Project Report

Group 014

COMP2021 Object-Oriented Programming (Fall 2023)

Author: ZHEN ZHANG Alex

Other group members:

LIN Zhanzhi

Raynell Lu Soon Hong

KONG Zirui

1 Introduction

This document describes Group 014's design and implementation of a command-line-based task management system. The project is part of the course COMP2021 Object-Oriented Programming at PolyU.

2 My Contribution and Role Distinction

In this project I worked along with three amazing people LIN Zhanzhi, KONG Zirui and Raynell Lu Soon Hong, in which was a great experience to share our ideas among ourselves and learn about the importance and the functionality of a Command-Line Task Management System. This project was divided into many subtasks which included writing the code for the application, the user manual, each member's individual report and presentation slides including the presentation video. I did some research on how a Command-Line Task Management System works and learned the key components of it. Compared to the other group members, my role was mainly focused on the presentation part, gathering all the ideas implemented on the code, inside the presentation I worked on, I divided it into 5 parts, team introduction, project architecture, project design, project implementation and discussion. Throughout the presentation creation, for the team introduction part, I added each member of the group, next for project architecture I included what is the main function of a Command-Line Task Management System and what its usage is, also included what the systems keys components are and explained the purpose of each component. Next, for the project design, I divided it into three subparts, the task structure in which consists of differencing the purpose of a primitive task and composite task, afterwards comes the task functions which resembles the main functionality of the project which are task creation, duration and its prerequisites. And then the modification functions which are mainly functions that modify the task like undo/redo, change and delete. Furthermore, it also contains the potential of the features, essentials for the management of the tasks like scheduling and prioritization, dependency visualization and efficiency and simplicity, briefly describing each one of them. The fourth part is the project implementations, it contains the java functions our group used while creating the

application code, our group used the java function HashMap in which I highlighted 3 important aspects of this java function HashMap, the Built-in functionality the retrieval efficiency and the flexibility. Also included the algorithm for outputting the minimum time and its steps for accomplishing its objective. Finally, the discussion we talked about what we have learned and improved using Object Orientation in coding.

3 A Command-Line Task Management System

3.1 Design

In the diagram, the system is designed following the Model-View-Controller (MVC) pattern. The Application class acts as the user interface (View), the TMS class acts as the controller, and the other classes serve as the model.

The Application (GUI) communicates user input to the TMS and displays execution messages from the TMS. The TMS, as the controller, interprets commands, assigns tasks to the appropriate model components, and communicates execution messages. Additionally, the TMS manages the program's state, supporting operations like saving, loading from files, and reverting to previous states.

The Model components, represented by other classes, receive instructions from the controller, perform tasks through communication, and provide feedback via execution messages.

Our System is composed of 10 classes, which have 1 *public* class (*TMS*) and 9 *private* classes (*Task, TaskSet, Node, Criterion, BasicCriterion, BinaryCriterion, NegatedCriterion, CriterionSet* and *FinishTime*).

TMS Class is the core class in TMS.java. It contains 10 functions including 2 *boolean* functions, 6 *void* functions and 2 String functions.

Task class is used to manipulate the data for the Task object.

TaskSet class is used to store, create, delete and manipulate Task within the set.

FinishTime class is used to calculate the duration of task(s).

Node class is used to manipulate the Arraylist used in the system.

CriterionSet class is used to manipulate actions that can be acted upon criteria for a task.

BinaryCriterion class is used to combine two different basic criteria's onto one binary criterion.

NegatedCriterion class is used to negate one existing basic criterion.

BasicCriterion class is used for the creation of a basic criterion.

Criterion class is used to set as a criterion to search for needed tasks.

3.2 Requirements

[REQ1] The system should support creating primitive tasks.

1) The requirement is implemented.

2) The arguments are validated, it goes to the task constructor in which creates a new Task. Then, this new task is registered in the TaskSet.

3) Errors such as invalid parameters, invalid format, duplicate tasks, such as name, description or duration, it will be noticed to the user. In the case that the given name is duplicated, or all the prerequisites are not fulfilled, the process will stop and it will report an error.

[REQ2] The system should support creating composite tasks.

1) The requirement is implemented.

2) The arguments are validated, it goes to the task constructor in which creates a new Task. Then, this new task is registered in the TaskSet.

3) Errors such as invalid parameters, invalid format, duplicate tasks, such as name, description, it will be noticed to the user. In the case that the given name is duplicated, or all the prerequisites are not fulfilled, the process will stop and it will report an error.

[REQ3] It should support deleting tasks.

1) The requirement is implemented.

2) The arguments are validated, then it will check the deletability of the task. If the task is deletable, it is delated with their relevant tasks.

3) If the given name is invalid, the system will stop the execution and it will report an error to the user. If the given task does not correspond to any exiting tasks, the system will stop working and will notice the

user about the non existing task. If the given task is not delatable, the system will terminate and will inform the user about the usableness to delete the task.

[REQ4] It should support changing the properties of an existing task.

1) The requirement is implemented.

2) The arguments are validated. Then it will check if the new task is changeable, if the new task is changeable, it will apply a relevant update onto it.

3) If the new value is invalid, the system will stop and report an error. If the new name is already used, then it will stop and report an error. It also applies in the same way for prerequisites if they are incompletable tasks.

[REQ5] It should support printing the information of a task.

1) The requirement is implemented.

2) The arguments are validated. It will print out the information of the task including its attributes.

3) If the given argument is invalid, the system will stop and will report an error. It also applies if the given name does not correspond to any task.

[REQ6] It should support printing the information of all tasks.

1) The requirement is implemented.

2) It will print all the tasks that have been registered. All the tasks registered will be printed along with information.

3) No error cases.

[REQ7] It should support reporting the duration of a task.

1) The requirement is implemented.

2) The arguments are validated, It will start calculating the duration of the task, for primitive tasks, the duration will be stored during its definition and for composite tasks, its duration is calculated by building a graph with the task and its subtasks.

3) If the given name is invalid, the system will stop and will report an error. It also applies if the given name does not correspond to any task.

[REQ8] It should support reporting the earliest finish time of a task.

1) The requirement is implemented.

2) The arguments are validated. The earliest time for the task is calculated. For composite task, it will be the same as the duration, and for primitive task, it takes to count the minimum hours required to finish the task.

3) If the given name is invalid, the system will stop and will report an error. It also applies if the given name does not correspond to any task.

[REQ9] It should support defining basic task selection criteria.

1) The requirement is implemented.

2) The argument is validated. I will process to be identified by the critical path of the system, then it will calculate the latest start and finish time of each task.

3) If the given argument is invalid, the system will stop and will report an error. It also applies if the given name does not correspond to any task.

[REQ10] It should support a criterion to check whether a task is primitive.

1) The requirement is implemented.

2) It will create a IsPrimitive criterion. This criterion is implemented as a basic criterion to check wheter the duration is greater than 0. If so, then the IsPrimitive criterion evaluates as true.

3) No error cases.

[REQ11] It should support defining composite task selection criteria.

1) The requirement is implemented.

2) The argument is validated. Then it will create and register a new corresponding criterion.

3) If the given name is invalid, the system will stop and will report an error. It also applies if the given name does not correspond to any task.

[REQ12] The tool should support printing all defined criteria.

1) The requirement is implemented.

2) It will collect the defined criteria from the registry, CriterionSet and it wil be outputting the imformation one after another. For the criteria that are composed of other criteria, the printing is recursive until it revilves to the information of a basic criterion.

3) No error cases.

[REQ13] The tool should support searching for tasks based on an existing criterion.

1) The requirement is implemented.

2) Each registered task is tested onto one condition, after passing that condition, it will be outputted.

3) If the serach criterion is not defined, the process will be terminated, and it will report an error.

[REQ14] The tool should support storing the defined tasks and criteria from a file.

1) The requirement is implemented.

2) It will certify that the file exists in the given path and it saves it in the current program state, taskset and criterionset from the file.

3) If the given path is invalid, the process will be terminated and it will report an error.

[REQ15] The tool should support loading tasks and criteria from a file.

1) The requirement is implemented.

2) It will certify that the file exists in the given path and it will read the current program state, taskset and criterionset from the file.

3) If the given path is invalid, the process will be terminated and it will report an error. It also applies if the file is organized incorrectly.

[BON1] Support for a graphical user interface (GUI) to visually show the tasks created or selected and their relationship.

1) The requirement is implemented.

2) In the execution of the program, it will create three components. The first component will let the user handle the input and output messages. Next, the second component will let the users enter the commands. And the third component will trigger a communication with the TMS using the entered command.

3) To prevent the excision of results, in the botton there will be a scroll bar in which shows the lasts outputs. When the users adjust the dimensions of thye frame, the three components will adjust simultaneosly.

[BON2] Support for the undo and redo of all the required commands except PrintTask, PrintAllTask, ReportDuration, ReportEarliestFinishTime, PrintAllCriteria, Search, Store, Load and Quit.

1) The requirement is implemented.

2) When using undo, the state the is currently in use, is moved to the top of the redo stack and it will be stored at the top of the undo will be retored. For redo, the state of the top of redo stack is restored, in which it becomes the top of the undo stack. When applying it to the program, the redo stack gets cleared because the program state will be developed in another branch.

3) If the undo stack contains only the current state, it will terminate the undo operation and it will send a report. For redo, if the redo stack is empty, it will terminate the process of the redo operation and will send a report.

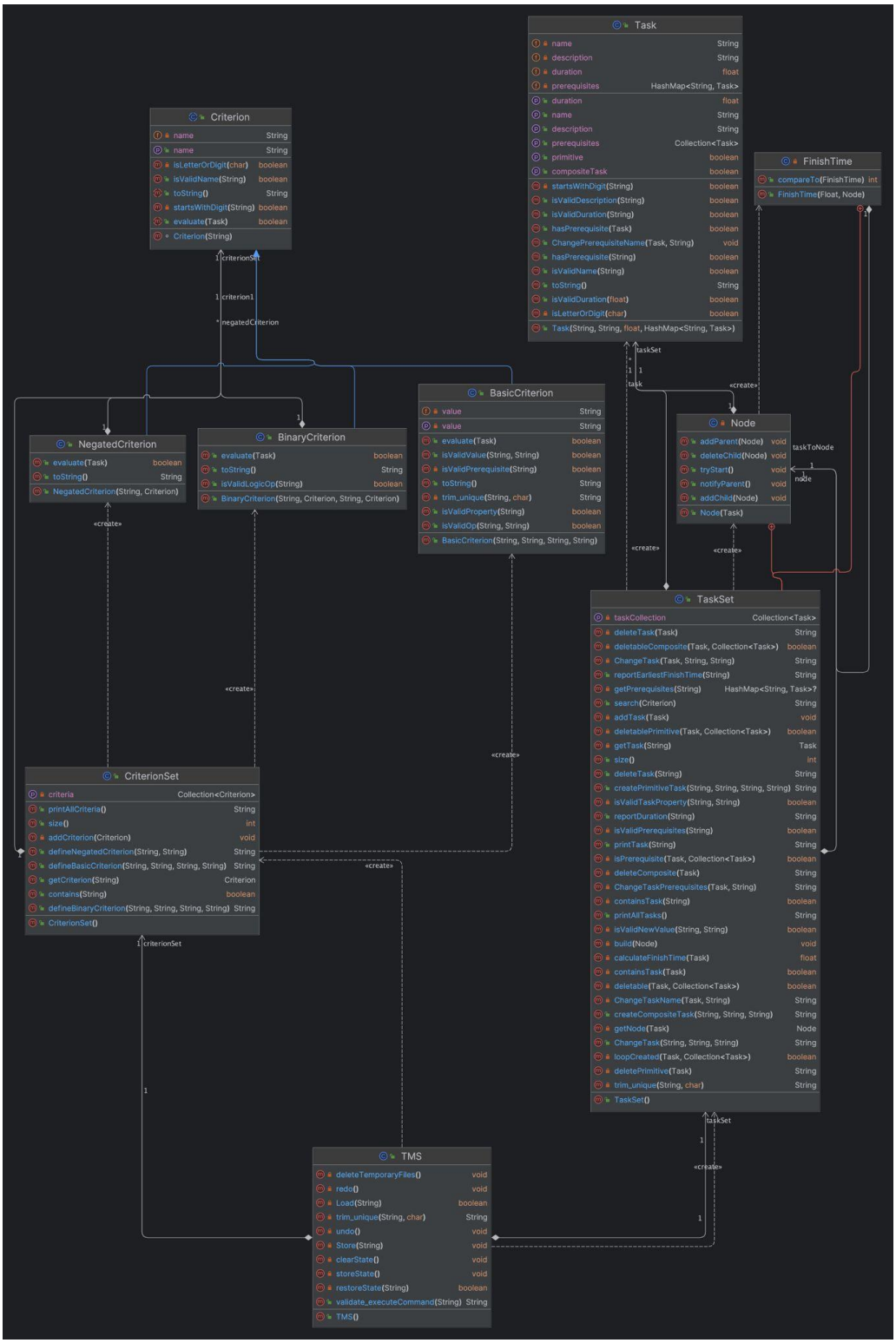

Diagram 1 Struture Design