Command-Line Task Management System
Individual Project Report

Project Group 3

COMP2021 Object-Oriented Programming (Fall 2023)

Author: Lee Na TAI
SID: 22076162d

Other group members:
Bing Yui HON 22079086d
Yui Hei HUANG 22079056d
Muze XIANG 22106753d

Contents

1. Introduction *3*

2. My Contributions and Role Distinction *3*

2.1. Requirement analysis 3

2.2 System Design 3

2.3 Coding 3

3. A Command-Line Task Management System *4*

3.1 Design 4

3.1.1 Overall Conceptual Design 4

3.1.2 UML Design 5

3.2 Requirements 6

[REQ5] 6

[REQ6] 6

[REQ7] 7

[REQ8] 7

4. GROUP 3 User Manual for Task Management System (TMS) *9*

1. Introduction

In today's fast-paced modern society, individuals are frequently assigned a multitude of tasks, and multitasking has become an integral part of our lifestyle. The Command-Line Task Management System holds significant importance due to its ability to streamline task organization, enhance the management and boost overall productivity.

This project report aims to provide a comprehensive overview of the implementation of the Task Management System developed by our group. The system is a software application designed to assist users in efficiently organizing and tracking their tasks. The report will delve into various aspects, including my role and contribution to the development of the system, our design concepts, fundamental functionalities, and a detailed user manual for the Command-Line Task Management System.

2. My Contributions and Role Distinction

Throughout the development of the Command-Line Task Management System, our team made valuable contributions in various crucial aspects of the project. This included conceptualizing the program design, allocating work assignments, implementing coding skills, and engaging in passionate discussions to address encountered challenges over the complex functionalities within the system and explore potential improvements, such as code optimization.

During the process, I have participated actively in the following parts:

2.1. Requirement analysis

I played an active role in the initial phase of the project by engaging in requirement analysis. I had been collaborating with my team both on-site and remotely to identify the functional and non-functional requirements of the Command-Line Task Management System. Through research and discussions, I contributed to the creation of our workflow and functionality ideas.

2.2 System Design

In the design phase, I have participated in architectural discussions with my groupmates. During our discussions, we evaluate our individual designs ideas collectively to select the most suitable framework.

After that, we optimize in refining the chosen design by studying and learning from the advantages of each other's designs. Through this process, we gradually synthesized the best elements from our individual proposals, resulting in a comprehensive and cohesive final design.

2.3 Coding

During the development phase, I undertook the responsibility of developing the following

functionalities based on the project description: [REQ5] PrintTask, [REQ6] PrintAllTasks, [REQ7] ReportDuration and [REQ8] ReportEarliestFinishTime.

3. A Command-Line Task Management System

3.1 Design

3.1.1 Overall Conceptual Design

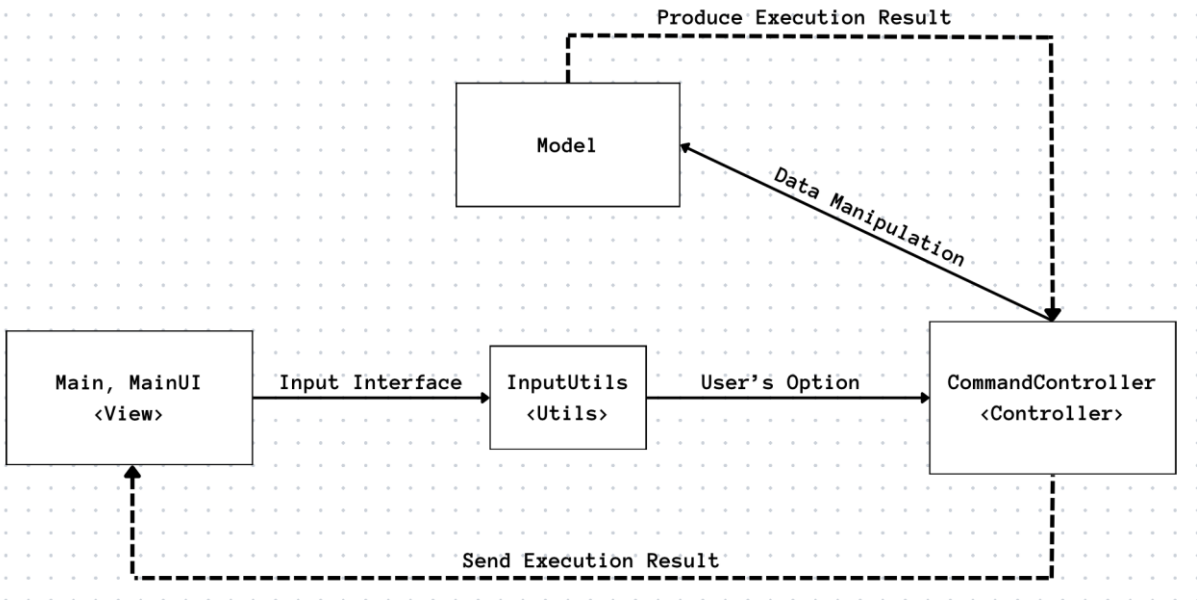


Figure 1: conceptual design

In our overall system design, we have incorporated the MVC (Model-View-Controller) design pattern (See Figure 1). Both the Main and MainUI classes are considered as the View, where users interact with the application. In both Main and MainUI, a menu of commands is provided to users. We have included a 'Help' command to provide users with a detailed guide on the available commands when they enter 'Help'.

To provide an input interface for users, the InputUtils class will be utilized. The InputUtils class essentially contains a Scanner class imported from the Java library, which is implemented to prompt users for input and retrieve their commands. It includes a method called read() that returns the users' input values.

Next, the users' input values will be evaluated to determine one of the functionalities in the View before being passed to the CommandController, which is an essential component of the system. The CommandController acts as the controller of our program, facilitating communication between the View and the Model. For each user's option processed from their input values, the CommandController identifies the arguments and checks their presence to ensure readiness for executing the selected functionality.

Once each argument from the command is checked, the arguments are passed from the Controller to the Model, which is a major component of the system responsible for data manipulation. The Model stores, inserts, deletes, and updates the data after performing the necessary checks. Finally, the Model returns an execution result, which includes a success message or, if applicable, an error message.

The execution result is represented by a Result object, which generally includes a Boolean value indicating the success of the execution and a String containing the result message. The View receives this object from the Controller. This design allows the system to provide users with a response displaying the execution result, indicating whether it was successful or not.

3.1.2 UML Design

In the actual implementation of the system, this is the UML class diagram of our application, which included the structure of the methods and all the key objects defined in the program. (See Figure 2 & Figure 3)

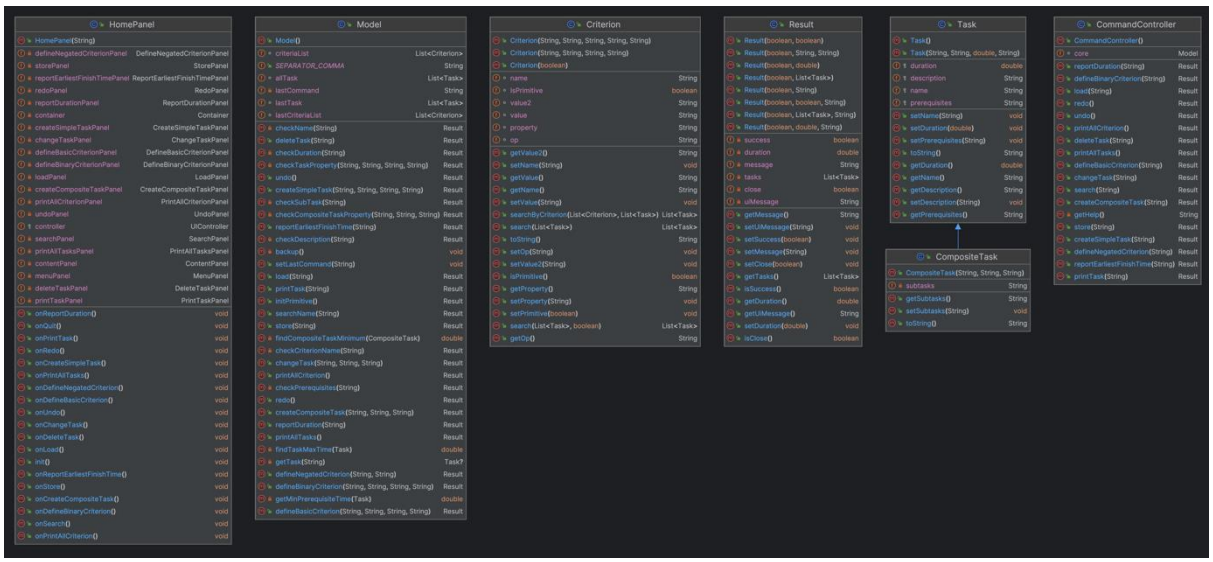


Figure 2: UML class diagram

Figure 3: UML class diagram

3.2 Requirements

[REQ5]

(1) The requirement is implemented.

(2) By invoking the method printTask(String option) of the CommandController object and passing the user's command input as a parameter, the method splits the user's command input into two parts: the main command and its input arguments. To return an execution result, this method invokes another method printTask(String name) of the Model object, taking the task name as a parameter, and performs the following actions:

1. Linear search: It searches for the target task by checking its name in the array list <allTask>, which stores all the tasks created by the user. The search concludes once the target task to be printed is found. The TextUtils class's .equals() method is used to check the name's length. The average time complexity of this search is O(n).

2. Printing the target task: We override the toString() method in the Task class to achieve this. The toString() method can be overridden since the Task class inherits from the Object class. The overridden toString() method returns a string with the format "property: "+the property+"\n" for every property of the Task object. Finally, the method returns a Result object indicating the success of the execution.

(3) We check the name length of the target task (see i), that is supposed to be at least 2 based on our project description, and the existence of the target task in the array list <allTasks> (see ii) respectively. For every error, we return a failure execution result and display the specific error message:

i. "Parameters error!"

ii. "Failed, task <the task name> does not exist!"

[REQ6]

1) The requirement is implemented.

2) By invoking the method printAllTasks() in the CommandController object. We do not take any parameters for the method. To return an execution result, the method invokes another method printAllTasks() in the Model object. It performs the following actions:

- Initialize an object inherited from the StringBuilder class imported from the Java Library. StringBuilder is responsible for string formatting.

- Print every task by iterating through the tasks in the array list <allTasks> with the format constructed using the StringBuilder object.

- Return an execution result

3) For any errors, a success execution result is not returned.

[REQ7]

(1) The requirement is implemented.

(2) By invoking the method reportDuration(String option) in the CommandController object, it takes the user's command input as the parameter. The method is splitting users' command input into two parts, main command, and its input arguments. To return an execution result, this method invokes another method reportDuration(String name) of Model object, taking the task name is a parameter, and performs the following actions:

- Linear search in searching for the target task by checking its name in the array list <allTask>

- If the target task is a simple task, get the duration of the target task by the method getDuration() inside the Task object. Print the message to report the duration.

- If the target task is a composite task, invoke the method findCompositeTaskMinimum() of the Model object to get the duration. This method performs the following actions:

A. Get the subtasks of the composite task.

B. By iterating through the subtasks of the composite task, get the maximum time needed for every subtask. To find the maximum time needed for every subtask, this invokes the method findTaskMaxTime(Task subtask), which calculates the sum of the duration of the subtask and the maximum duration of its prerequisites.

C. Print the message to report the duration.

(3) We check the name length of the target task (see i), that is supposed to be at least 2 based on our project description, and the existence of the target task in the array list <allTasks> (see ii) respectively. For every error, we return a failure execution result and display the specific error message:

i. "Parameters error!"

ii. "Failed, task <the task name> does not exist!"

[REQ8]

(1) The requirement is implemented.

(2) By invoking the method reportEarliestFinishTime(String option) in the CommandController object, it takes the user's command input as the parameter. The method is splitting users' command input into two parts: main command, and its input arguments. To return an execution

result, this method invokes another method reportDuration(String name) of Model object, taking the task name is a parameter, and performs the following actions:

- Linear search for the target task by checking its name in the array list <allTask>

- If the target task is a Composite Task, print the message: "Please don't select Composite Task"

- If the target task is a Simple Task, get duration by invoking the method getMinPrerequisiteTime(Task task). This method performs the following actions:

A. Get the prerequisites of the target task

B. Find the prerequisites with the minimum time needed recursively, by iterating through every prerequisites.

C. Return the sum of the duration of the target task and the minimum time needed for its prerequisites.

- Print the message to report the duration that is the earliest finish time of the target task.

(3) We check the name length of the target task (see i), that is supposed to be at least 2 based on our project description, and the existence of the target task in the array list <allTasks> (see ii) respectively. For every error, we return a failure execution result and display the specific error message:

i. "Parameters error!"

ii. "Failed, task <the task name> does not exist!"

-The END of the Report-

GROUP 3 User Manual for Task Management System (TMS)

The TMS offers both a Command Line Interface (CLI) and a Graphical User Interface (GUI) for convenient task management.

The TMS system implements all functions 1 to 16 as well as bonus1 and bonus2.

Creating Tasks
Create a Simple Task:
Command: CreateSimpleTask [name] [description] [duration] [prerequisites]
Description: Creates a simple task with specified details.
Example: CreateSimpleTask ReadBook "Read a book" 2h ""

Create a Composite Task:
Command: CreateCompositeTask [name] [description] [subTask]
Description: Creates a composite task that includes sub-tasks.
Example: CreateCompositeTask StudySession "Study session" ReadBook,WriteEssay
Modifying and Deleting Tasks

Delete a Task:
Command: DeleteTask [name]
Description: Deletes the specified task.
Example: DeleteTask ReadBook

Change a Task:
Command: ChangeTask [name] [property] [newValue]
Description: Modifies a specified property of a task.
Example: ChangeTask StudySession duration 3h

Reporting and Listing Tasks
Print Task Details:
Command: PrintTask [name]
Description: Displays details of a specific task.
Example: PrintTask StudySession

Print All Tasks:
Command: PrintAllTasks
Description: Lists all tasks in the system.

Report Task Duration:
Command: ReportDuration [name]
Description: Reports the total duration of a task.
Example: ReportDuration StudySession

Report Earliest Finish Time:

Command: ReportEarliestFinishTime [name]
Description: Reports the earliest time a task can be finished.
Example: ReportEarliestFinishTime StudySession

Defining Criteria for Tasks
Define Basic Criterion:
Command: DefineBasicCriterion [name] [property] [op] [value]
Description: Defines a basic criterion for task filtering.
Example: DefineBasicCriterion DurationCriterion duration > 1h

Define Negated Criterion:
Command: DefineNegatedCriterion [name] [negatedName]
Description: Defines a negated criterion.
Example: DefineNegatedCriterion NonUrgentCriterion UrgentTasks

Define Binary Criterion:
Command: DefineBinaryCriterion [name1] [name2] [logicOp] [name3]
Description: Combines two criteria using a logical operator.
Example: DefineBinaryCriterion CombinedCriterion UrgentTasks AND NonUrgentCriterion

Data Management
Store Data:
Command: Store [path]
Description: Saves the current state of the system to a file.
Example: Store data/savefile

Load Data:
Command: Load [path]
Description: Loads the system state from a file.
Example: Load data/savefile

Undo & Redo Operations
Undo an Action:
Command: Undo
Description: Reverses the most recent action.

Redo an Action:
Command: Redo
Description: Reapplies the most recently undone action.

Exiting the System
Command: Quit
Description: Exits the TMS.

Graphical User Interface (GUI)
The GUI provides a more visual and interactive way to manage tasks.
Users can perform similar actions as in the CLI but through graphical components like buttons, text

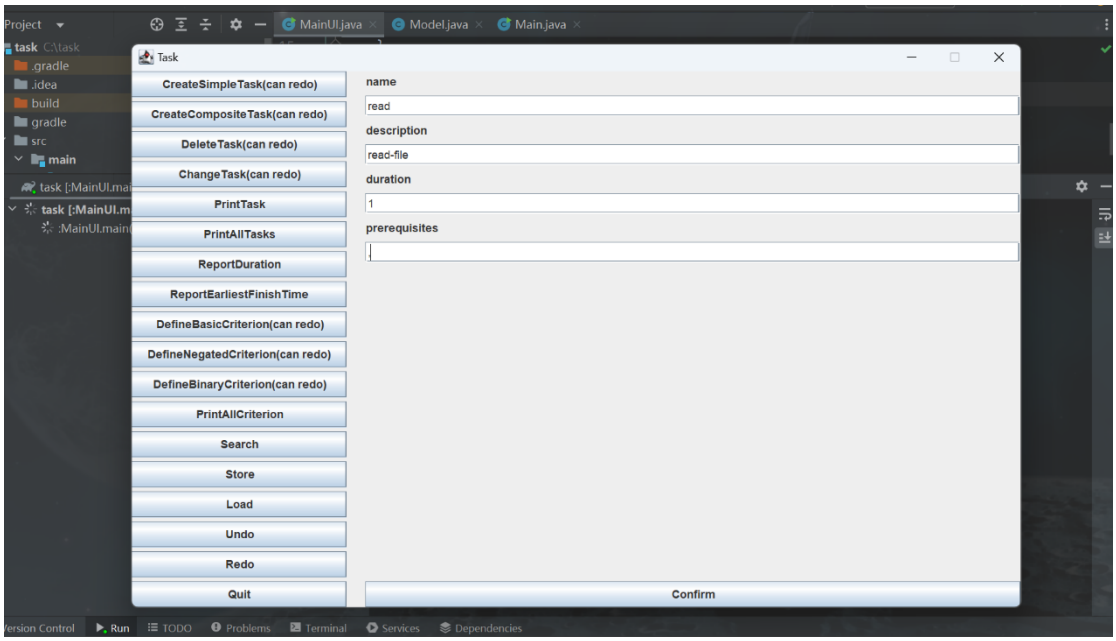
fields, and menus.

The operations of GUI and command line are roughly the same, but there are some differences between undo and redo.

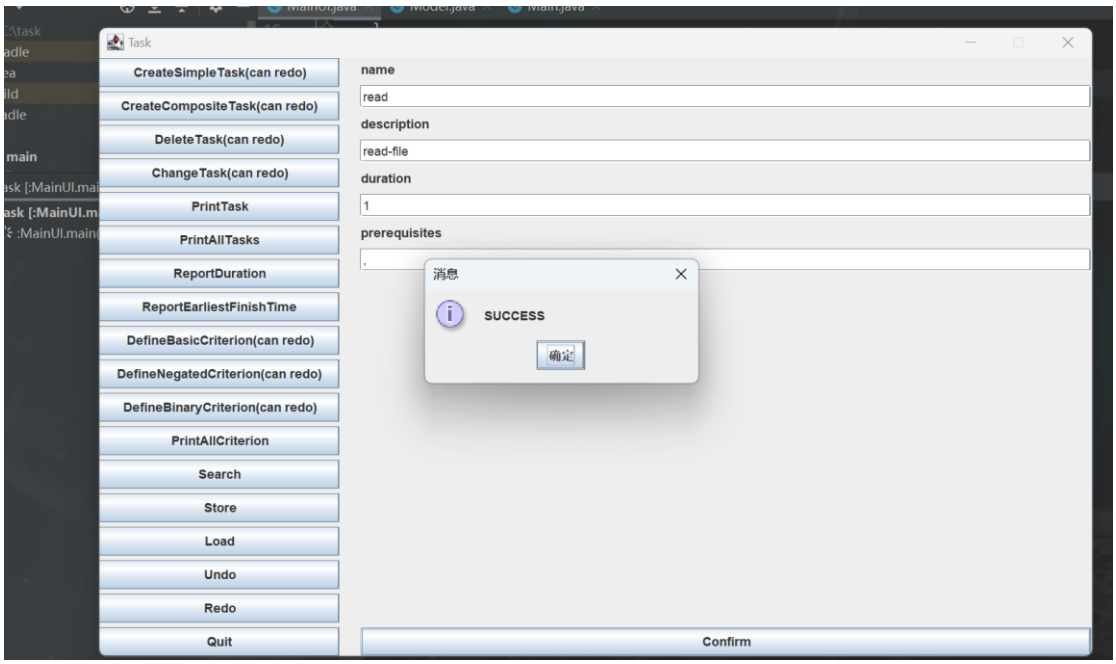
Redo instruction:

1. Input the value according to the GUI page.

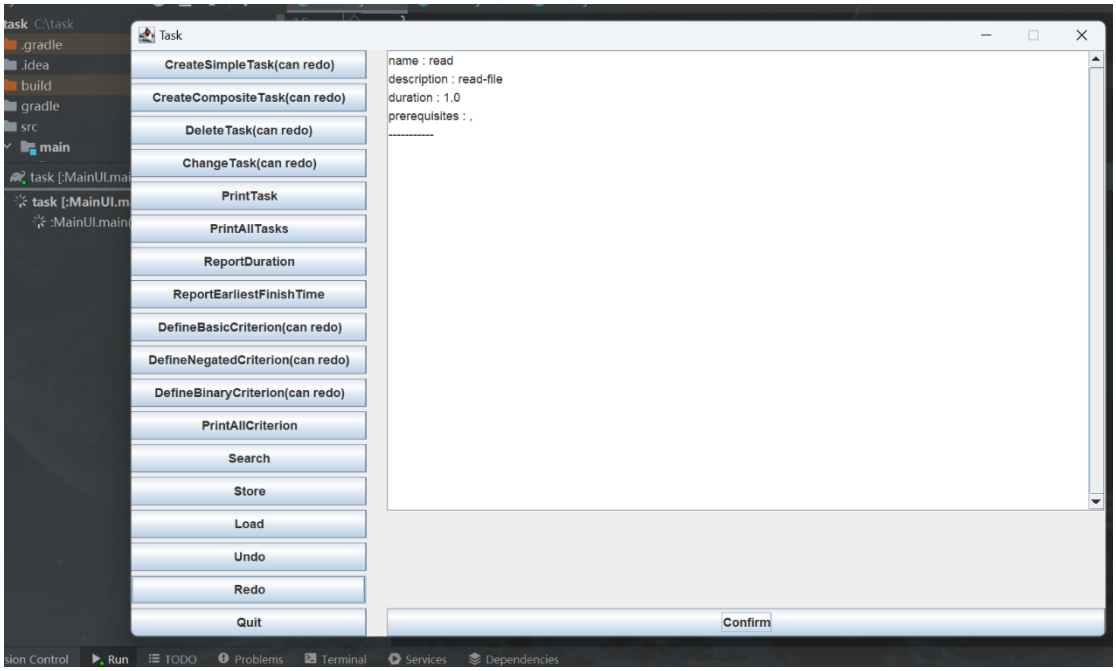
   Redo instruction:



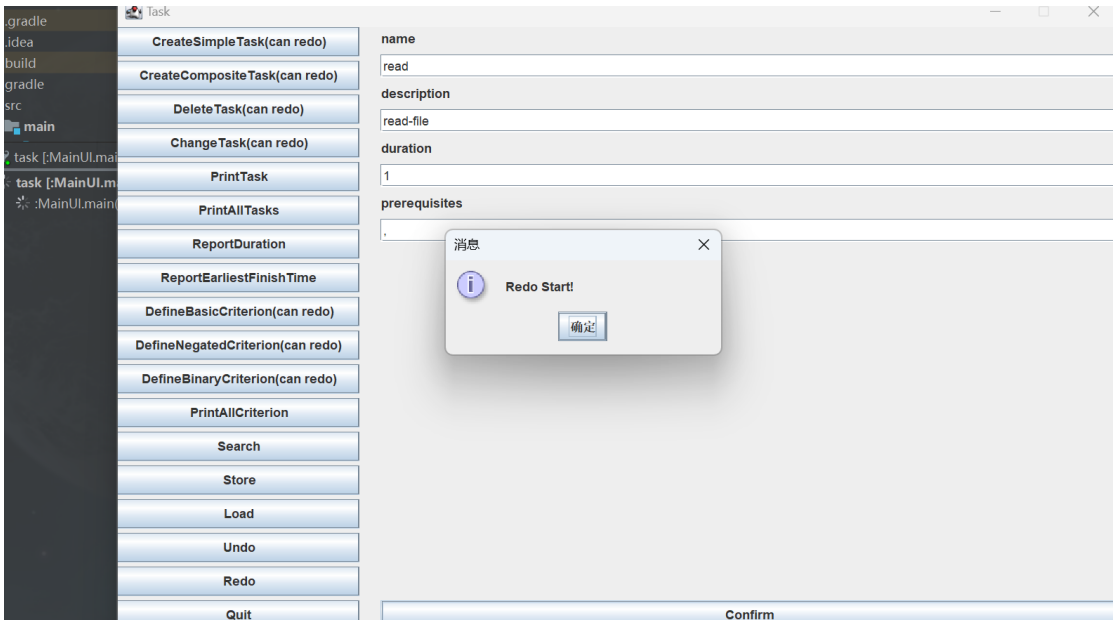
2. You need to confirm before you do next execution.



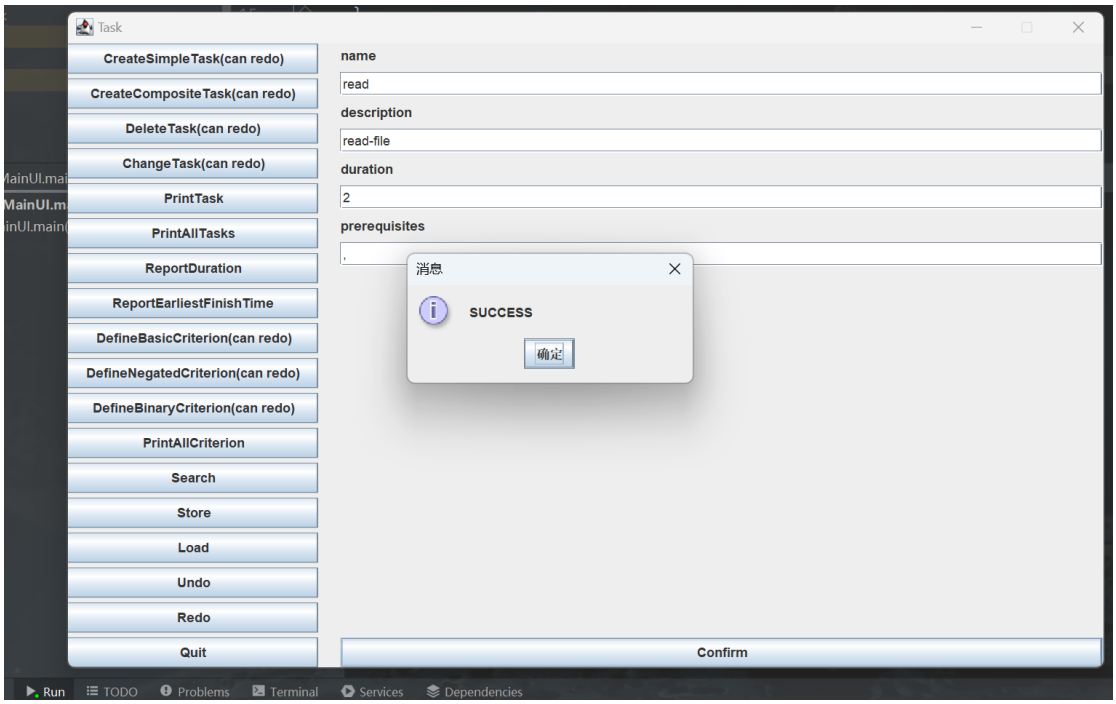
3. Print all task to show the panel.



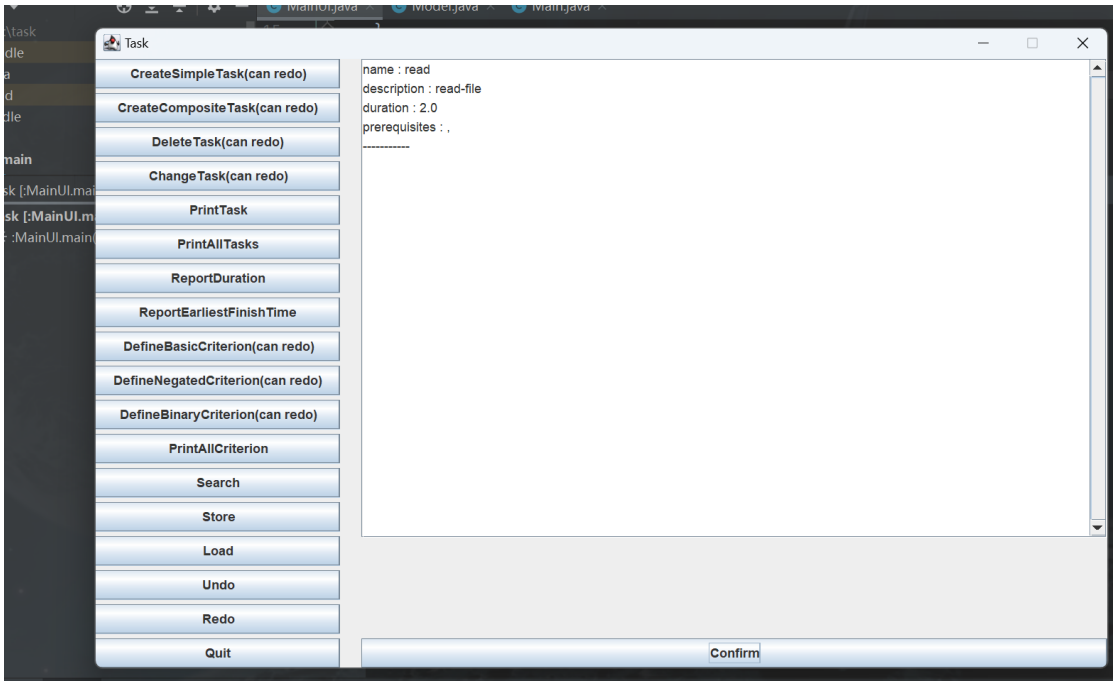
4. Click redo button



5. change duration to 2.

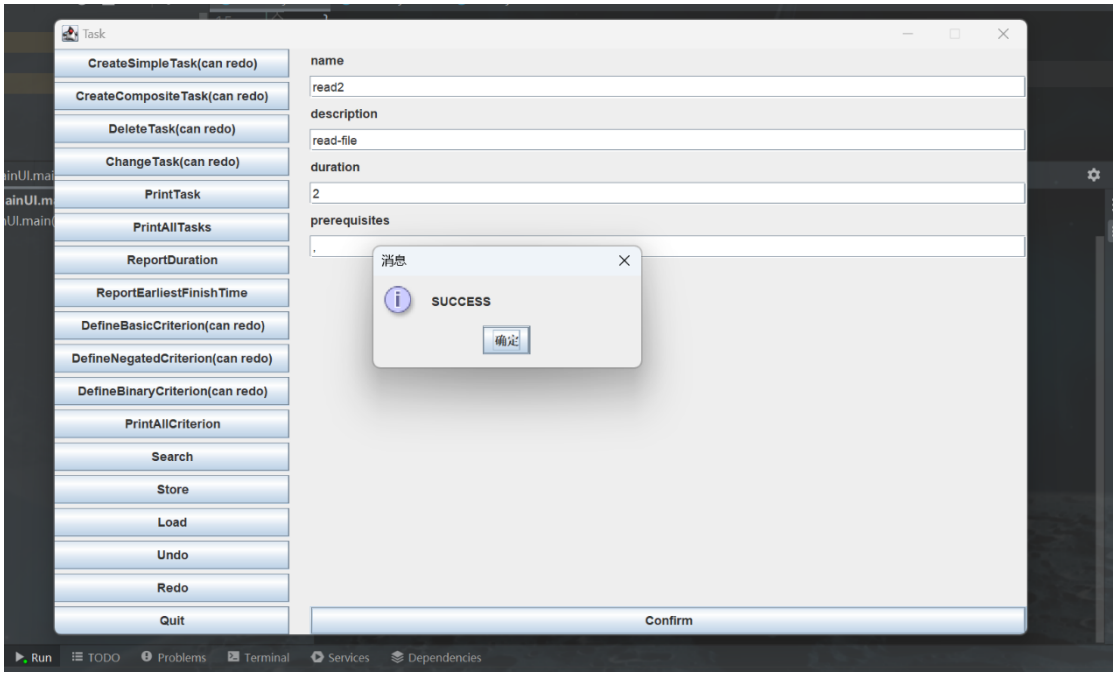


6.After confirming, you successfully redo.

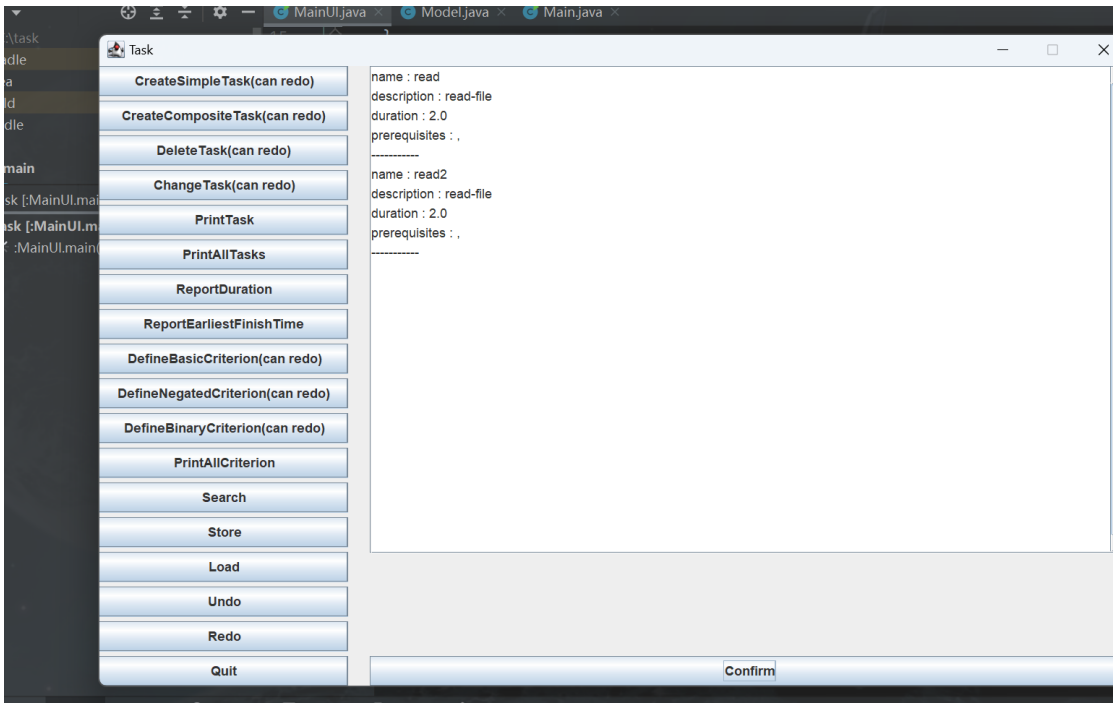


Undo instruction:

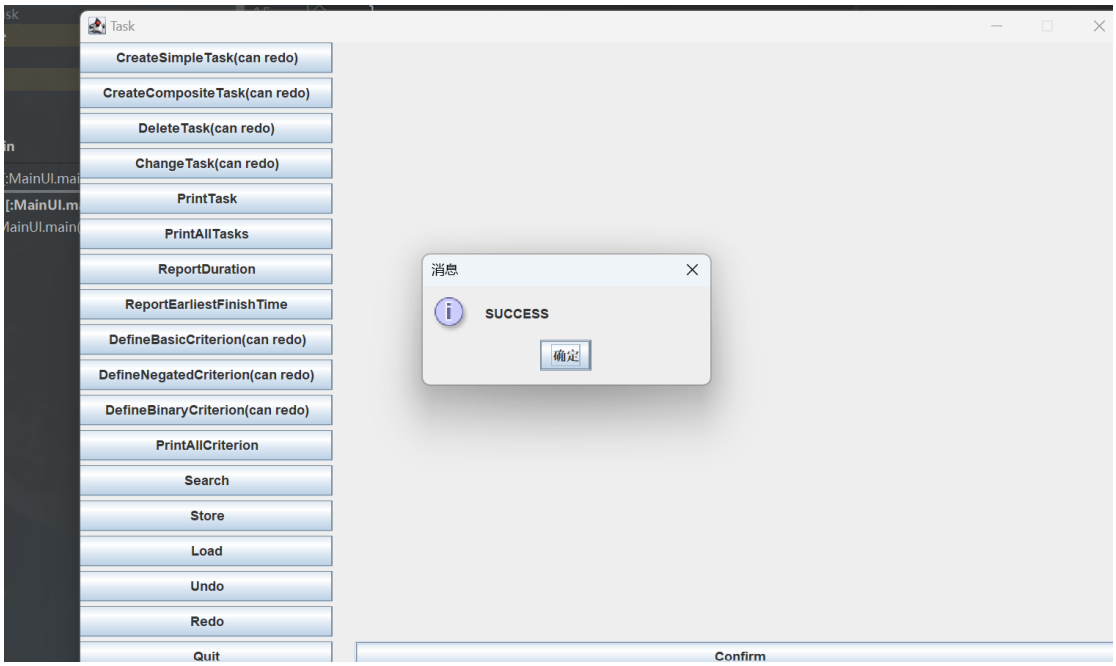
1.Follow the instructions from gui page.



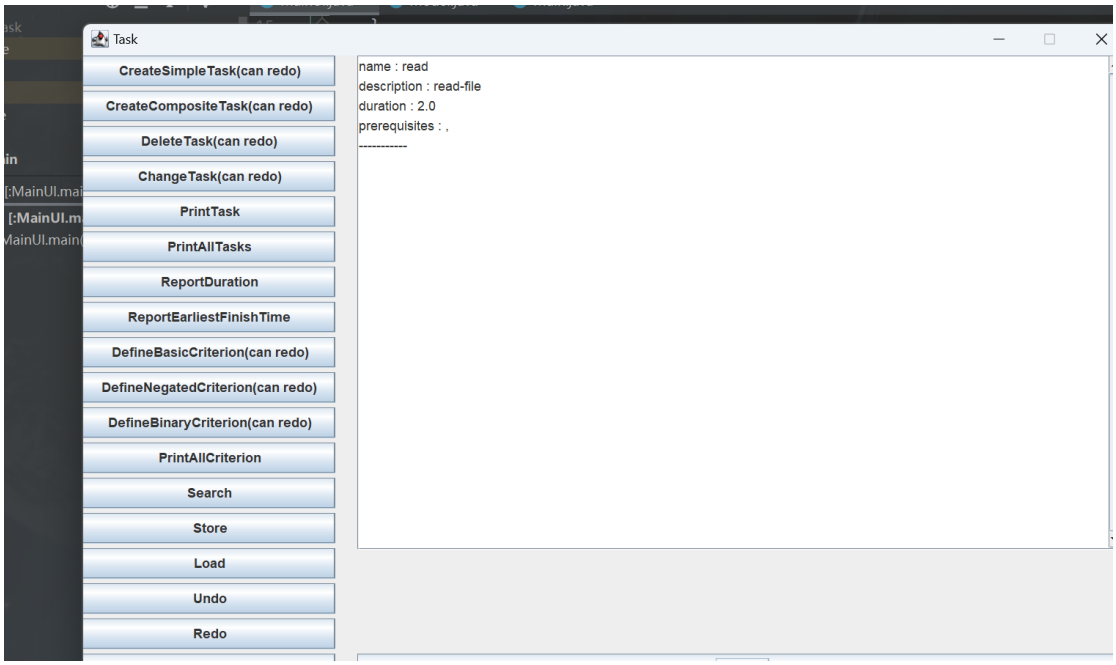
2. Confirm, and to check the penal status.



3.click the undo



4.After confirming and check to status, you have already undo successfully.



Below are the commands that I write (feel free for users to try):

CreateSimpleTask read_file read_file 1 ,
CreateSimpleTask readfiles read_file 1 ,
CreateSimpleTask readfile read_file 1 ,
CreateSimpleTask readfile read-file 1a ,
CreateSimpleTask readfile read-file 1 ,
CreateSimpleTask readfile read-file 1 ,

```
CreateSimpleTask read read-file 1 ,
CreateSimpleTask process process-file 2 ,
CreateSimpleTask save save-file 4 ,
CreateSimpleTask list list-file 1 ,
CreateSimpleTask print print-file 2 ,

CreateCompositeTask read read_file ,
CreateCompositeTask read_file read_file ,
CreateCompositeTask read-file read_file ,
CreateCompositeTask readfiles read_file ,
CreateCompositeTask cfile read-file ,
CreateCompositeTask cfile read-file read
CreateCompositeTask cfile read-file read, process, save

ChangeTask cfile name cread

PrintAllTasks
DeleteTask cread
PrintAllTasks

CreateCompositeTask cread read-task read,process,save
CreateCompositeTask cprint print-task list,print

PrintTask cread

PrintAllTasks

CreateSimpleTask read read-file 1 ,
CreateSimpleTask process process-file 2 ,
CreateSimpleTask save save-file 4 ,
CreateSimpleTask list list-file 1 ,
CreateSimpleTask print print-file 2 ,

CreateSimpleTask t1 read-file 1 ,
CreateSimpleTask t2 process-file 2 ,
CreateSimpleTask t3 save-file 2 t1,t2
CreateCompositeTask t4 read-task t1,t2,t3
ReportDuration t4
ReportEarliestFinishTime save
ReportEarliestFinishTime print
ReportEarliestFinishTime t3

DefineBasicCriterion c1 duration > 2
DefineBasicCriterion c2 duration < 2
DefineBasicCriterion c3 duration == 2
DefineBasicCriterion c4 name contains t
```

PrintAllCriterion

Search c1
Search c2
Search c3

DefineNegatedCriterion n1 c1
DefineNegatedCriterion n2 c2
Search n1
Search n2

DefineBinaryCriterion b1 c1 && c2
DefineBinaryCriterion b2 c3 && c4
search b1
search b2

Below is the screen shot of functions display:

```
:
CreateSimpleTask read_file read_file 1 ,
name only contains english letters and digits!
| ----------< MENU >-------
| CreateSimpleTask(can redo)
| CreateCompositeTask(can redo)
| DeleteTask(can redo)
| ChangeTask(can redo)
| PrintTask
| PrintAllTasks
| ReportDuration
| ReportEarliestFinishTime
| DefineBasicCriterion(can redo)
| DefineNegatedCriterion(can redo)
```

```
| -------------------------
:
CreateSimpleTask readfiles read_file 1 ,
name too length, only 8 characters!
| ----------< MENU >-------
| CreateSimpleTask(can redo)
| CreateCompositeTask(can redo)
| DeleteTask(can redo)
| ChangeTask(can redo)
| PrintTask
| PrintAllTasks
| ReportDuration
| ReportEarliestFinishTime
| DefineBasicCriterion(can redo)
| DefineNegatedCriterion(can redo)
```

```
| Quit
| ------------------------
:
CreateSimpleTask readfile read_file 1 ,
description only contains english letters, digits and hyphen letter(-)!
| ----------< MENU >-------
| CreateSimpleTask(can redo)
| CreateCompositeTask(can redo)
| DeleteTask(can redo)
| ChangeTask(can redo)
```

```
| Quit
| ------------------------
:
CreateSimpleTask readfile read-file 1a ,
Duration not a number!
| ----------< MENU >-------
| CreateSimpleTask(can redo)
| CreateCompositeTask(can redo)
| DeleteTask(can redo)
| ChangeTask(can redo)
```

```
16            option = InputUtils.read("Redo " + redoCommand + "! Input Message")
17            redo = false;
18            redoCommand = "";
19        } else {
20            if (help) {
```

```
| Redo
| Quit
| ------------------------
:
CreateSimpleTask readfile read-file 1 ,
SUCCESS
| ----------< MENU >-------
| CreateSimpleTask(can redo)
| CreateCompositeTask(can redo)
| DeleteTask(can redo)
| ChangeTask(can redo)
```

```
18            redoCommand = "";
19        } else {
20            if (help) {
```

```
| ------------------------
:

CreateSimpleTask readfile read-file 1 ,
Input invalid!
name should be unique!
| ----------< MENU >-------
| CreateSimpleTask(can redo)
| CreateCompositeTask(can redo)
| DeleteTask(can redo)
| ChangeTask(can redo)
```

```
| -------------------------
:
CreateSimpleTask read read-file 1 ,
SUCCESS
| ----------< MENU >-------
| CreateSimpleTask(can redo)
| CreateCompositeTask(can redo)
| DeleteTask(can redo)
| ChangeTask(can redo)
```

```
| -------------------------
:
CreateSimpleTask process process-file 2 ,
SUCCESS
| ----------< MENU >-------
| CreateSimpleTask(can redo)
| CreateCompositeTask(can redo)
| DeleteTask(can redo)
| ChangeTask(can redo)
```

```
| -------------------------
:
CreateSimpleTask save save-file 4 ,
SUCCESS
| ----------< MENU >-------
| CreateSimpleTask(can redo)
| CreateCompositeTask(can redo)
| DeleteTask(can redo)
| ChangeTask(can redo)
```

```
| Quit
| -------------------------
:
CreateSimpleTask list list-file 1 ,
SUCCESS
| ----------< MENU >-------
| CreateSimpleTask(can redo)
| CreateCompositeTask(can redo)
| DeleteTask(can redo)
| ChangeTask(can redo)
```

```
Main
MainUI
resources
17                redo = false;
18                redoCommand = "";
19            } else {
20                if (help) {
[:Main.main()]
[:Main.main()]: Run tasks...        2 min, 14 sec
Main.main()                         2 min, 14 sec
:
CreateSimpleTask print print-file 2 ,
SUCCESS
| ----------< MENU >-------
| CreateSimpleTask(can redo)
| CreateCompositeTask(can redo)
| DeleteTask(can redo)
| ChangeTask(can redo)
| PrintTask
| PrintAllTasks
| ReportDuration
Run  TODO  Problems  Terminal  Services  Build  Dependencies
```

```
| Redo
| Quit
| -------------------------
:
CreateCompositeTask read read_file ,
name should be unique!
| ----------< MENU >-------
| CreateSimpleTask(can redo)
```

```
4 min, 13 sec
| Quit
| -------------------------
:
CreateCompositeTask read_file read_file ,
name only contains english letters and digits!
| ----------< MENU >-------
| CreateSimpleTask(can redo)
| CreateCompositeTask(can redo)
| DeleteTask(can redo)
| ChangeTask(can redo)
```

```
| -------------------------
:
CreateCompositeTask read-file read_file ,
name only contains english letters and digits!
| ----------< MENU >-------
| CreateSimpleTask(can redo)
| CreateCompositeTask(can redo)
| DeleteTask(can redo)
| ChangeTask(can redo)
```

```
Main
MainUI
resources
test
task [:Main.main()]
task [:Main.main()]: Run tasks...                5 min, 36 sec
:Main.main()                                     5 min, 35 sec
18                redoCommand = "";
19            } else {
20                if (help) {
:
CreateCompositeTask readfiles read_file ,
name too length, only 8 characters!
| ----------< MENU >-------
| CreateSimpleTask(can redo)
| CreateCompositeTask(can redo)
| DeleteTask(can redo)
| ChangeTask(can redo)
| PrintTask
| PrintAllTasks
| ReportDuration
```

```
Main.main()]: Run tasks...                       6 min, 3 sec
in.main()                                        6 min, 3 sec
:
CreateCompositeTask cfile read-file ,
sub task's count must >= 2
| ----------< MENU >-------
| CreateSimpleTask(can redo)
| CreateCompositeTask(can redo)
| DeleteTask(can redo)
| ChangeTask(can redo)
| PrintTask
| PrintAllTasks
| ReportDuration
```

```
| ------------------------
:
CreateCompositeTask cfile read-file read
sub task's count must >= 2
| ----------< MENU >-------
| CreateSimpleTask(can redo)
| CreateCompositeTask(can redo)
| DeleteTask(can redo)
| ChangeTask(can redo)
TODO  Problems  Terminal  Services  Build  Dependencies
```

```
| ------------------------
:
CreateCompositeTask cfile read-file read,process,save
SUCCESS
| ----------< MENU >-------
| CreateSimpleTask(can redo)
| CreateCompositeTask(can redo)
| DeleteTask(can redo)
| ChangeTask(can redo)
```

```
sk [:Main.main()]: Run tasks...                  7 min, 25 sec
:Main.main()                                     7 min, 25 sec
:
ChangeTask cfile name cread
SUCCESS
| ----------< MENU >-------
| CreateSimpleTask(can redo)
| CreateCompositeTask(can redo)
| DeleteTask(can redo)
| ChangeTask(can redo)
| PrintTask
| PrintAllTasks
| ReportDuration
ntrol  Run  TODO  Problems  Terminal  Services  Build  Dependencies
```

```
:Main.main()                                  8 min, 22 sec
| -------------------------
:
PrintAllTasks
name : readfile
description : read-file
duration : 1.0
prerequisites : ,
--------
name : read
description : read-file
duration : 1.0
prerequisites : ,
--------
name : process
description : process-file
duration : 2.0
prerequisites : ,
--------
name : save
description : save-file
duration : 4.0
prerequisites : ,
```

```
duration : 4.0
prerequisites : ,
--------
name : list
description : list-file
duration : 1.0
prerequisites : ,
--------
name : print
description : print-file
duration : 2.0
prerequisites : ,
--------
name : cread
description : read-file
subtasks : read,process,save
--------
SUCCESS!
| ----------< MENU >-------
| CreateSimpleTask(can redo)
| CreateCompositeTask(can redo)
| DeleteTask(can redo)
| ChangeTask(can redo)
```

```
The duration of task t4 is : 4.0
null
| ----------< MENU >-------
```

```
| -------------------------
:
SUCCESS
| ----------< MENU >-------
```

```
| -------------------------
:
SUCCESS
| ----------< MENU >-------
```

```
task [:Main.main()]: Run tasks...
:Main.main()
name : save
description : save-file
duration : 4.0
prerequisites : ,
--------
name : list
description : list-file
duration : 1.0
prerequisites : ,
--------
name : print
description : print-file
duration : 2.0
prerequisites : ,
--------
SUCCESS!
| ----------< MENU >-------
| CreateSimpleTask
```

```
| -------------------------
:
CreateCompositeTask create add-task read process save
SUCCESS
| ----------< MENU >-------
```

```
task [:Main.main()]: Run tasks...
:Main.main()
| -------------------------
:
CreateCompositeTask create cprint task list print
SUCCESS
| ----------< MENU >-------
```

```
task [:Main.main()]: Run tasks...
:Main.main()
name : cprint
description : print-task
subtasks : list,print
--------
SUCCESS!
| ----------< MENU >-------
```

```
Run: task [:Main.main()]  TestModel
task [:Main.main()]: Run tasks...
:Main.main()
:
DefineBasicCriterion c1 duration 1.0
SUCCESS
```

```
task [:Main.main()]: Run tasks...
:Main.main()
:
ReportForTestFindTime t3
The duration of task t3 is 3.0
null
```

```
search 02
name : read
description : read-file
duration : 1.0
prerequisites : ,
-------
name : list
description : list-file
duration : 1.0
prerequisites : ,
-------
name : t1
description : read-file
duration : 1.0
prerequisites : ,
-------
name : t4
```

```
Run: task [:Main.main()]  TestModel
task [:Main.main()]: Run tasks...
:Main.main()
:
DefineBasicCriterion c1 add domains f
SUCCESS
```

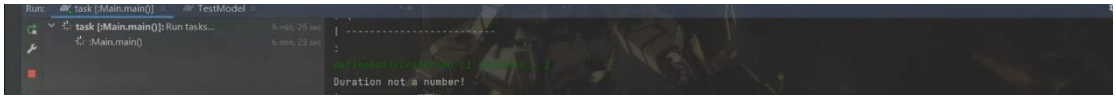
```
task [:Main.main()]: Run tasks...
:Main.main()
name : cread
description : read-task
subtasks : read,process,save
SUCCESS!
```

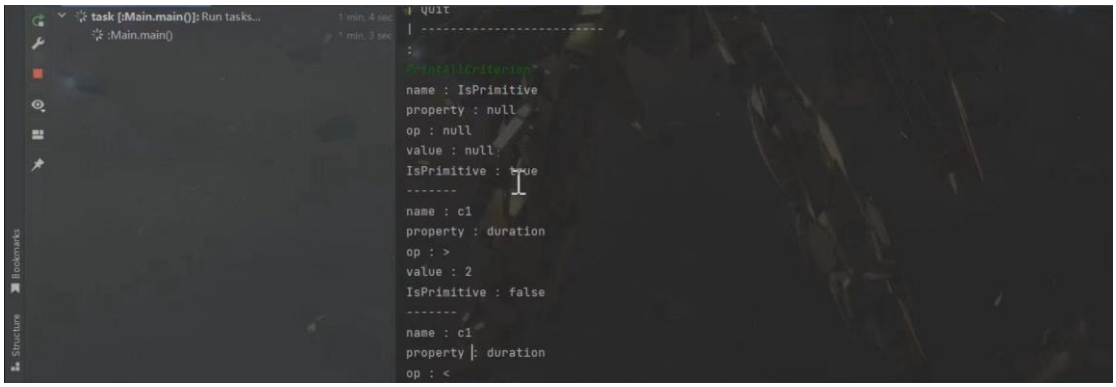
```
search c1
name : save
description : save-file
duration : 4.0
prerequisites : ,
-------
null
```

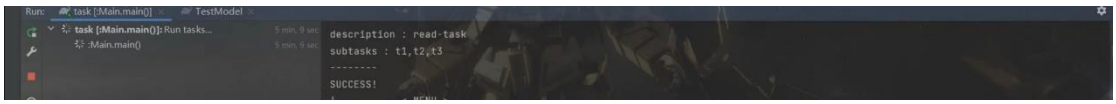

task [:Main.main()]: Run tasks...
:Main.main()
SUCCESS
| ----------< MENU >-------
| CreateSimpleTask


Run:
task [:Main.main()]
TestModel
task [:Main.main()]: Run tasks...
:Main.main()
SUCCESS


Run:
task [:Main.main()]
TestModel
task [:Main.main()]: Run tasks...
:Main.main()
The duration of task save is : 4.0
null


Run:
task [:Main.main()]
TestModel
task [:Main.main()]: Run tasks...
:Main.main()
| ----------
Duration not a number!

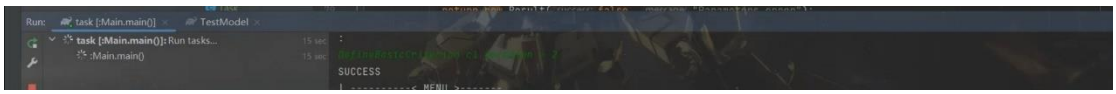

task [:Main.main()]: Run tasks...
:Main.main()
| Quit
| ----------
PrintAllCriterion
name : IsPrimitive
property : null
op : null
value : null
IsPrimitive : true
----
name : c1
property : duration
op : >
value : 2
IsPrimitive : false
----
name : c1
property : duration
op : <


Run:
task [:Main.main()]
TestModel
task [:Main.main()]: Run tasks...
:Main.main()
description : read-task
subtasks : t1,t2,t3

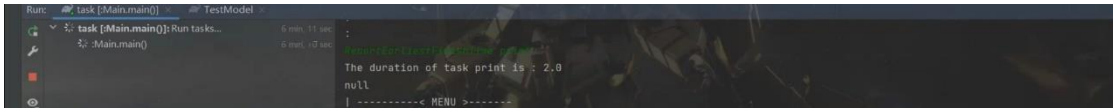
SUCCESS!

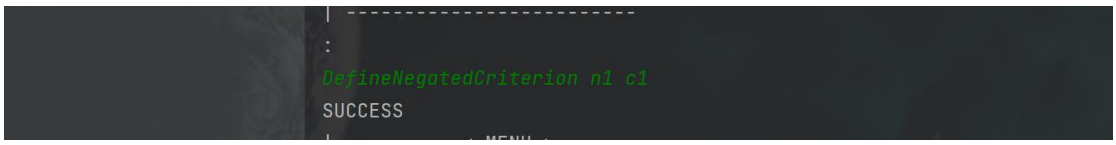

Search-cf
name : process
description : process-file
duration : 2.0
prerequisites : ,
----
name : print
description : print-file
duration : 2.0
prerequisites : ,
----
name : t2


task [:Main.main()]: Run tasks...
:Main.main()
| ----------
Failed, task t4 not exist!


Run:
task [:Main.main()]
TestModel
task [:Main.main()]: Run tasks...
:Main.main()
SUCCESS


model
bean
import ...
public class Main {
Run:
task [:Main.main()]
TestModel
task [:Main.main()]: Run tasks...
:Main.main()
| ----------
SUCCESS
| ----------< MENU >-------





-The END of the User Manual-