoList();
private final Validation validation = new Validation(toDoList.getTasks());
private final FileHandler fileHandler = new FileHandler(toDoList);
private final IO io = new IO();
private boolean repeat;
private int id;
```

And more...

Class Diagram

