USER MANUAL

COMP2021 Object-Oriented Programming (Fall 2023)

Group Project

Task Management System

Group_014

Group Members:

LIN Zhanzhi Eric

KONG Zirui Jerry

LU Soon Hong Raynell

ZHEN ZHANG Alex

NOTICES

- **No** part of the manual included with this TMS may be rewritten, transmitted, transcribed, stored in a retrieval system, or translated into any language in **any** form, by **any** means, **without** COMP2021 Object-Oriented Programming (Fall 2023) Group 014' s prior written permission.

- COMP2021 Object-Oriented Programming (Fall 2023) Group 014 reserves the right to change the specifications of the hardware and software described in this manual at **any** time and **without** prior notice.

- COMP2021 Object-Oriented Programming (Fall 2023) Group 014 will **not** be held liable for **any** damages resulting from the use of the TMS.

- While every effort has been made to ensure that the information in this manual is accurate and complete, we would appreciate it were you bring any errors or omissions to the attention of any group member of COMP2021 Object-Oriented Programming (Fall 2023) Group 014 (email address provided at the end of this manual).

CONTENTS

1 Introduction

1.1 Purpose of the User Manual

1.2 Overview of the Application

1.3 System Requirements

2 Getting Started

2.1 Installation and Setup

2.2 User Interface Overview

3 Task Management Instructions

3.1 Creating Tasks

3.1.1 Creating Primitive Tasks

3.1.2 Creating Composite Tasks

3.2 Viewing Tasks

3.2.1 Task Details

3.2.2 Printing All Tasks

3.2.3 Printing One Task

3.3 Updating Tasks

3.3.1 Modifying Task Name

3.3.2 Modifying Task Description

3.3.3 Modifying Primitive Task Duration

3.4 Deleting Tasks

3.5 Duration of Tasks

3.5.1 ReportDuration

3.5.2 ReportEarliestFinishTime

3.6 Criterions

3.6.1 Define Basic Criterion

3.6.2 IsPrimitive

3.6.3 Define Negated Criterion

3.6.4 Define Binary Criterion

3.6.5 Print All Criterion

3.7 Other Commands

3.7.1 Search

3.7.2 Store

3.7.3 Load

3.7.4 Undo

3.7.5 Redo

3.7.6 Quit

4 Additional Resources

4.1 Useful Links

4.2 Contact Us

1 Introduction

1.1 Purpose of the User Manual

- Thank you for using our Command-Line Task Management System designed by LIN Zhanzhi, KONG Zirui, Raynell Lu Soon Hong, and ZHEN ZHANG Alex. To get the most from our TMS, please be sure to read all instructions thoroughly and keep them where they will be read by all who use our TMS.
- The user manual is for the user who is not familiar with the Command-Line Task Management System. With this instruction, we believe that you will proficiently use this system.

1.2 Overview of the Application

- Our TMS system utilizes the MVC (Model–View–Controller) architecture as per the requirement stated in the question prompt. The TMS system is built upon 11 different classes, with the class *TMS* serving as the Controller components. Class *Application*, which is the GUI serving as the Viewing component. Lastly, the remaining classes are the Model parts.

1.3 System Requirements

- After testing our TMS, it has been found that all common computer operating systems (Windows, Mac OS X, Linux, Solaris) can use our TMS. Please note that before using TMS, make sure that Java Runtime Environment (JRE) is installed on your computer. If not, please click on Java Downloads for All Operating Systems and download it.

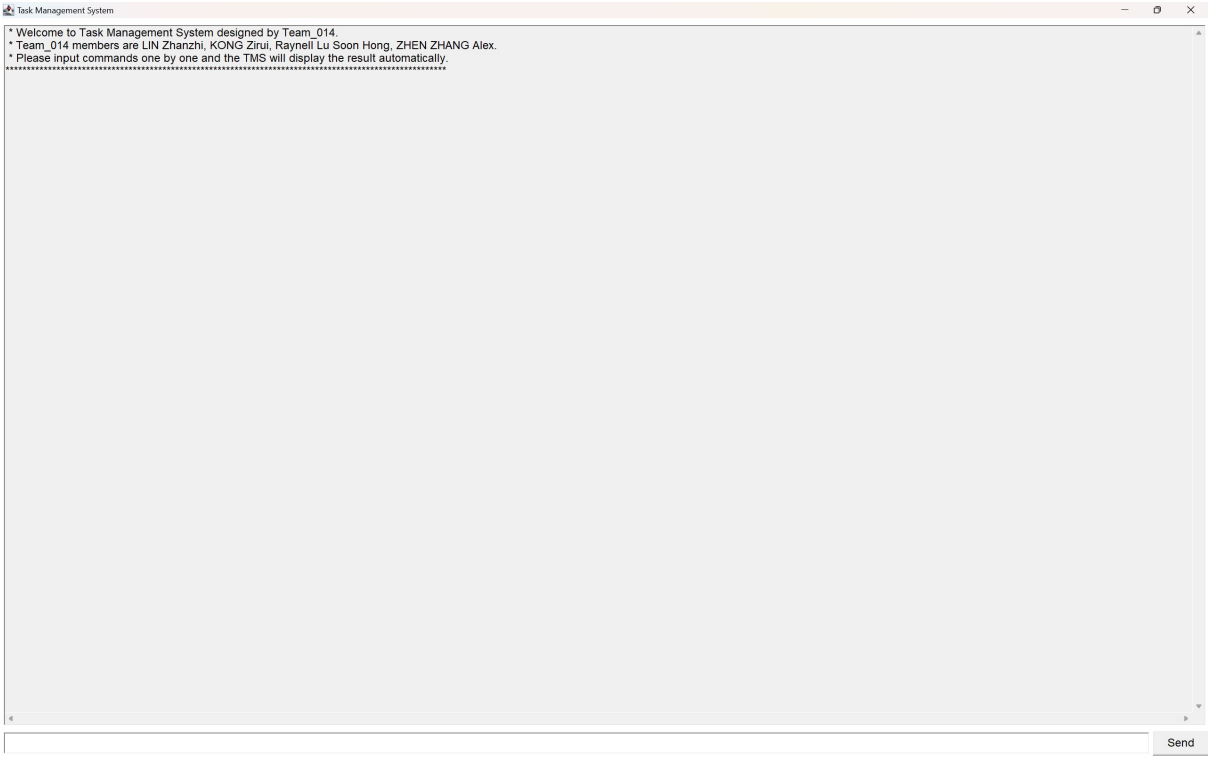
2 Getting Started

2.1 Installation and Setup

- Please make sure you have downloaded Application.jar and check whether there are any other applications that may interfere with the normal operation of TMS.

2.2 User Interface Overview

The following picture is our user interface:



3 Task Management Instructions

Please Pay Attention to VALIDITY:

1) The format of each method **must** be in the right format. Otherwise, it will return an **error**.
Correct example: CreatePrimitiveTask WD WashDishes 0.5 ,
Wrong example (missing prerequisites): CreatePrimitiveTask WD WashDishes 0.5

2) The capital for the input also **matters greatly**, the **incorrect** capitals will result in an **error**.
Correct example: CreatePrimitiveTask WD WashDishes 0.5 ,
Wrong example (lower case in CreatePrimitiveTask): createprimitivetask WD WashDishes 0.5 ,

3) For a name to be valid, it should suit the criteria of **eight** characters of either **English letters** or **digits**.
Correct example: CreatePrimitiveTask WD WashDishes 0.5 ,
Wrong example (invalid characters): CreatePrimitiveTask 哈喽 WashDishes 0.5 ,

4) For a duration to be valid, it should be a **valid positive float** number.
Correct example: CreatePrimitiveTask WD WashDishes 0.5 ,
Wrong example (negative float number): CreatePrimitiveTask WD WashDishes –0.5 ,

3.1 Creating Tasks

3.1.1 Creating Primitive Tasks

Command: CreatePrimitiveTask name description duration prerequisites

Effect: It creates a new primitive task.

Example: CreatePrimitiveTask WC WashCar 0.4 ,

CreatePrimitiveTask SC SoapCar 0.2 ,

CreatePrimitiveTask DC DryCar 0.4 ,

```
Command: CreatePrimitiveTask WC WashCar 0.4 ,
Done. Primitive Task "WC" created.
Task Name: WC
        Description: WashCar
        Duration: 0.4
        Prerequisites:

Command: CreatePrimitiveTask SC SoapCar 0.2 ,
Done. Primitive Task "SC" created.
Task Name: SC
        Description: SoapCar
        Duration: 0.2
        Prerequisites:

Command: CreatePrimitiveTask DC DryCar 0.4 WC,SC
Done. Primitive Task "DC" created.
Task Name: DC
        Description: DryCar
        Duration: 0.4
        Prerequisites: SC WC
```

3.1.2 Creating Composite Tasks

Command: CreateCompositeTask name0 description name1,name2,...,namek

Effect: It creates a composite task. The new task is named name0 and has k subtasks, named name1, name2, ..., namek, and you need atleast 2 subtasks (k ⩾ 2)

Example: CreateCompositeTask WP WholeProcess WC,SC,DC

```
Command: CreateCompositeTask WP WholeProcess WC,SC,DC
Done. Composite Task "WP" created.
Task Name: WP
        Description: WholeProcess
        Duration: 0.0
        Sub-tasks: SC WC DC
```

3.2 Viewing Tasks

3.2.1 Task Details

Effect: It shows the task name, description, duration and prerequisites of that specific task.

Example: Task Name: name

Description: description

Duration: duration

Prerequisites: prerequisites

3.2.2 Printing All Tasks

Command: PrintAllTasks

Effect: It prints out the information of all existing tasks.

Example: PrintAllTasks

```
Command: PrintAllTasks
4 defined tasks.
Task Name: SC
        Description: SoapCar
        Duration: 0.2
        Prerequisites:
Task Name: WP
        Description: WholeProcess
        Duration: 0.0
        Sub-tasks: SC WC DC
Task Name: WC
        Description: WashCar
        Duration: 0.4
        Prerequisites:
Task Name: DC
        Description: DryCar
        Duration: 0.4
        Prerequisites: SC WC
```

3.2.3 Printing One Task

Command: PrintTask name

Effect: It prints out the information of a certain task.

Example: PrintTask WP

```
Command: PrintTask WP
Task Name: WP
        Description: WholeProcess
        Duration: 0.0
        Sub-tasks: SC WC DC
```

3.3 Updating Tasks

3.3.1 Modifying Task Name

Command: ChangeTask name property newValue

Effect: It changes the name of a task to a new name.

Example: ChangeTask WC name WashC

```
Command: ChangeTask WC name WashC
Done. Task "WC" changes name into "WashC"
```

3.3.2 Modifying Task Description

Command: ChangeTask description property newValue

Effect: It changes the description of a task to a new description.

Example: ChangeTask WashC description WashCarWithWater

```
Command: ChangeTask WashC description WashCarWithWater
Done. Task "WashC" changed description into "WashCarWithWater"
```

3.3.3 Modifying Primitive Task Duration

Command: ChangeTask name duration new_duration

Effect: It changes the duration of a primitive task to a new updated duration.

Example: ChangeTask WashC duration 0.3

```
Command: ChangeTask WashC duration 0.3
Done. Task "WashC" changed duration into 0.3
```

3.4 Deleting Tasks

Command: DeleteTask name

Effect: It deletes a primitive task without affecting the rest of the system.

Example: DeleteTask PC

```
Command: CreatePrimitiveTask PC PolishCar 0.5 ,
Done. Primitive Task "PC" created.
Task Name: PC
        Description: PolishCar
        Duration: 0.5
        Prerequisites:

Command: DeleteTask PC
Done. Task "PC" is deleted.
```

3.5 Duration of Tasks

3.5.1 ReportDuration

Command: ReportDuration name

Effect: It reports the duration of a task.

Example: ReportDuration WashC

```
Command: ReportDuration WashC
The duration of task "WashC" is 0.3 hours.
```

3.5.2 ReportEarliestFinishTime

Command: ReportEarliestFinishTime name

Effect: It reports the earliest finish time of a task.

Example: ReportEarliestFinishTime WP

```
Command: ReportEarliestFinishTime WP
Starting at 0.0 hour, the earliest finish time of task "WP" is 0.70000005 hour.
```

3.6 Criterions

3.6.1 Define Basic Criterion

Command: DefineBasicCriterion name1 property op value

Effect: It defines a basic task selection criterion.

Example: DefineBasicCriterion LS1 duration < 1

```
Command: DefineBasicCriterion LS1 duration < 1
Done. Basic Criterion "LS1" created: duration < 1
```

3.6.2 IsPrimitive

Command: Search IsPrimitive

Effect: Evaluates if a task is primitive or not.

Example: Search IsPrimitive

```
Command: Search IsPrimitive
Search result: 3 tasks:
Task Name: SC
        Description: SoapCar
        Duration: 0.2
        Prerequisites:
Task Name: WashC
        Description: WashCarWithWater
        Duration: 0.3
        Prerequisites:
Task Name: DC
        Description: DryCar
        Duration: 0.4
        Prerequisites: SC WashC
```

3.6.3 Define Negated Criterion

Command: DefineNegatedCriterion name1 name2

Effect: To create a composite criterion named name1 using DefineNegatedCriterion, they can make name1 the opposite of an existing criterion named name2.

Example: DefineNegatedCriterion MT1 LS1

```
Command: DefineNegatedCriterion MT1 LS1
Done. Negated Criterion "MT1" defined: !(duration < 1)
```

3.6.4 Define Binary Criterion

Command: DefineBinaryCriterion name1 name2 logicOp name3

Effect: To create a composite criterion named name1 using DefineBinaryCriterion, they can combine an existeing criterion named name2 with another criterion named name3 using logical operators like && or ||.

Example: DefineBinaryCriterion BWT0208 MT02 && LS08

```
Command: DefineBasicCriterion MT02 duration > 0.2
Done. Basic Criterion "MT02" created: duration > 0.2

Command: DefineBasicCriterion LS08 duration < 0.8
Done. Basic Criterion "LS08" created: duration < 0.8

Command: DefineBinaryCriterion BWT0208 MT02 && LS08
Done. Binary Criterion "BWT0208" created: (duration > 0.2) && (duration < 0.8)
```

3.6.5 Print All Criterion

Command: PrintAllCriteria

Effect: It prints out all the criteria defined.

Example: PrintAllCriteria

```
Command: PrintAllCriteria
6 defined criteria.
IsPrimitive
duration < 1
!(duration < 1)
duration < 0.8
duration > 0.2
(duration > 0.2) && (duration < 0.8)
```

3.7 Other Commands

3.7.1 Search

Command: Search name

Effect: It list all tasks that satisfy the criterion with the given name.

Example: Search BWT0208

```
Command: Search BWT0208
Search result: 2 tasks:
Task Name: WashC
        Description: WashCarWithWater
        Duration: 0.3
        Prerequisites:
Task Name: DC
        Description: DryCar
        Duration: 0.4
        Prerequisites: SC WashC
```

3.7.2 Store

Command: Store path

Effect: Stores the defined tasks and criteria into the file at path.

Example: Store C:\Users\RaynelI\OneDrive\桌面\testfile.txt

```
Command: Store C:\Users\RaynelI\OneDrive\桌面\testfile.txt
Current state stored: 4 tasks and 6 criteria successfully!
```

3.7.3 Load

Command: Load path

Effect: Loads the defined tasks and criteria from the file at path.

Example: Load C:\Users\RayneII\OneDrive\桌面\testfile.txt

```
Command: Load C:\Users\RayneII\OneDrive\桌面\testfile.txt
Loaded 4 Tasks and 6 criteria successfully!
```

3.7.4 Undo

Command: Undo

Effect: It restores the previous action.

Example: undo

```
Command: CreatePrimitiveTask PM PayMoney 0.01 ,
Done. Primitive Task "PM" created.
Task Name: PM
        Description: PayMoney
        Duration: 0.01
        Prerequisites:

Command: undo
Undo successful.
```

3.7.5 Redo

Command: Redo

Effect: It redoes the action.

Example: redo

```
Command: CreatePrimitiveTask PM PayMoney 0.01 ,
Done. Primitive Task "PM" created.
Task Name: PM
        Description: PayMoney
        Duration: 0.01
        Prerequisites:

Command: undo
Undo successful.

Command: redo
Redo successful.
```

3.7.6 Quit

Command: Quit

Effect: Terminates the execution of the system.

Example: Quit

4 Additional Resources

4.1 Useful Links

There are some useful links here:

PolyU Website: www.polyu.edu.hk

Java Runtime Environment Download: www.java.com/en/download/manual.jsp

PolyU COMP Website: www.polyu.edu.hk/comp

HKPU COMP Intranet: intranet.comp.polyu.edu.hk

GitHub Website: github.com

4.2 Contact Us

If you have any questions or problems, feel free to contact us by E-mail if necessary:

LIN Zhanzhi: 22097456d@connect.polyu.hk

KONG Zirui: 22103493d@connect.polyu.hk

Raynell Lu Soon Hong: 22104193d@connect.polyu.hk

ZHEN ZHANG Alex: 22103518d@connect.polyu.hk