TODO LIST PROGRAM

Yu Duo Zhang

Vanier College
[420-SF2-RE] DATA STRUCTURES OOP

Table of Content

Project Description:

Scenario:

The project simulates a user-based Todo List management system. Users can log in, create tasks, view existing tasks, and mark them as completed. The system supports two user types (students or guests) and varies the level of access. This helps users organize their daily activities efficiently and prioritizes tasks based on deadlines or urgency.

Expected Output:

Users will interact with the system via the console. Based on their role, users can view or manage their task lists. Tasks can be saved and loaded across sessions. Tasks will be sorted and displayed according to user-selected preferences (e.g., by priority). The system will reflect changes (e.g., task completion) in real-time.

Functionalities:

The system provides different services based on the user type:

Student User

1. Add a new task
2. Mark a task as completed
3. Delete a task
4. Sort tasks by deadline
5. Sort tasks by priority
6. View all tasks
7. Save and load task records
8. Undo and Redo the last action

Guest User

1. View all Tasks
2. Load task records
3. Sort tasks by deadline
4. Sort tasks by priority

Implementation Details:

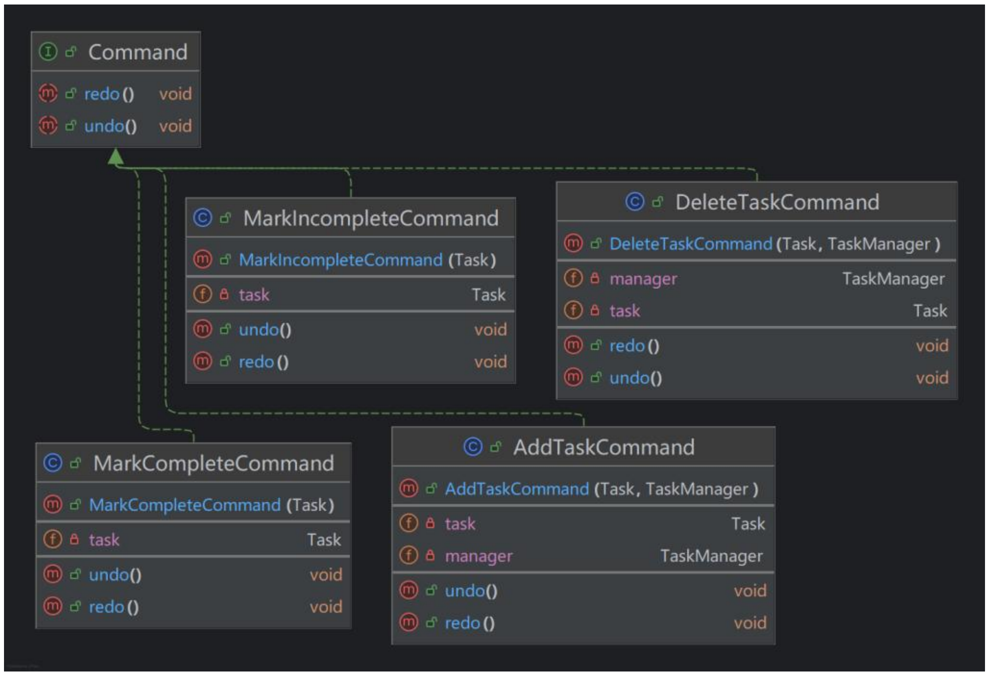
1. addTask(Task task): Adds a new task to the Task list. Can be canceled by undo().
2. markComplete(int index): Set a certain task in the list as complete status to true. Can be canceled by undo().
3. deleteTask(int index): Removes a task from the list. Can be restored by undo().
4. sortByCreationDate: Sort the list by using compareTo().

5. sortByDeadline(): Sort the list by using DeadlineComparator.
6. sortByPriority(): Sorts the list by using PriorityComparator, which puts urgent tasks before the rest.
7. saveTasks(String filePath): Writes all current task records to a csv file.
8. loadTasks(String filePath): Reads all tasks from a csv file and constructs them into a task list.
9. displayAllTasks(): display on the console all current tasks
10. searchTasks(String Keyword): Using Stream and Lambda Expression to search for tasks with a given keyword.
11. undo(): Pops the most recent command from the undo stack and pushes it to the redo stack.
12. redo(): Pops the most recent command from the redo stack and pushes it back to the undo stack.

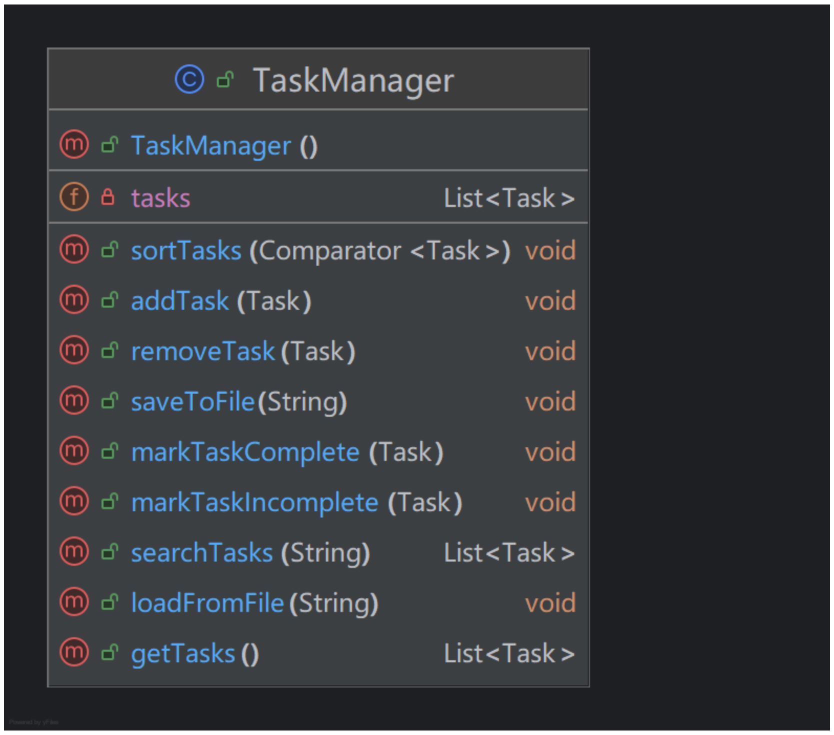
Program Features:

2.1: UML Diagram

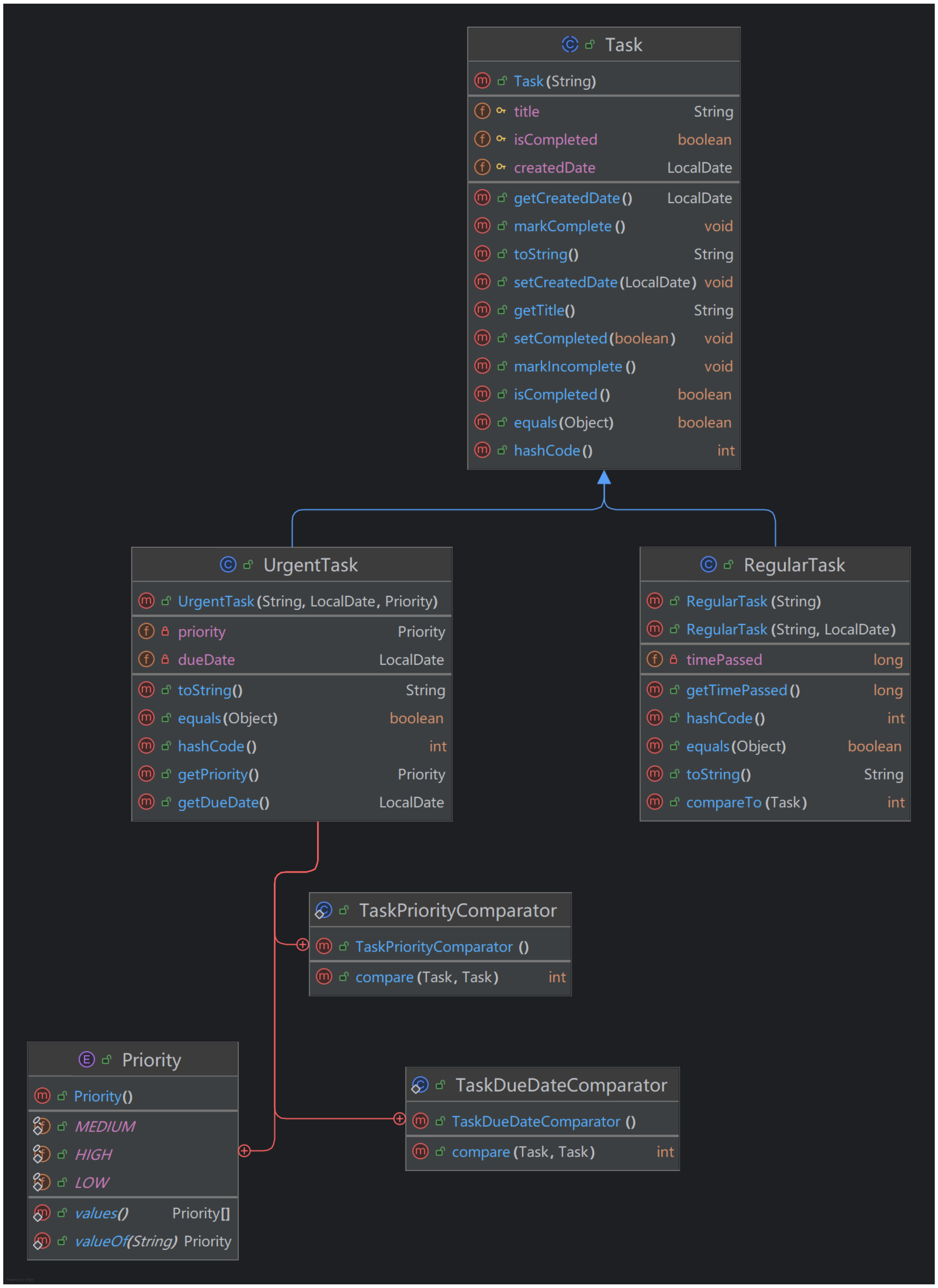
Command:



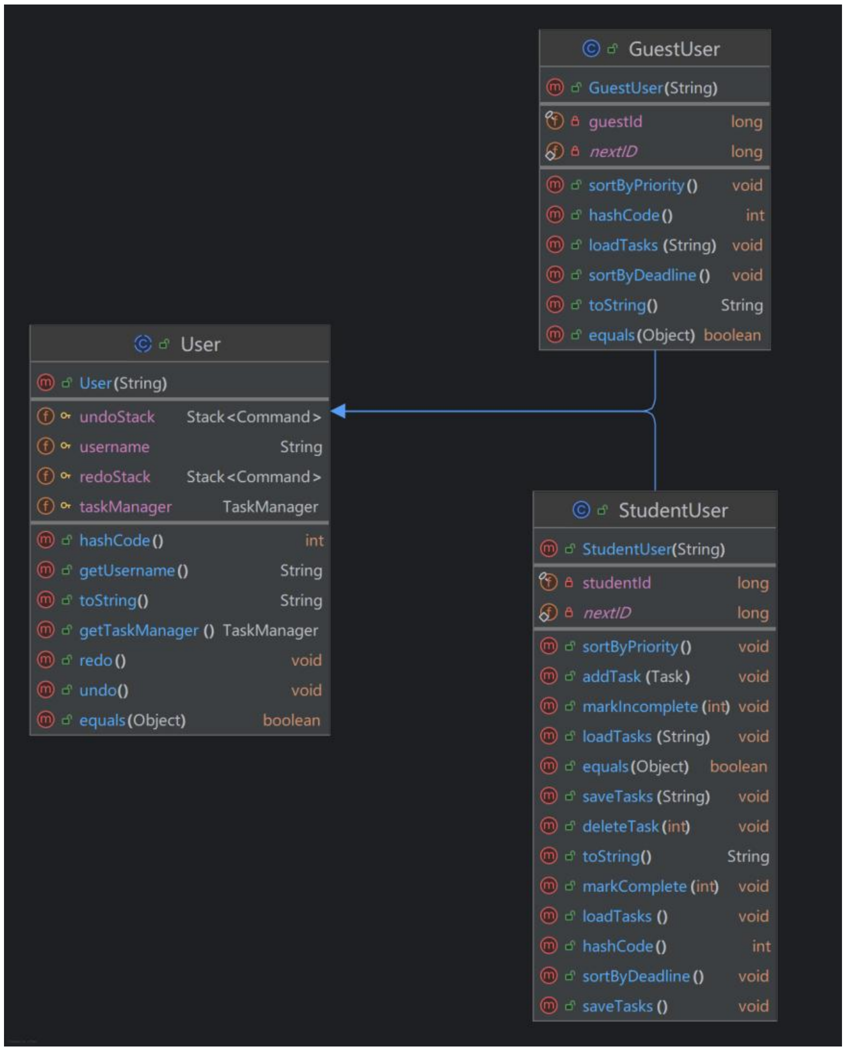
Task Manager:


TaskManager
TaskManager ()
tasks List<Task>
sortTasks (Comparator <Task>) void
addTask (Task) void
removeTask (Task) void
saveToFile(String) void
markTaskComplete (Task) void
markTaskIncomplete (Task) void
searchTasks (String) List<Task>
loadFromFile (String) void
getTasks () List<Task>

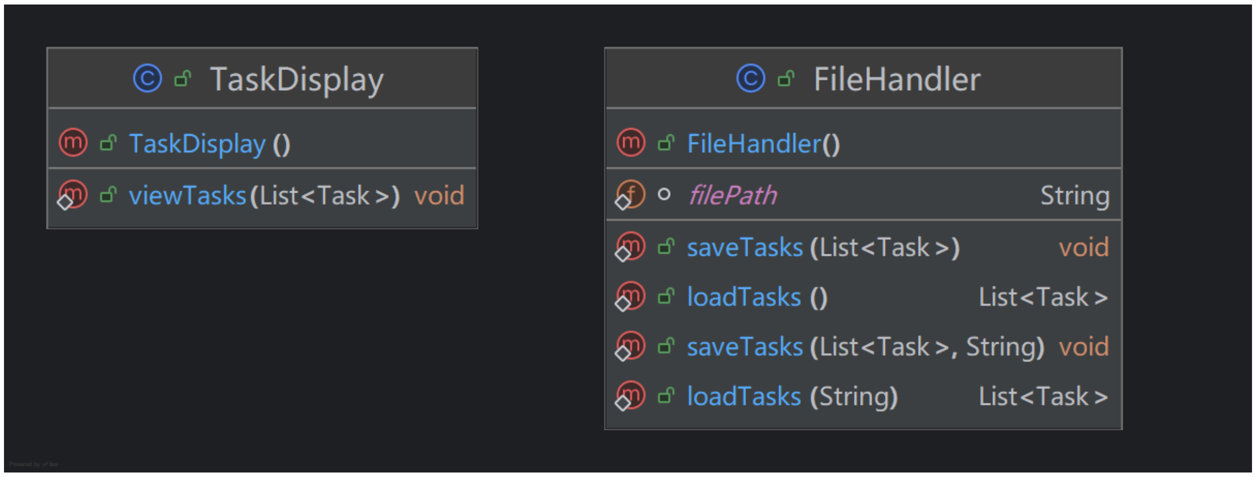
Tasks:


Task
Task(String)
title String
isCompleted boolean
createdDate LocalDate
getCreatedDate() LocalDate
markComplete() void
toString() String
setCreatedDate(LocalDate) void
getTitle() String
setCompleted(boolean) void
markIncomplete() void
isCompleted() boolean
equals(Object) boolean
hashCode() int
UrgentTask
UrgentTask(String, LocalDate, Priority)
priority Priority
dueDate LocalDate
toString() String
equals(Object) boolean
hashCode() int
getPriority() Priority
getDueDate() LocalDate
RegularTask
RegularTask(String)
RegularTask(String, LocalDate)
timePassed long
getTimePassed() long
hashCode() int
equals(Object) boolean
toString() String
compareTo(Task) int
TaskPriorityComparator
TaskPriorityComparator()
compare(Task, Task) int
Priority
Priority()
MEDIUM
HIGH
LOW
values() Priority[]
valueOf(String) Priority
TaskDueDateComparator
TaskDueDateComparator()
compare(Task, Task) int

Users:



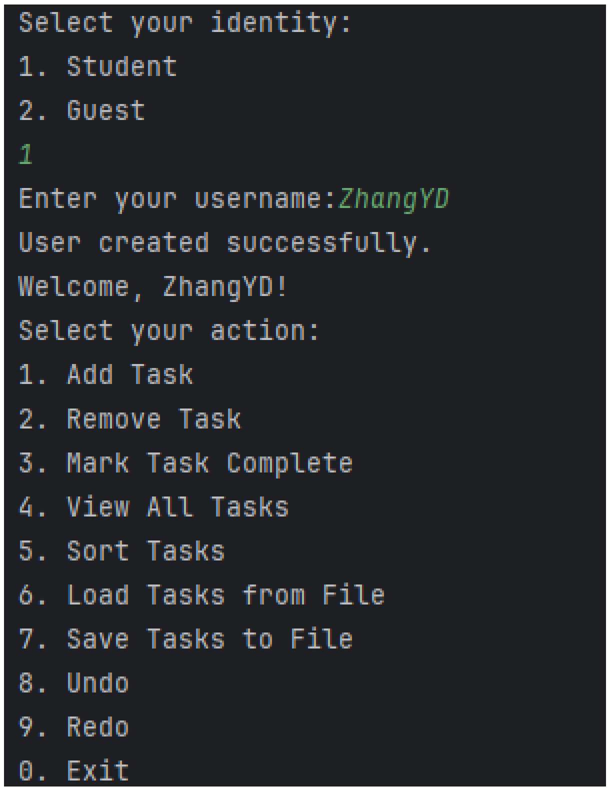
Utilities:


TaskDisplay
TaskDisplay()
viewTasks(List<Task>) void
FileHandler
FileHandler()
filePath String
saveTasks(List<Task>) void
loadTasks() List<Task>
saveTasks(List<Task>, String) void
loadTasks(String) List<Task>

2.2: Simple Demo

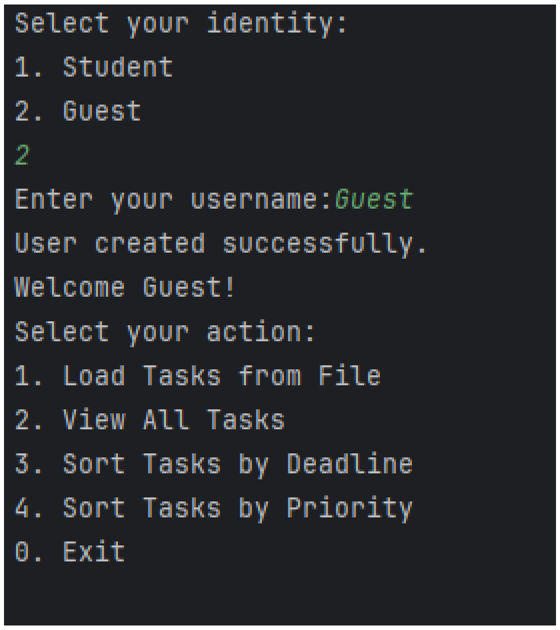
As the Main program starts to run, the program will ask for your identity, notice that if you're a student, you will have more command options than a guest.

Student Menu:



```
Select your identity:
1. Student
2. Guest
1
Enter your username:ZhangYD
User created successfully.
Welcome, ZhangYD!
Select your action:
1. Add Task
2. Remove Task
3. Mark Task Complete
4. View All Tasks
5. Sort Tasks
6. Load Tasks from File
7. Save Tasks to File
8. Undo
9. Redo
0. Exit
```

Guest Menu:

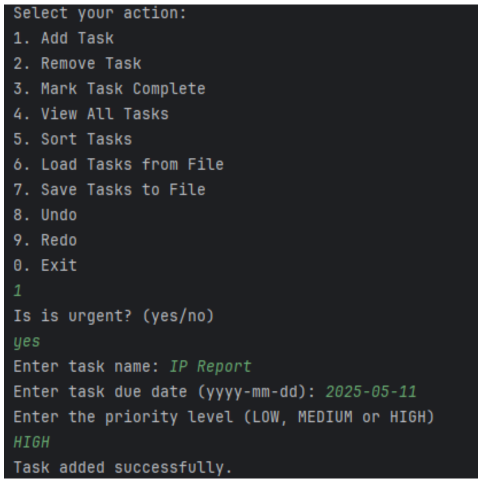


```
Select your identity:
1. Student
2. Guest
2
Enter your username:Guest
User created successfully.
Welcome Guest!
Select your action:
1. Load Tasks from File
2. View All Tasks
3. Sort Tasks by Deadline
4. Sort Tasks by Priority
0. Exit
```

If you choose to use the system as a student, you'll be able to add and remove tasks, as well as you can mark them into complete status. You can search for specific tasks by using the "View All Tasks" function, as well as you can look up for all the tasks. Once you have a big list of tasks, you can use the sorting to sort them by deadline or priority, both will put the regular tasks to the bottom of the list.

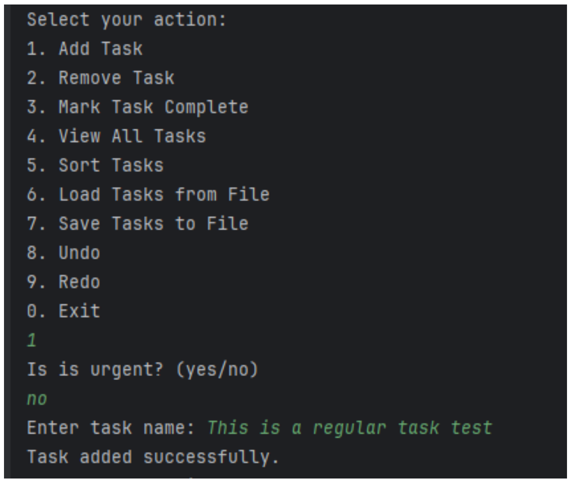
In case that you need to switch device or show the list to another person, you can export your task list to a comma separate value (.csv) file, then load the same list somewhere else using the loading function.

In case that you've created / removed / marked complete the wrong task, you can always use the undo function to cancel your last command, as well as you can redo.



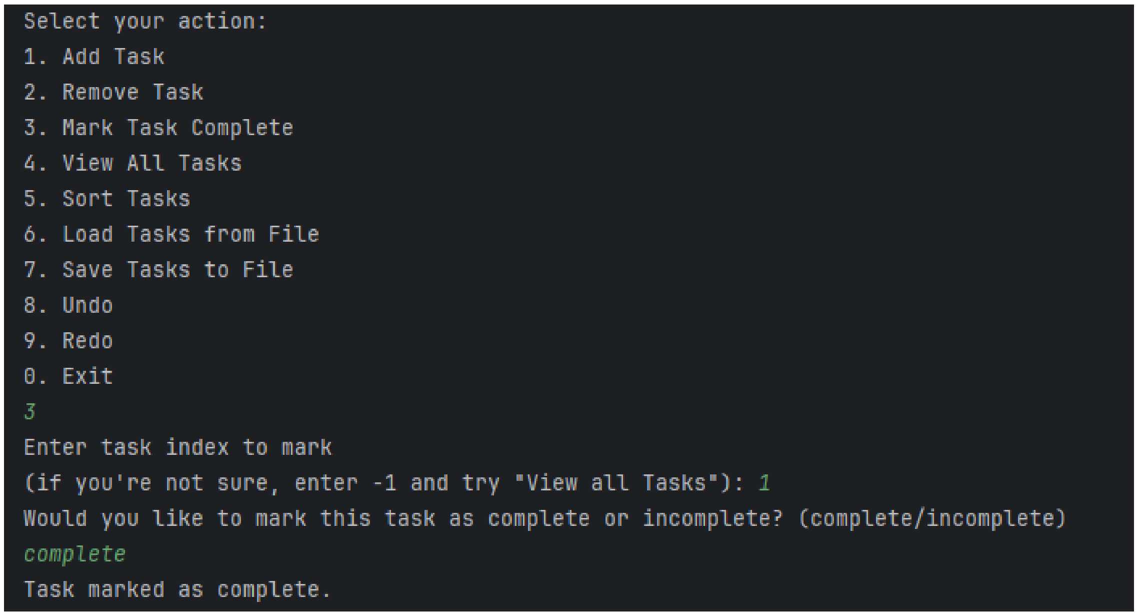
```
Select your action:
1. Add Task
2. Remove Task
3. Mark Task Complete
4. View All Tasks
5. Sort Tasks
6. Load Tasks from File
7. Save Tasks to File
8. Undo
9. Redo
0. Exit
1
Is is urgent? (yes/no)
yes
Enter task name: IP Report
Enter task due date (yyyy-mm-dd): 2025-05-11
Enter the priority level (LOW, MEDIUM or HIGH)
HIGH
Task added successfully.
```

Add an urgent task



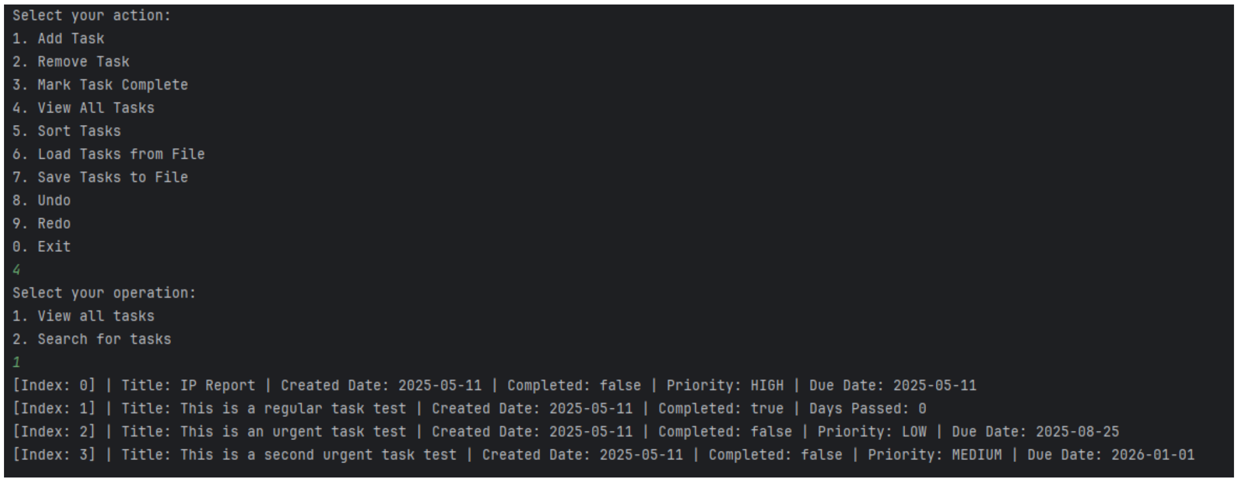
```
Select your action:
1. Add Task
2. Remove Task
3. Mark Task Complete
4. View All Tasks
5. Sort Tasks
6. Load Tasks from File
7. Save Tasks to File
8. Undo
9. Redo
0. Exit
1
Is is urgent? (yes/no)
no
Enter task name: This is a regular task test
Task added successfully.
```

Add a regular task



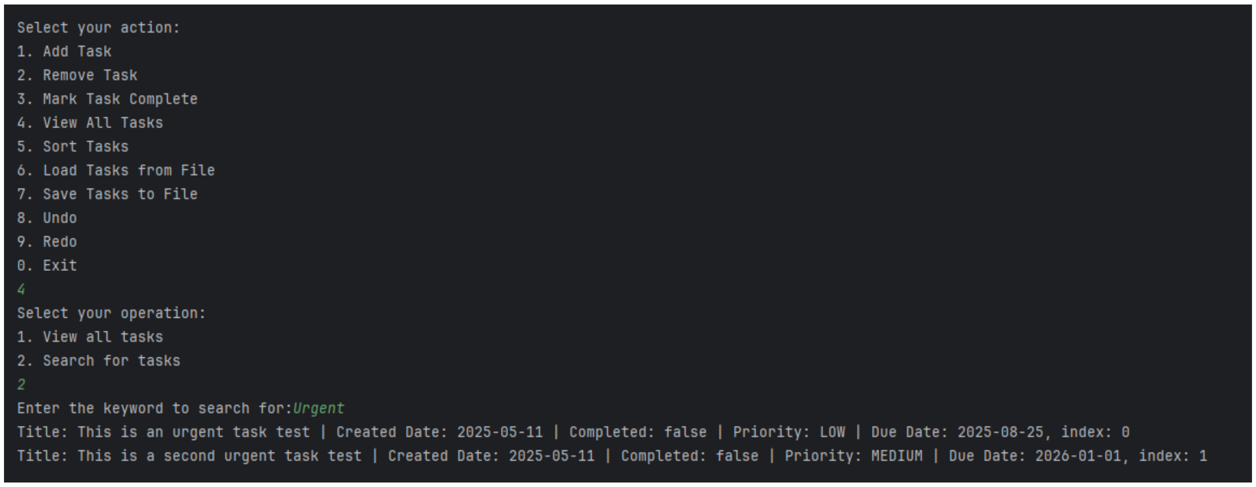
```
Select your action:
1. Add Task
2. Remove Task
3. Mark Task Complete
4. View All Tasks
5. Sort Tasks
6. Load Tasks from File
7. Save Tasks to File
8. Undo
9. Redo
0. Exit
3
Enter task index to mark
(if you're not sure, enter -1 and try "View all Tasks"): 1
Would you like to mark this task as complete or incomplete? (complete/incomplete)
complete
Task marked as complete.
```

Mark a task as complete



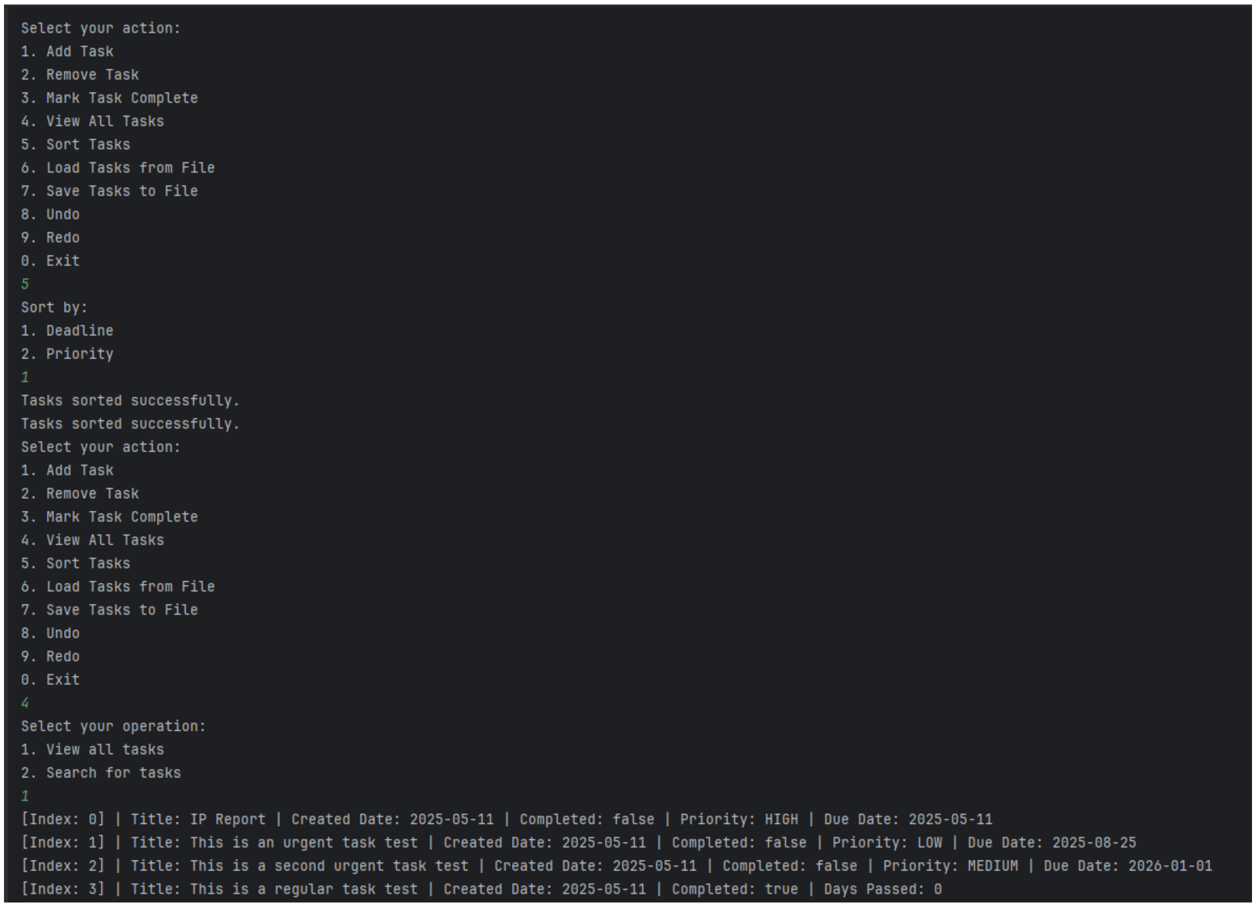
```
Select your action:
1. Add Task
2. Remove Task
3. Mark Task Complete
4. View All Tasks
5. Sort Tasks
6. Load Tasks from File
7. Save Tasks to File
8. Undo
9. Redo
0. Exit
4
Select your operation:
1. View all tasks
2. Search for tasks
1
[Index: 0] | Title: IP Report | Created Date: 2025-05-11 | Completed: false | Priority: HIGH | Due Date: 2025-05-11
[Index: 1] | Title: This is a regular task test | Created Date: 2025-05-11 | Completed: true | Days Passed: 0
[Index: 2] | Title: This is an urgent task test | Created Date: 2025-05-11 | Completed: false | Priority: LOW | Due Date: 2025-08-25
[Index: 3] | Title: This is a second urgent task test | Created Date: 2025-05-11 | Completed: false | Priority: MEDIUM | Due Date: 2026-01-01
```

View all tasks



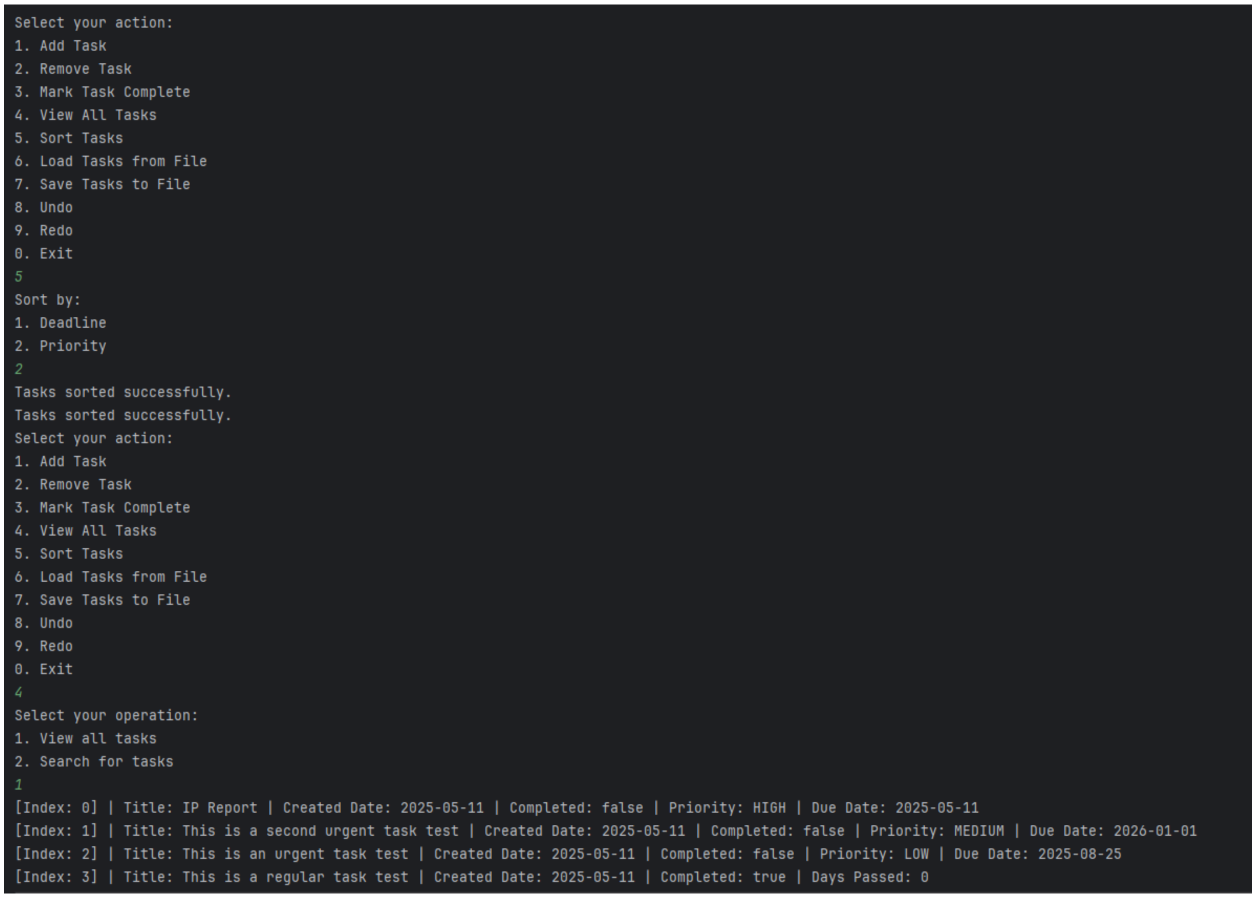
```
Select your action:
1. Add Task
2. Remove Task
3. Mark Task Complete
4. View All Tasks
5. Sort Tasks
6. Load Tasks from File
7. Save Tasks to File
8. Undo
9. Redo
0. Exit
4
Select your operation:
1. View all tasks
2. Search for tasks
2
Enter the keyword to search for:Urgent
Title: This is an urgent task test | Created Date: 2025-05-11 | Completed: false | Priority: LOW | Due Date: 2025-08-25, index: 0
Title: This is a second urgent task test | Created Date: 2025-05-11 | Completed: false | Priority: MEDIUM | Due Date: 2026-01-01, index: 1
```

Search for specific tasks



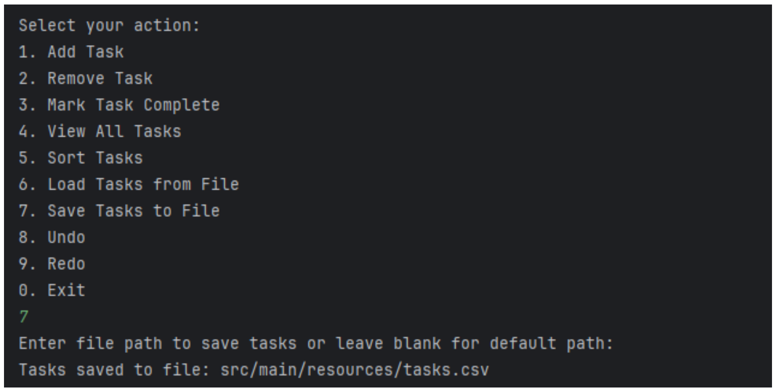
```
Select your action:
1. Add Task
2. Remove Task
3. Mark Task Complete
4. View All Tasks
5. Sort Tasks
6. Load Tasks from File
7. Save Tasks to File
8. Undo
9. Redo
0. Exit
5
Sort by:
1. Deadline
2. Priority
1
Tasks sorted successfully.
Tasks sorted successfully.
Select your action:
1. Add Task
2. Remove Task
3. Mark Task Complete
4. View All Tasks
5. Sort Tasks
6. Load Tasks from File
7. Save Tasks to File
8. Undo
9. Redo
0. Exit
4
Select your operation:
1. View all tasks
2. Search for tasks
1
[Index: 0] | Title: IP Report | Created Date: 2025-05-11 | Completed: false | Priority: HIGH | Due Date: 2025-05-11
[Index: 1] | Title: This is an urgent task test | Created Date: 2025-05-11 | Completed: false | Priority: LOW | Due Date: 2025-08-25
[Index: 2] | Title: This is a second urgent task test | Created Date: 2025-05-11 | Completed: false | Priority: MEDIUM | Due Date: 2026-01-01
[Index: 3] | Title: This is a regular task test | Created Date: 2025-05-11 | Completed: true | Days Passed: 0
```

Sorting Tasks by Due Date

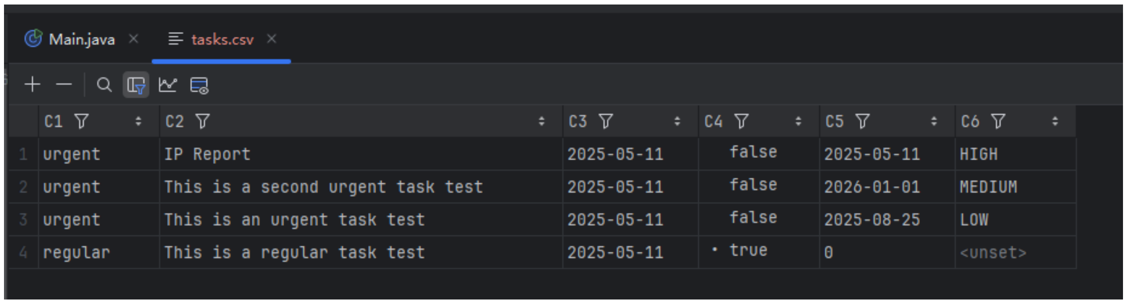


```
Select your action:
1. Add Task
2. Remove Task
3. Mark Task Complete
4. View All Tasks
5. Sort Tasks
6. Load Tasks from File
7. Save Tasks to File
8. Undo
9. Redo
0. Exit
5
Sort by:
1. Deadline
2. Priority
2
Tasks sorted successfully.
Tasks sorted successfully.
Select your action:
1. Add Task
2. Remove Task
3. Mark Task Complete
4. View All Tasks
5. Sort Tasks
6. Load Tasks from File
7. Save Tasks to File
8. Undo
9. Redo
0. Exit
4
Select your operation:
1. View all tasks
2. Search for tasks
1
[Index: 0] | Title: IP Report | Created Date: 2025-05-11 | Completed: false | Priority: HIGH | Due Date: 2025-05-11
[Index: 1] | Title: This is a second urgent task test | Created Date: 2025-05-11 | Completed: false | Priority: MEDIUM | Due Date: 2026-01-01
[Index: 2] | Title: This is an urgent task test | Created Date: 2025-05-11 | Completed: false | Priority: LOW | Due Date: 2025-08-25
[Index: 3] | Title: This is a regular task test | Created Date: 2025-05-11 | Completed: true | Days Passed: 0
```

Sorting Tasks by Priority



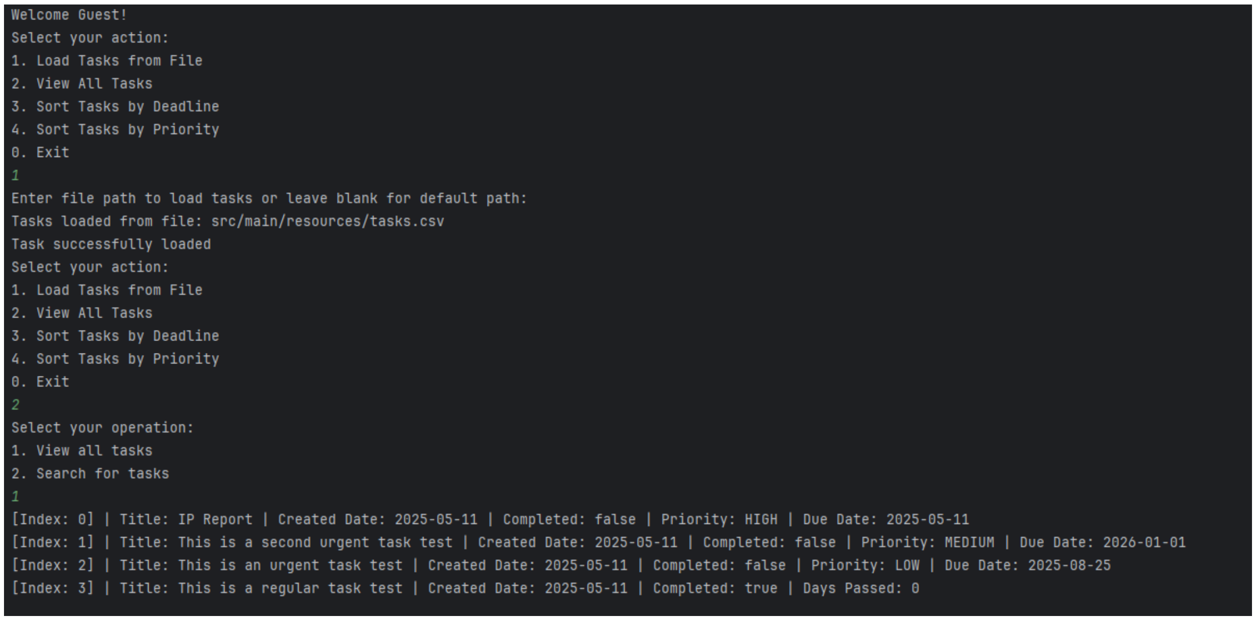
```
Select your action:
1. Add Task
2. Remove Task
3. Mark Task Complete
4. View All Tasks
5. Sort Tasks
6. Load Tasks from File
7. Save Tasks to File
8. Undo
9. Redo
0. Exit
7
Enter file path to save tasks or leave blank for default path:
Tasks saved to file: src/main/resources/tasks.csv
```



Main.java tasks.csv

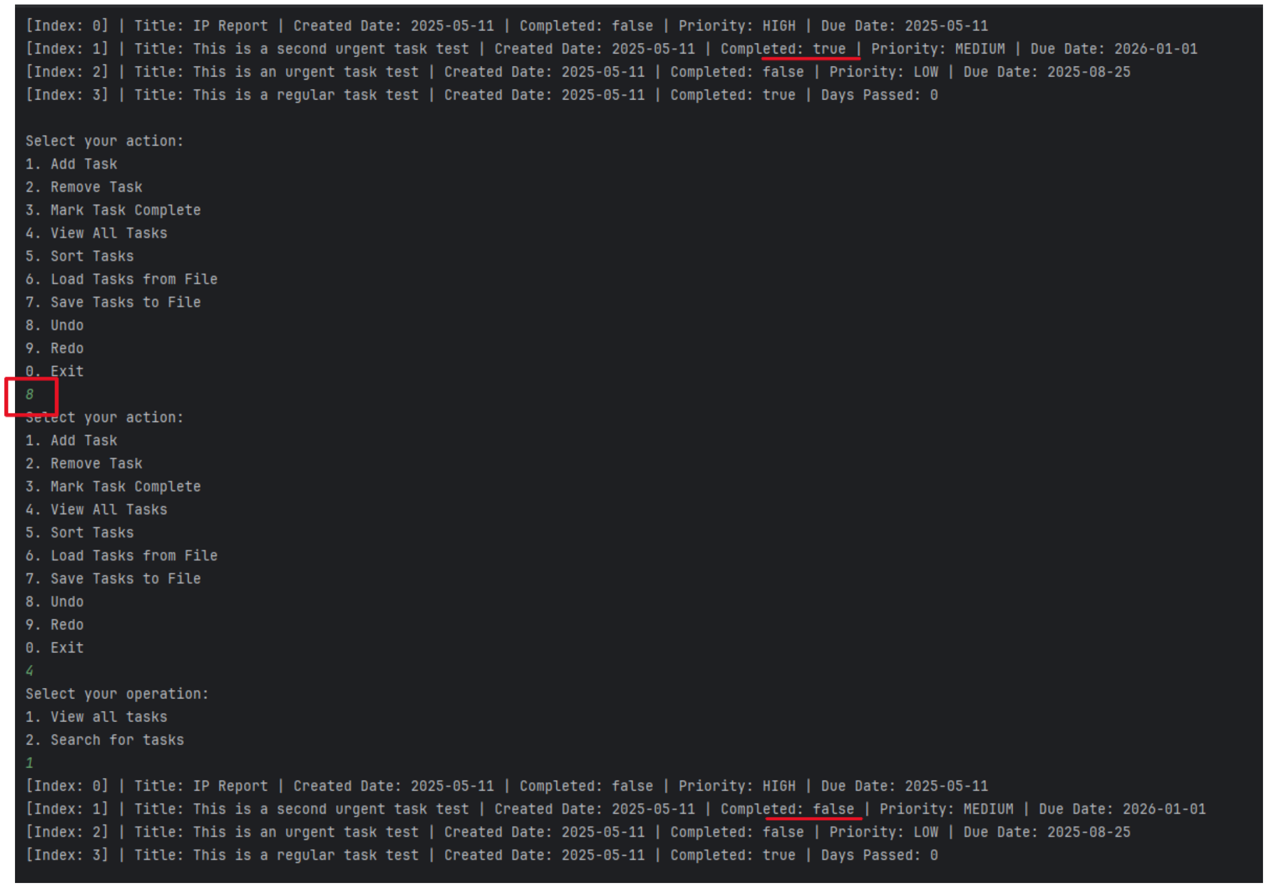
	C1	C2	C3	C4	C5	C6
1	urgent	IP Report	2025-05-11	false	2025-05-11	HIGH
2	urgent	This is a second urgent task test	2025-05-11	false	2026-01-01	MEDIUM
3	urgent	This is an urgent task test	2025-05-11	false	2025-08-25	LOW
4	regular	This is a regular task test	2025-05-11	true	0	<unset>

Saving Tasks



```
Welcome Guest!
Select your action:
1. Load Tasks from File
2. View All Tasks
3. Sort Tasks by Deadline
4. Sort Tasks by Priority
0. Exit
1
Enter file path to load tasks or leave blank for default path:
Tasks loaded from file: src/main/resources/tasks.csv
Task successfully loaded
Select your action:
1. Load Tasks from File
2. View All Tasks
3. Sort Tasks by Deadline
4. Sort Tasks by Priority
0. Exit
2
Select your operation:
1. View all tasks
2. Search for tasks
1
[Index: 0] | Title: IP Report | Created Date: 2025-05-11 | Completed: false | Priority: HIGH | Due Date: 2025-05-11
[Index: 1] | Title: This is a second urgent task test | Created Date: 2025-05-11 | Completed: false | Priority: MEDIUM | Due Date: 2026-01-01
[Index: 2] | Title: This is an urgent task test | Created Date: 2025-05-11 | Completed: false | Priority: LOW | Due Date: 2025-08-25
[Index: 3] | Title: This is a regular task test | Created Date: 2025-05-11 | Completed: true | Days Passed: 0
```

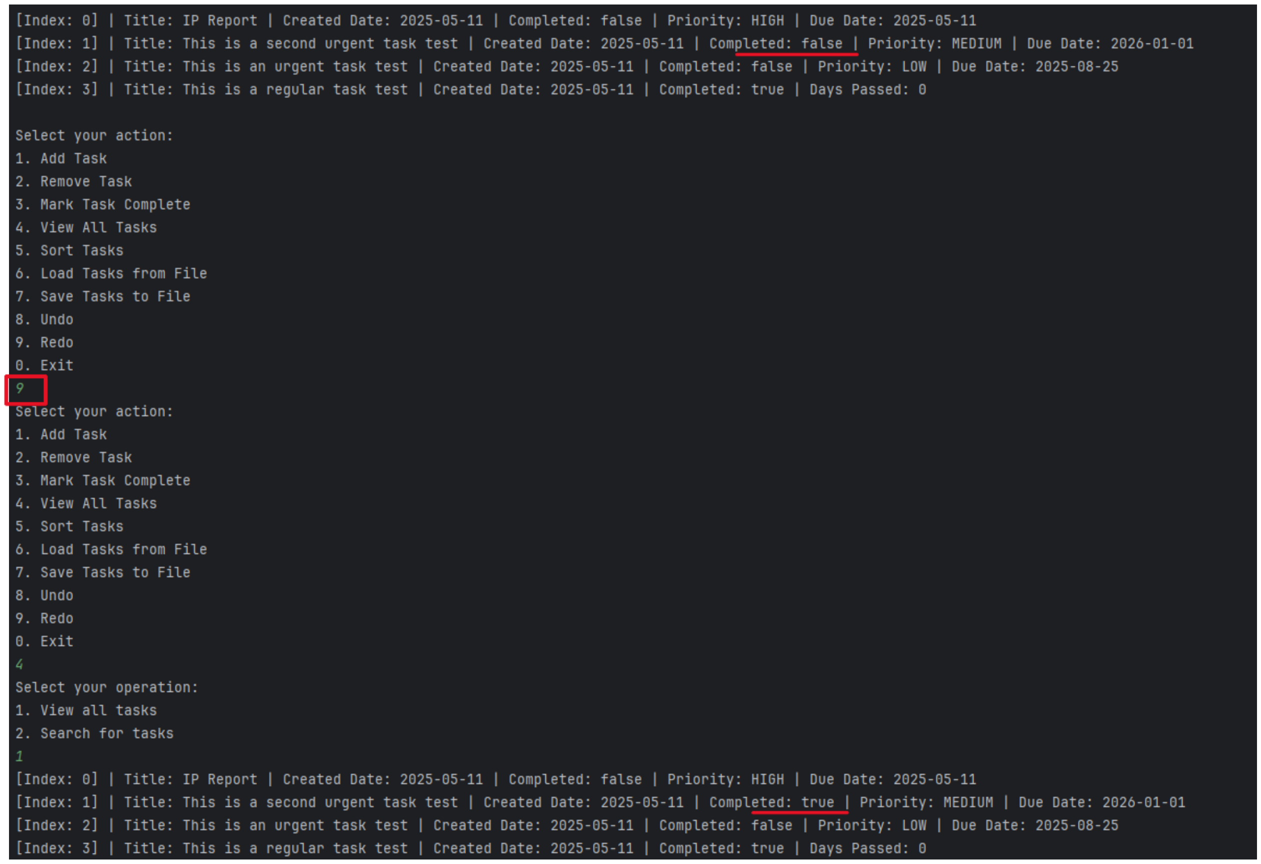
Loading Tasks



```
[Index: 0] | Title: IP Report | Created Date: 2025-05-11 | Completed: false | Priority: HIGH | Due Date: 2025-05-11
[Index: 1] | Title: This is a second urgent task test | Created Date: 2025-05-11 | Completed: true | Priority: MEDIUM | Due Date: 2026-01-01
[Index: 2] | Title: This is an urgent task test | Created Date: 2025-05-11 | Completed: false | Priority: LOW | Due Date: 2025-08-25
[Index: 3] | Title: This is a regular task test | Created Date: 2025-05-11 | Completed: true | Days Passed: 0

Select your action:
1. Add Task
2. Remove Task
3. Mark Task Complete
4. View All Tasks
5. Sort Tasks
6. Load Tasks from File
7. Save Tasks to File
8. Undo
9. Redo
0. Exit
8
Select your action:
1. Add Task
2. Remove Task
3. Mark Task Complete
4. View All Tasks
5. Sort Tasks
6. Load Tasks from File
7. Save Tasks to File
8. Undo
9. Redo
0. Exit
4
Select your operation:
1. View all tasks
2. Search for tasks
1
[Index: 0] | Title: IP Report | Created Date: 2025-05-11 | Completed: false | Priority: HIGH | Due Date: 2025-05-11
[Index: 1] | Title: This is a second urgent task test | Created Date: 2025-05-11 | Completed: false | Priority: MEDIUM | Due Date: 2026-01-01
[Index: 2] | Title: This is an urgent task test | Created Date: 2025-05-11 | Completed: false | Priority: LOW | Due Date: 2025-08-25
[Index: 3] | Title: This is a regular task test | Created Date: 2025-05-11 | Completed: true | Days Passed: 0
```

Undo a Mark as Complete command



```
[Index: 0] | Title: IP Report | Created Date: 2025-05-11 | Completed: false | Priority: HIGH | Due Date: 2025-05-11
[Index: 1] | Title: This is a second urgent task test | Created Date: 2025-05-11 | Completed: false | Priority: MEDIUM | Due Date: 2026-01-01
[Index: 2] | Title: This is an urgent task test | Created Date: 2025-05-11 | Completed: false | Priority: LOW | Due Date: 2025-08-25
[Index: 3] | Title: This is a regular task test | Created Date: 2025-05-11 | Completed: true | Days Passed: 0

Select your action:
1. Add Task
2. Remove Task
3. Mark Task Complete
4. View All Tasks
5. Sort Tasks
6. Load Tasks from File
7. Save Tasks to File
8. Undo
9. Redo
0. Exit
9
Select your action:
1. Add Task
2. Remove Task
3. Mark Task Complete
4. View All Tasks
5. Sort Tasks
6. Load Tasks from File
7. Save Tasks to File
8. Undo
9. Redo
0. Exit
4
Select your operation:
1. View all tasks
2. Search for tasks
1
[Index: 0] | Title: IP Report | Created Date: 2025-05-11 | Completed: false | Priority: HIGH | Due Date: 2025-05-11
[Index: 1] | Title: This is a second urgent task test | Created Date: 2025-05-11 | Completed: true | Priority: MEDIUM | Due Date: 2026-01-01
[Index: 2] | Title: This is an urgent task test | Created Date: 2025-05-11 | Completed: false | Priority: LOW | Due Date: 2025-08-25
[Index: 3] | Title: This is a regular task test | Created Date: 2025-05-11 | Completed: true | Days Passed: 0
```

Redo the last undo command

2.3: Hierarchies

I made up two hierarchies: one with the users, and one with the tasks.

The separation between student and guest made a difference between the two types of users. The biggest difference between them is the amount of method that they have, the student user has much more choices on functions than a guest.

On the side of tasks, we have the difference between a regular task and urgent task. Their differences are mostly on their data field. They both have a title and creation date, but I added a "how many days have passed since user created this task" data in the regular task class, and I added a due date and priority to the urgent task class.

In this case, we have two hierarchies with two layers each, notice that both has an abstract class as parent.

2.4: Interface

I implemented one interface for the commands. Since add, remove and mark as complete are commands that has undo and redo, then they are commands that are "undoable" and "redoable", then I guess I can make an interface for them to get them having two mandatory methods (undo and redo).

The Command interface in this project is having the function of "undoable" and "redoable", and by making the three commands implement the interface, I can directly create a stack of command for the user and push all three types in the undo stack and redo stack.

2.5: Polymorphism

The polymorphism is used on the toString method for both tasks. Because of the difference between the data field between them, their display must have some difference too, so an override for toString on both was necessary.

Great thing is we can use "super" for the title and creation date part…

2.6: Text IO

The text IO made up the whole system of file handling about saving and loading.

For saving, I used I FileWritter to save all data from the tasks into the csv format in the list placed in the task manager corresponding to each user. Similarly, I used a Scanner to scan the data in the same order, so they can temporarily be saved in the task manager again.

2.7: Comparable

Since the regular task class has only one extra field: the number of days passed, I decided to implement Comparable to this class. Since the number of days is set as a "long" variable, it is simpler to directly use the compare method provided by the Long class.

One thing important to know is that the method has to make sure both tasks we are comparing are regular tasks, which is why it will throw an ClassCastException if it's not the case. The rest will be handled by the task manager.

2.8: Comparator

On the side of urgent task, we have a priority level and a due date field. In this case, we can make two comparators, one for each of the two data. This time, we can handle both urgent and regular, because regular tasks doesn't have any of them, so they are less important than urgent tasks anyway. In this case, if an urgent task meets a regular task, then the previous is automatically more important than the second. When both are urgent, then we can let de the compareTo method handle them.

2.9: Unit Test

The one class which needs the most of testing is the TaskManager class, since it is where all the functions are being performed. For this project, I made one regular test per method to test their performance, then I tried several exception cases in the main program to simulate a real user experience. TaskManager tested all the functions about tasks needed to be implemented, except TextIO, tested by the test made for FileHandler. Undo and Redo are tested in the User test, since for most of the time, user will be the one doing those commands. Finally, TaskTest had a simple test about mark complete and mark incomplete, which will be useful for the mark complete / incomplete command.

Challenges

Honestly, starting a new project from zero with no instructions seems already be the hard part for me. I've never been a creative person, so I didn't have much idea of how to build the structure of this system at the very beginning. While writing the deliverable 1, I wanted to use most of what we learned in class, including stream and lambda expressions. At first, I wanted to write the method for task searching by using stream, but when the time of coding arrived, I totally forgot about I wanted to use that, and I ended up by doing a usual for-loop.

Something similar happened to the comparing regular tasks. I did write a compareTo method for the regular tasks, but I totally forgot to implement it into the main program or to the comparators that could have used it when it meets two regular tasks.

I guess I just don't have a good memory to realise all my ideas from the beginning…?

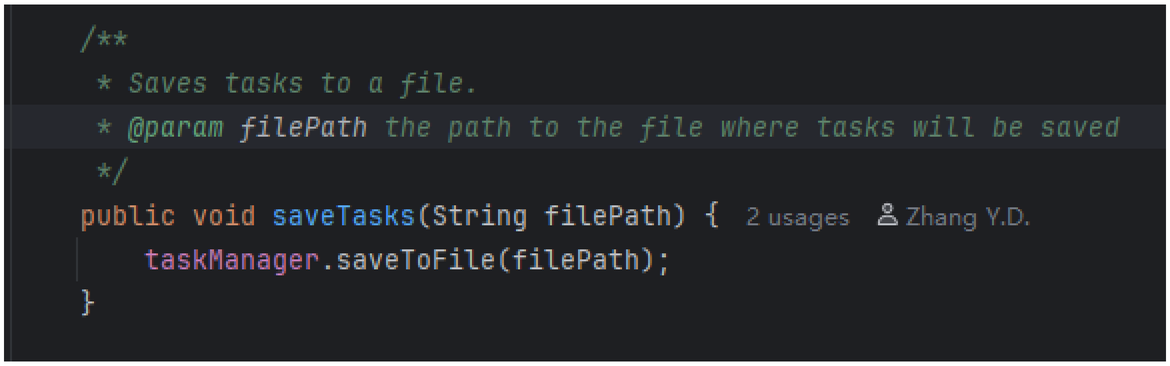
On the user side, it would be better if I tried to write a whole log in system instead of creating a new user each time we restart the program, but I only had half of the idea about "how this should be realised", so I ended up by abandoning.

During development, I also realized that if the folder resources didn't exist, java can't directly create the folder for the user. This issue ended up being fixed by making some research on Google…

Learning Outcomes

In general, this project gave me a better understanding of the topics we mentioned in class this semester, including the importance of exception handling, hierarchies, etc.

As a first try of starting everything from zero, my logic of data structure seems far from what a real project should look like, but after the realisation, I should be able to avoid some completely unnecessary methods such as those method doing nothing more than calling another method:



```java
/**
 * Saves tasks to a file.
 * @param filePath the path to the file where tasks will be saved
 */
public void saveTasks(String filePath) {  2 usages  Zhang Y.D.
    taskManager.saveToFile(filePath);
}
```

Hope the next project will look better and cleaner…