User Manual

COMP2021 Project Fall 2023

Group 24

Liu Luyang

Wang Ruijie

Zhu Jin Shun

Zeng Tianyi

Introduction

This project system from group 24 is a command-line-based task management system (TMS) that will enable its users to define, query, and modify the primitive tasks they need to complete.

With TMS, users can create two types of tasks: primitive tasks and composite tasks. Primitive tasks have basic properties such as name, description, duration, and prerequisites. Composite tasks are made up of subtasks and can be used to represent more complex work structures. Users can modify task properties, delete tasks if they are not prerequisites for other tasks, and print task information.

TMS allows users to define criteria for task selection. They can create basic criteria based on properties like name, description, duration, or prerequisites. Advanced composite criteria can be constructed by combining or negating existing criteria using logical operators. Users can print all defined criteria for reference.

The system provides search functionality to filter tasks based on criteria. Users can search for tasks that satisfy a specific criterion and get a list of matching tasks. TMS also supports reporting the duration and earliest finish time for tasks, taking into account prerequisites and subtasks.

TMS offers features for storing tasks and criteria in files, allowing users to save their work and reload it later. Additionally, users can undo and redo most actions, enabling them to revert or replay changes made to tasks and criteria.

Overall, to users, the Task Management System aims to simplify task management by providing a command-line interface with intuitive commands for creating, modifying, searching, and organizing tasks based on criteria. It offers flexibility, efficiency, and the ability to track task progress effectively.

To start the system, run the application.java file!!!

Command:

Some basic rules before using commands on creating tasks:

A task name 1) may contain only English letters and digits, 2) cannot start with digits, and 3) may contain at most eight characters. A description may contain English letters, digits, and the hyphen letter (-).

A duration is a positive real number, and it gives the minimum amount of time in hours required to complete the task.

The prerequisites and subtasks are comma-separated list of task names.

REQ1: CreatePrimitiveTask

Description:

The functionalities for this command: Create a new primitive task.

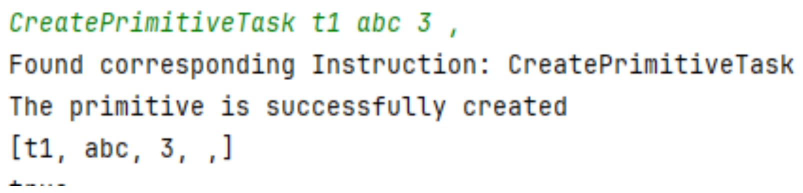
It can be used by calling the command: CreatePrimitiveTask name description duration.

Example: CreatePrimitiveTask task 1 boil-water 0.3

Step by step Instructions:

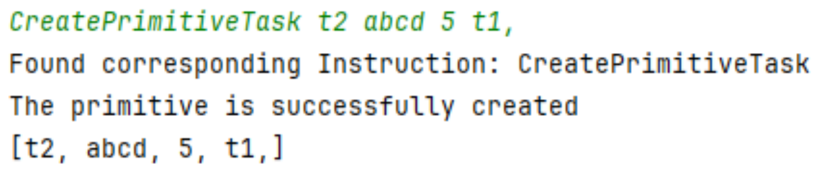
Everything starts from creating a primitive task, so if you would like to create a primitive task to start, you would need to check whether the value are appropriate, the task name should not be repeated as it is unique, also the values for each property should follow the basic rules provided above, if not, an error will occur. Once all conditions met, you can choose to use the (CreatePrimitiveTask Taskname description duration value) command to create a new primitive task.

Some examples of creating primitive task



```
CreatePrimitiveTask t1 abc 3 ,
Found corresponding Instruction: CreatePrimitiveTask
The primitive is successfully created
[t1, abc, 3, ,]
```

Create a primitive task whose name is t1, description is abc, duration is 3, and without prerequisite task



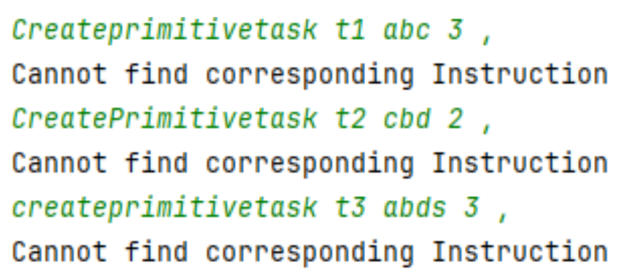
```
CreatePrimitiveTask t2 abcd 5 t1,
Found corresponding Instruction: CreatePrimitiveTask
The primitive is successfully created
[t2, abcd, 5, t1,]
```

Create a primitive task whose name is t2, description is abcd, duration equals to 5 and prerequisite is t1

Troubleshooting:

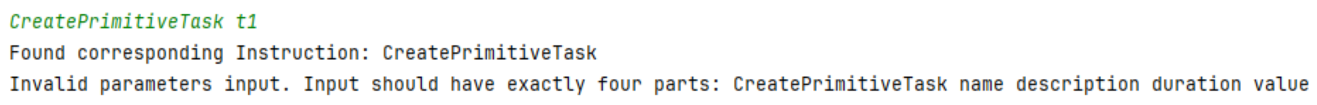
Some errors occur when you don't have the correct command and capital letters for the command. For this command, the C, P and T should all be capitalized for successfully execute. At the same time, remember to execute the command with an no repeated task's name or it will return an error. At the same time, please remember the value assigned to each property should also be valid or it will also return an error.

Some mistake errors when having wrong commands.

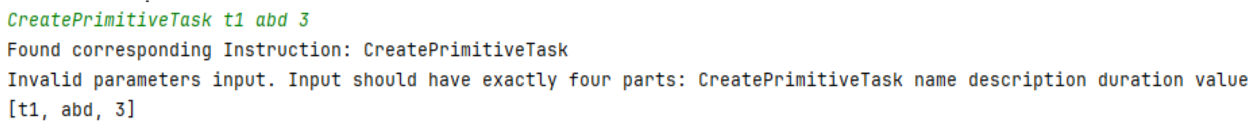


```
Createprimitivetask t1 abc 3 ,
Cannot find corresponding Instruction
CreatePrimitivetask t2 cbd 2 ,
Cannot find corresponding Instruction
createprimitivetask t3 abds 3 ,
Cannot find corresponding Instruction
```

Without C&&P&&T capitalized when writing command

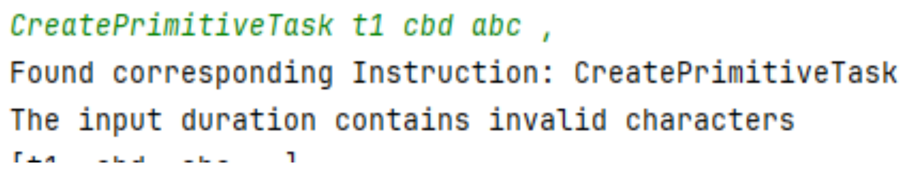


```
CreatePrimitiveTask t1
Found corresponding Instruction: CreatePrimitiveTask
Invalid parameters input. Input should have exactly four parts: CreatePrimitiveTask name description duration value
```

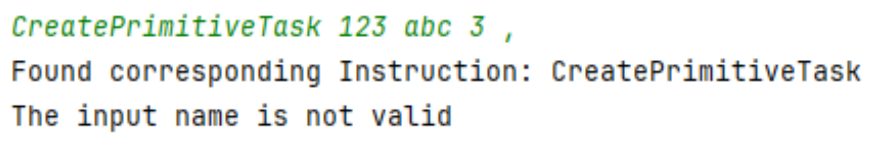


```
CreatePrimitiveTask t1 abd 3
Found corresponding Instruction: CreatePrimitiveTask
Invalid parameters input. Input should have exactly four parts: CreatePrimitiveTask name description duration value
[t1, abd, 3]
```

Error when input is not equal to four parts (Taskname, description, duration, value)

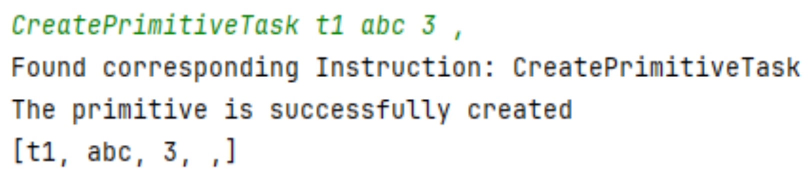


```
CreatePrimitiveTask t1 cbd abc ,
Found corresponding Instruction: CreatePrimitiveTask
The input duration contains invalid characters
```

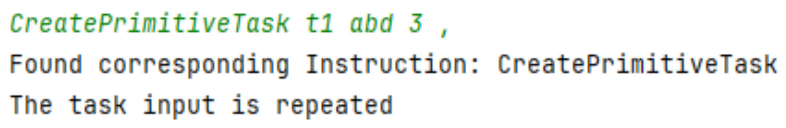


```
CreatePrimitiveTask 123 abc 3 ,
Found corresponding Instruction: CreatePrimitiveTask
The input name is not valid
```

Error when value input is not valid.



```
CreatePrimitiveTask t1 abc 3 ,
Found corresponding Instruction: CreatePrimitiveTask
The primitive is successfully created
[t1, abc, 3, ,]
```



```
CreatePrimitiveTask t1 abd 3 ,
Found corresponding Instruction: CreatePrimitiveTask
The task input is repeated
```

Error when task name(repeated) already exist.

REQ2: CreateCompositeTask

Description:

The functionalities for this command: Create a composite task

It can be used by calling the command: CreateCompositeTask name0 description name1, name2,...,namek

Example:

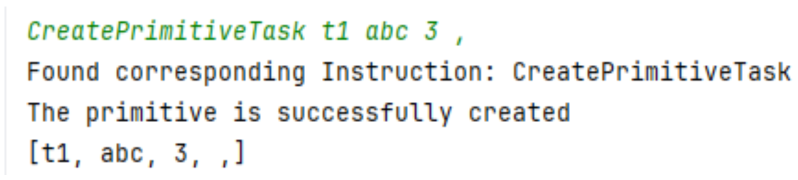
CreateCompositeTask composite1 make-coffee task1, task2

Notes: The new task is named name0 and has k subtasks, named name1, name2, ..., namek ($k \geq 2$), respectively.

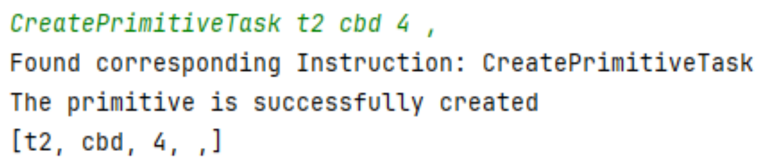
Step by step Instructions:

Before creating a composite criterion, remember to create at least two tasks to continue executing the command. When at least two valid tasks are created, you can execute the (CreateCompositeTask Taskname description existingtaskname1, existingtaskname2…existingtasknamek) command to create a new primitive task.

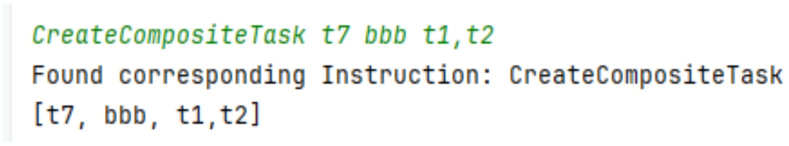
Some examples of creating composite task



```
CreatePrimitiveTask t1 abc 3 ,
Found corresponding Instruction: CreatePrimitiveTask
The primitive is successfully created
[t1, abc, 3, ,]
```



```
CreatePrimitiveTask t2 cbd 4 ,
Found corresponding Instruction: CreatePrimitiveTask
The primitive is successfully created
[t2, cbd, 4, ,]
```



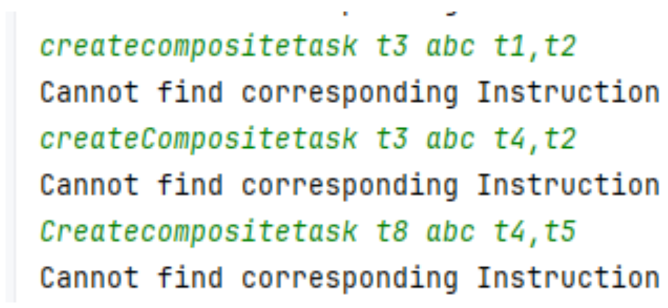
```
CreateCompositeTask t7 bbb t1,t2
Found corresponding Instruction: CreateCompositeTask
[t7, bbb, t1,t2]
```

Example of creating a composite task named t7 with description “bbb” and subtasks t1,t2

Troubleshooting:

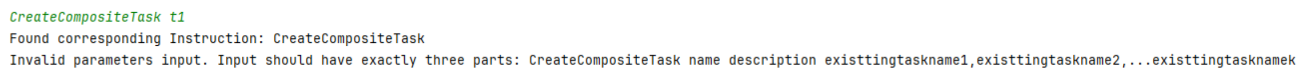
Some errors occur when you don’t have the correct command and capital letters for the command. For this command, the C, C and T should all be capitalized for successfully execute. At the same time, remember to execute the command with an no repeated task’s name or it will return an error. At the same time, please remember that the tasks assigned to create a composite task should also be valid and exist or it will also return an error.

Some mistake errors when having wrong commands.

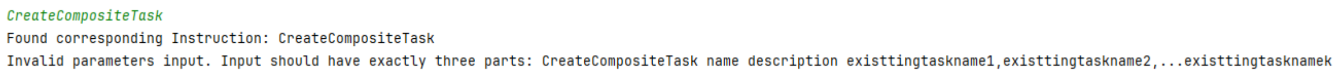


```
createcompositetask t3 abc t1,t2
Cannot find corresponding Instruction
createCompositetask t3 abc t4,t2
Cannot find corresponding Instruction
Createcompositetask t8 abc t4,t5
Cannot find corresponding Instruction
```

Without C&&C&&T capitalized when writing command

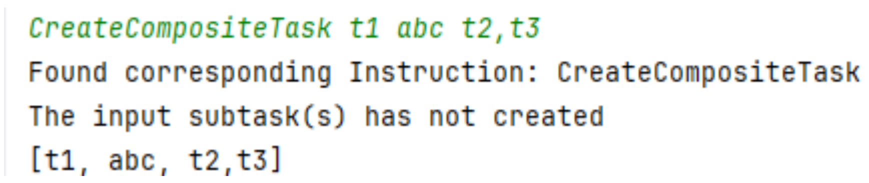


```
CreateCompositeTask t1
Found corresponding Instruction: CreateCompositeTask
Invalid parameters input. Input should have exactly three parts: CreateCompositeTask name description existtingtaskname1,existtingtaskname2,...existtingtaskname k
```



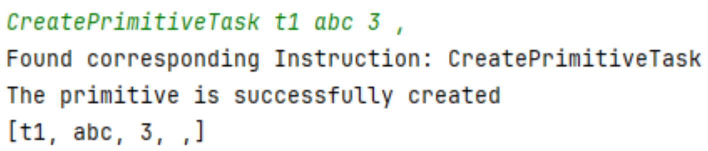
```
CreateCompositeTask
Found corresponding Instruction: CreateCompositeTask
Invalid parameters input. Input should have exactly three parts: CreateCompositeTask name description existtingtaskname1,existtingtaskname2,...existtingtaskname k
```

Error when input is not equal to three parts (Taskname, description, existingtaskname1, existingtaskname2…existingtasknamek)

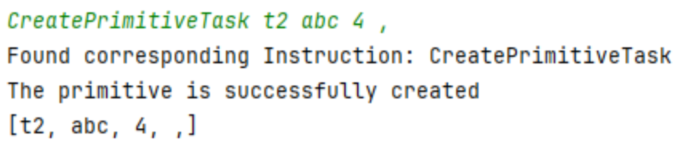


```
CreateCompositeTask t1 abc t2,t3
Found corresponding Instruction: CreateCompositeTask
The input subtask(s) has not created
[t1, abc, t2,t3]
```

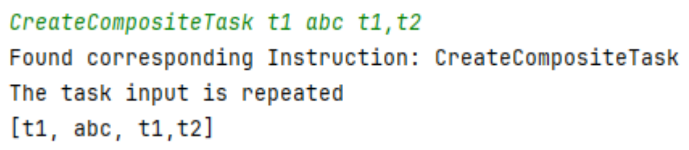
Error when primitive task required to composite doesn't exist.



```
CreatePrimitiveTask t1 abc 3 ,
Found corresponding Instruction: CreatePrimitiveTask
The primitive is successfully created
[t1, abc, 3, ,]
```



```
CreatePrimitiveTask t2 abc 4 ,
Found corresponding Instruction: CreatePrimitiveTask
The primitive is successfully created
[t2, abc, 4, ,]
```



```
CreateCompositeTask t1 abc t1,t2
Found corresponding Instruction: CreateCompositeTask
The task input is repeated
[t1, abc, t1,t2]
```

Error when task name entered are repeated.

REQ3:DeleteTask

Description:

The functionalities for this command: Delete an existing task.

It can be used by calling the command: DeleteTask name.

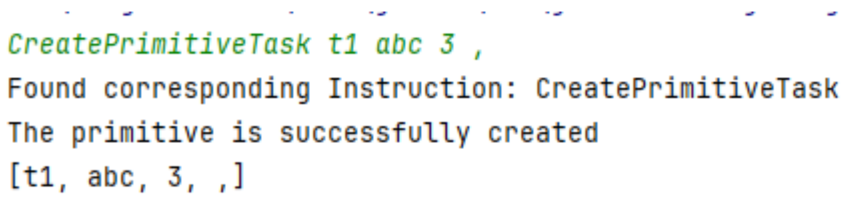
Example: DeleteTask task1

Note: A primitive task can be deleted if it is not the prerequisite of any other tasks; A composite task can be deleted if it does not contain any sub-task that is the prerequisite of another task. When a composite task is deleted, all the sub-tasks it contains are deleted too.

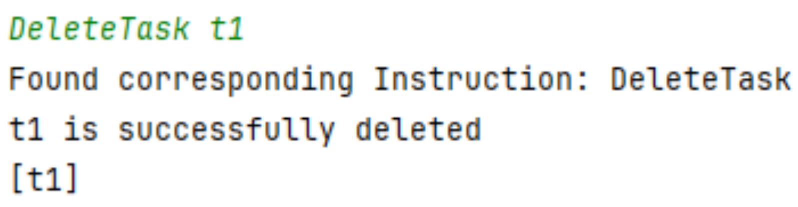
Step by step Instructions:

Before using this command, make sure that at least one of the tasks you would like to delete is created, if you delete a non-existing task, it will return an error. Once you have created a primitive or composite task, but you don't want to use it anymore, you can use the DeleteTask taskname command to delete the task (the deletion follows the note rules above). Then the task would be deleted if the conditions meet.

Some examples of deleting a task

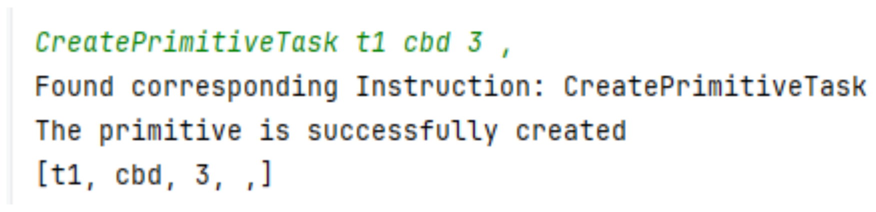


```
CreatePrimitiveTask t1 abc 3 ,
Found corresponding Instruction: CreatePrimitiveTask
The primitive is successfully created
[t1, abc, 3, ,]
```

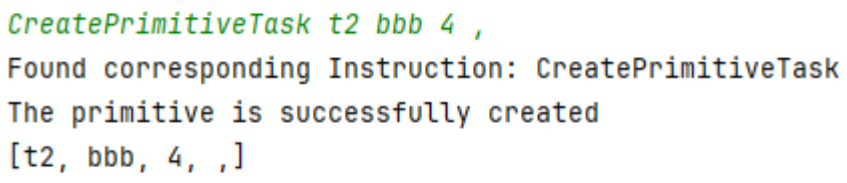


```
DeleteTask t1
Found corresponding Instruction: DeleteTask
t1 is successfully deleted
[t1]
```

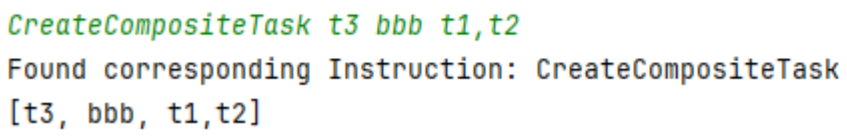
Deleted the primitive task t1.



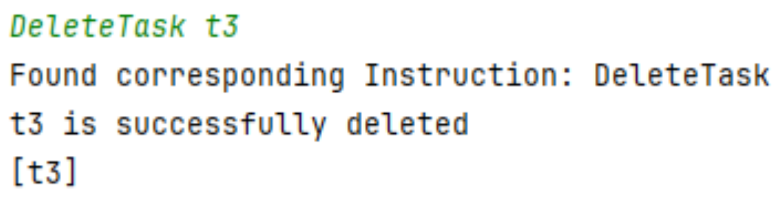
```
CreatePrimitiveTask t1 cbd 3 ,
Found corresponding Instruction: CreatePrimitiveTask
The primitive is successfully created
[t1, cbd, 3, ,]
```



```
CreatePrimitiveTask t2 bbb 4 ,
Found corresponding Instruction: CreatePrimitiveTask
The primitive is successfully created
[t2, bbb, 4, ,]
```



```
CreateCompositeTask t3 bbb t1,t2
Found corresponding Instruction: CreateCompositeTask
[t3, bbb, t1,t2]
```



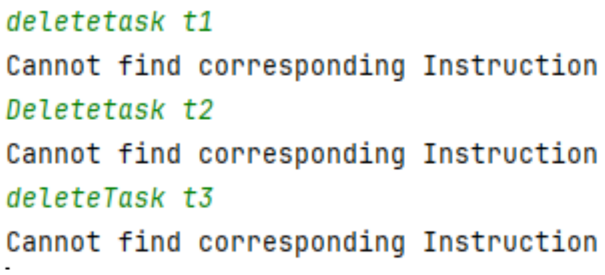
```
DeleteTask t3
Found corresponding Instruction: DeleteTask
t3 is successfully deleted
[t3]
```

Deleted both the composite task t3 and the primitive task t1, t2.

Troubleshooting:

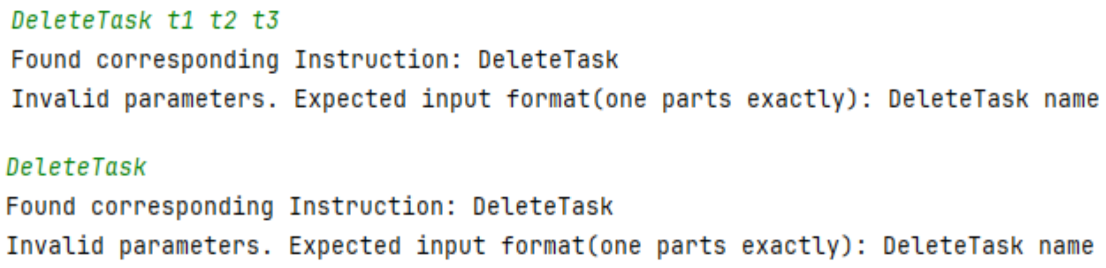
Some errors occur when you don't have the correct command and capital letters for the command. For this command, the D and T should all be capitalized for successfully execute. At the same time, remember to execute the command with an existing task's name or it will return an error. At the same time, or it will also return an error.

Some mistake errors when having wrong commands.



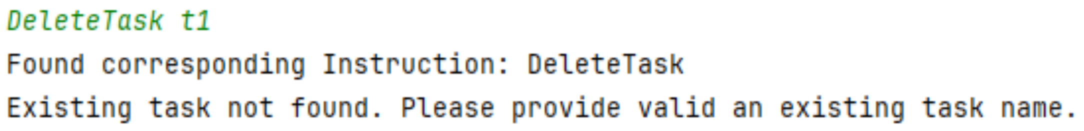
```
deletetask t1
Cannot find corresponding Instruction
Deletetask t2
Cannot find corresponding Instruction
deleteTask t3
Cannot find corresponding Instruction
```

Without D&&T capitalized when writing command.



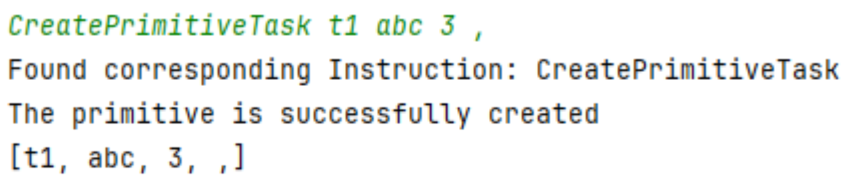
```
DeleteTask t1 t2 t3
Found corresponding Instruction: DeleteTask
Invalid parameters. Expected input format(one parts exactly): DeleteTask name

DeleteTask
Found corresponding Instruction: DeleteTask
Invalid parameters. Expected input format(one parts exactly): DeleteTask name
```

Error when input is not equal to one part (Taskname).

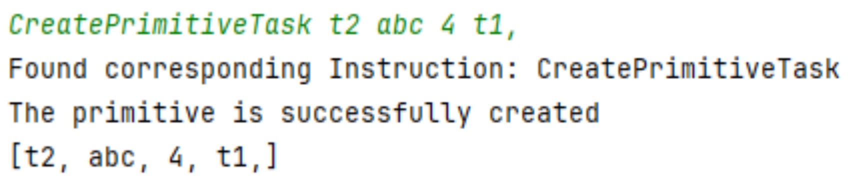


```
DeleteTask t1
Found corresponding Instruction: DeleteTask
Existing task not found. Please provide valid an existing task name.
```

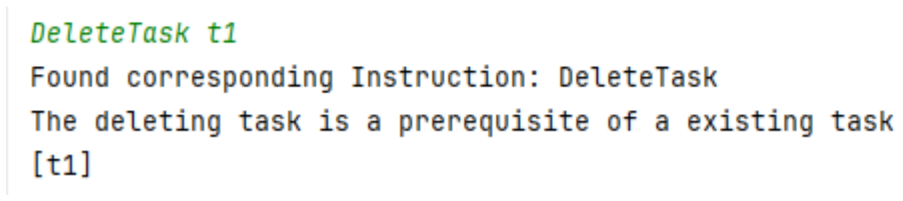
Error when task required to delete does not exist.



```
CreatePrimitiveTask t1 abc 3 ,
Found corresponding Instruction: CreatePrimitiveTask
The primitive is successfully created
[t1, abc, 3, ,]
```



```
CreatePrimitiveTask t2 abc 4 t1,
Found corresponding Instruction: CreatePrimitiveTask
The primitive is successfully created
[t2, abc, 4, t1,]
```



```
DeleteTask t1
Found corresponding Instruction: DeleteTask
The deleting task is a prerequisite of a existing task
[t1]
```

Error when task required to delete doesn’t meet the requirements mentioned in notes.

REQ4: ChangeTask

Description:

The functionalities for this command: Set the property of a specific task to the new value

It can be used by calling the command: ChangeTask name property newValue

Example: ChangeTask task1 duration 0.5

Notes:

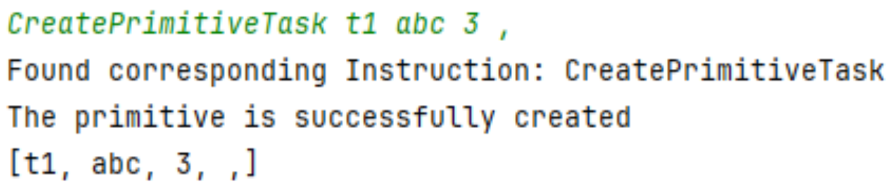
(1) If the task is primitive, property can be name, description, duration, or prerequisites; If the task is composite, property can be name, description, or subtasks. The newValue may take any valid value compatible with the corresponding property. (2) The prerequisites of

a task are not affected by changes to the other properties of the task

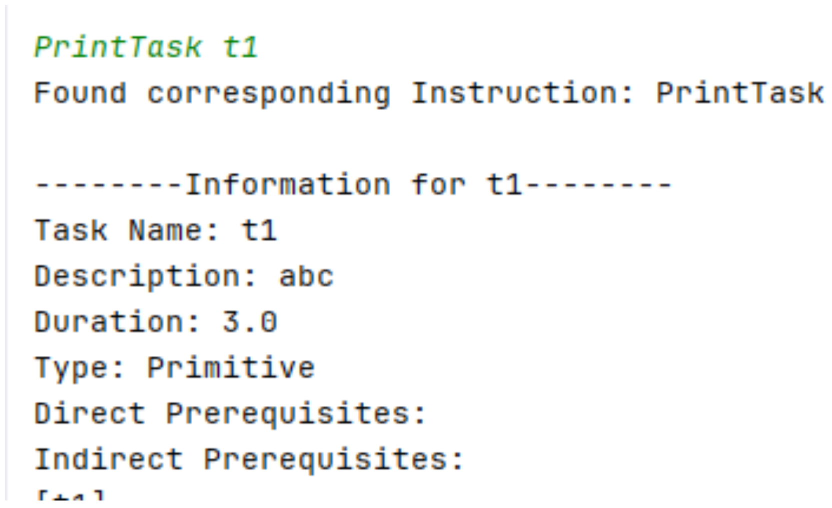
Step by step Instructions:

Once you have successfully created any primitive or composite tasks, and you would like to change a certain information for the task created, you can choose to use the ChangeTask Taskname property value command to change the information for one of the tasks you have created.

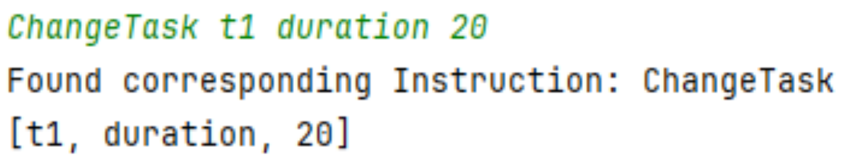
Some examples of changing task



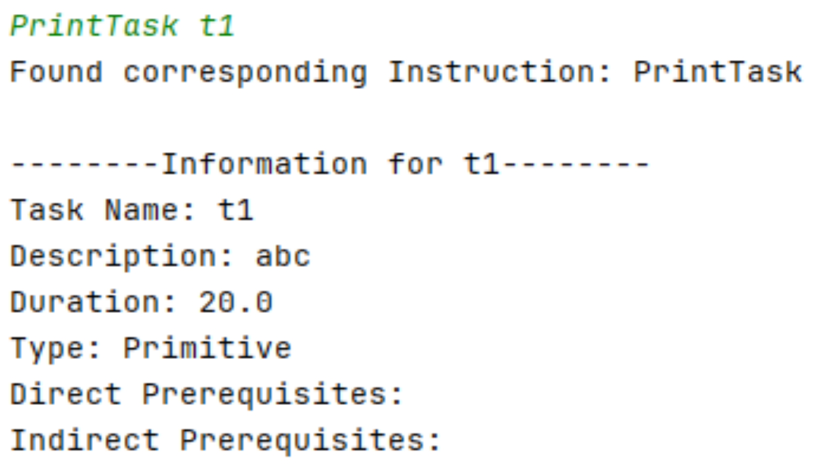
```
CreatePrimitiveTask t1 abc 3 ,
Found corresponding Instruction: CreatePrimitiveTask
The primitive is successfully created
[t1, abc, 3, ,]
```



```
PrintTask t1
Found corresponding Instruction: PrintTask

--------Information for t1--------
Task Name: t1
Description: abc
Duration: 3.0
Type: Primitive
Direct Prerequisites:
Indirect Prerequisites:
```



```
ChangeTask t1 duration 20
Found corresponding Instruction: ChangeTask
[t1, duration, 20]
```



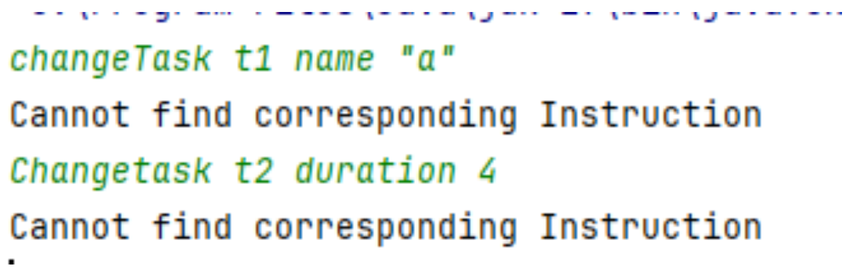
```
PrintTask t1
Found corresponding Instruction: PrintTask

--------Information for t1--------
Task Name: t1
Description: abc
Duration: 20.0
Type: Primitive
Direct Prerequisites:
Indirect Prerequisites:
```

Changed task t1's duration from 3 to 20.

Troubleshooting:

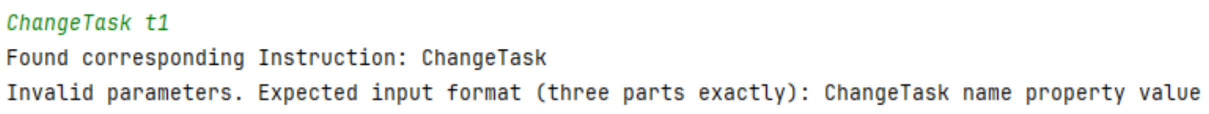
Some errors occur when you don't have the correct command and capital letters for the command. For this command, the C and T should all be capitalized for successfully execute. At the same time, remember to execute the command with an existing task's name or it will return an error. At the same time, please remember to the change value assigned should also be valid or it will also return an error.

Some mistake errors when having wrong commands.



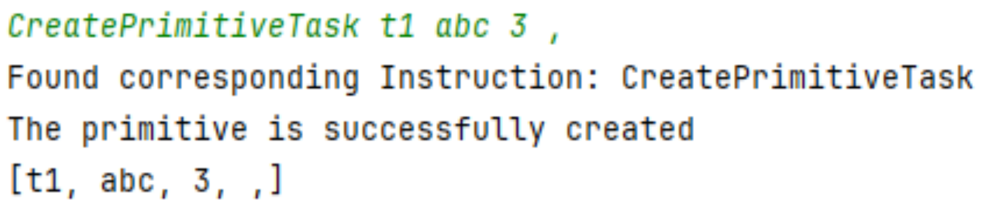
```
changeTask t1 name "a"
Cannot find corresponding Instruction
Changetask t2 duration 4
Cannot find corresponding Instruction
```

Without C&&T capitalized when writing command

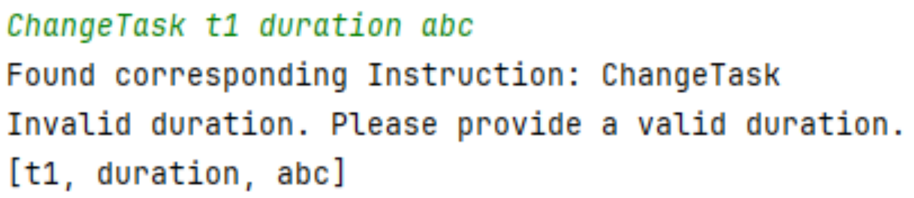


```
ChangeTask t1
Found corresponding Instruction: ChangeTask
Invalid parameters. Expected input format (three parts exactly): ChangeTask name property value
```

Error when input is not equal to three parts (Taskname, property, value)

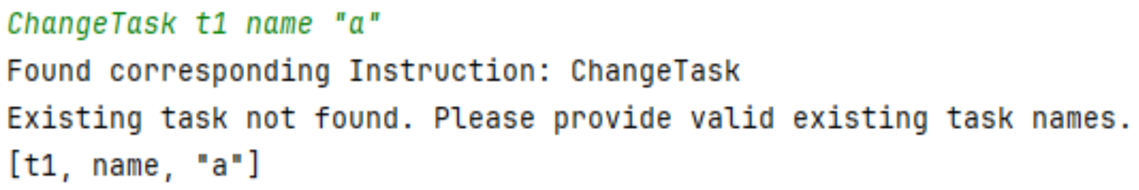


```
CreatePrimitiveTask t1 abc 3 ,
Found corresponding Instruction: CreatePrimitiveTask
The primitive is successfully created
[t1, abc, 3, ,]
```



```
ChangeTask t1 duration abc
Found corresponding Instruction: ChangeTask
Invalid duration. Please provide a valid duration.
[t1, duration, abc]
```

Error when value required to changed input is not equal is not valid.



```
ChangeTask t1 name "a"
Found corresponding Instruction: ChangeTask
Existing task not found. Please provide valid existing task names.
[t1, name, "a"]
```

Error when task required to changed doesn't exist.

REQ5: PrintTask

Description:

The functionalities for this command: Print the information of a task.

It can be used by calling the command: PrintTask taskname

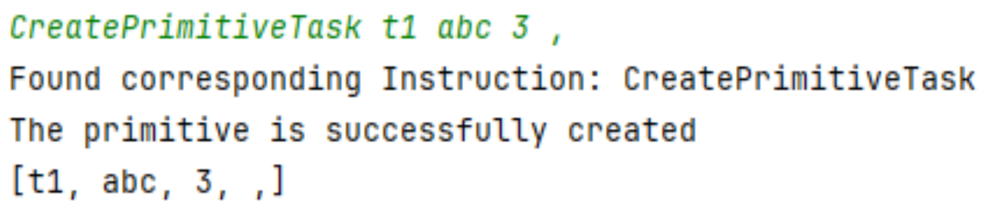
Example: PrintTask task1

Notes: Please be reminded that the task required to be printed should be an existing task. Create a primitive task or a composite task before you want to print it out.

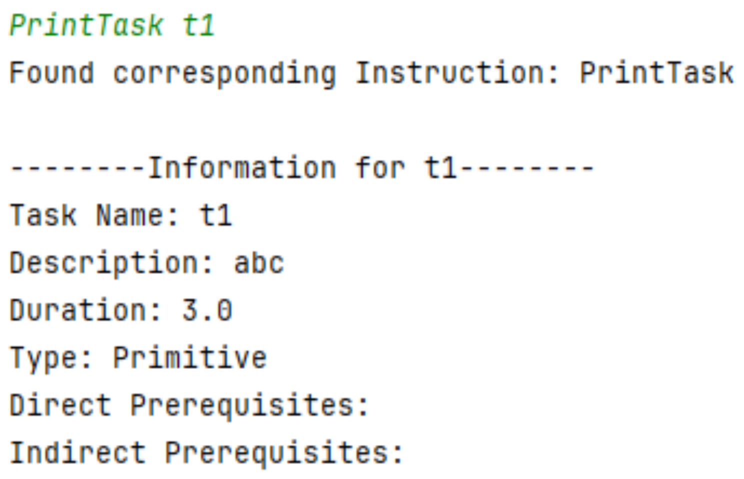
Step by step Instructions:

Once you have successfully created one primitive or composite tasks, and you would like to check what are the information for one certain task created, you can choose to use the PrintTask Taskname command to check the information for one of the tasks you have created.

Some examples of printing certain tasks

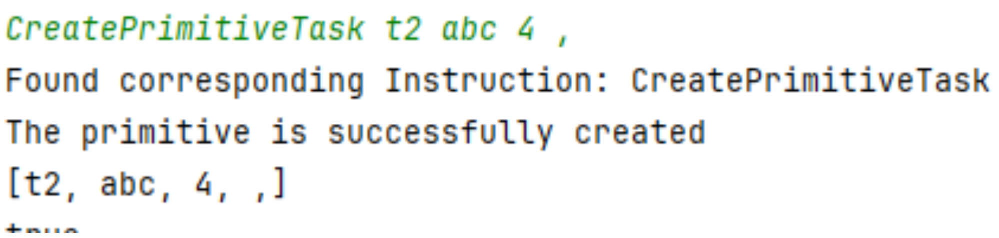


```
CreatePrimitiveTask t1 abc 3 ,
Found corresponding Instruction: CreatePrimitiveTask
The primitive is successfully created
[t1, abc, 3, ,]
```

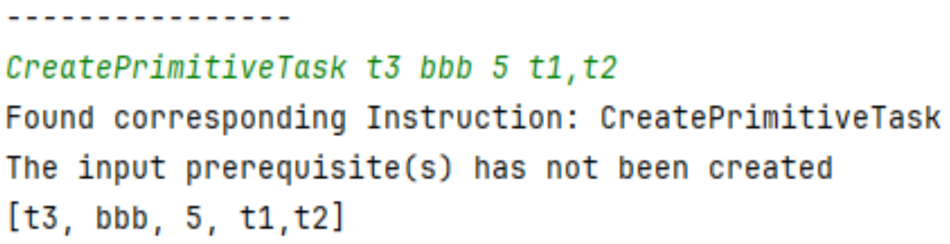


```
PrintTask t1
Found corresponding Instruction: PrintTask

--------Information for t1--------
Task Name: t1
Description: abc
Duration: 3.0
Type: Primitive
Direct Prerequisites:
Indirect Prerequisites:
```

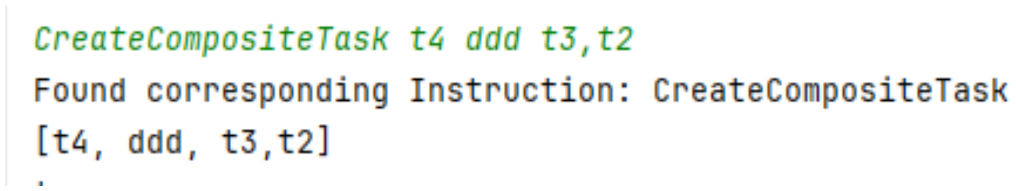
Create and print a primitive task with name t1



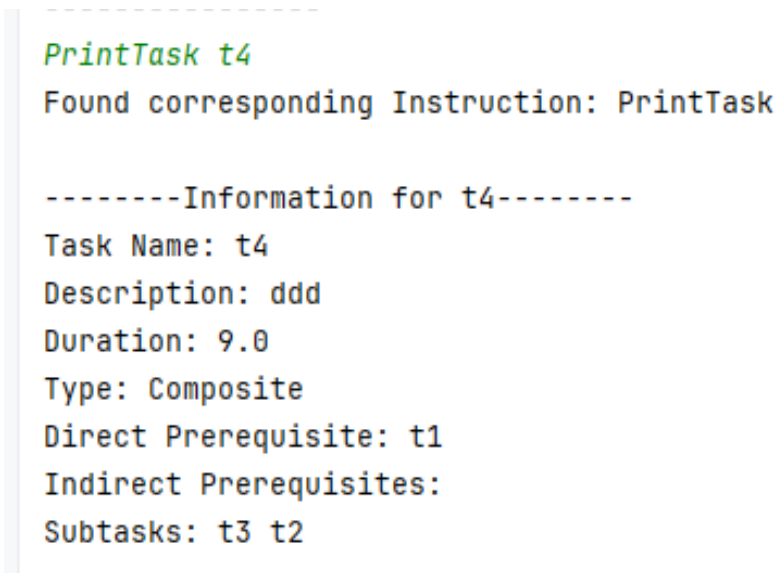
```
CreatePrimitiveTask t2 abc 4 ,
Found corresponding Instruction: CreatePrimitiveTask
The primitive is successfully created
[t2, abc, 4, ,]
```



```
----------------
CreatePrimitiveTask t3 bbb 5 t1,t2
Found corresponding Instruction: CreatePrimitiveTask
The input prerequisite(s) has not been created
[t3, bbb, 5, t1,t2]
```



```
CreateCompositeTask t4 ddd t3,t2
Found corresponding Instruction: CreateCompositeTask
[t4, ddd, t3,t2]
```



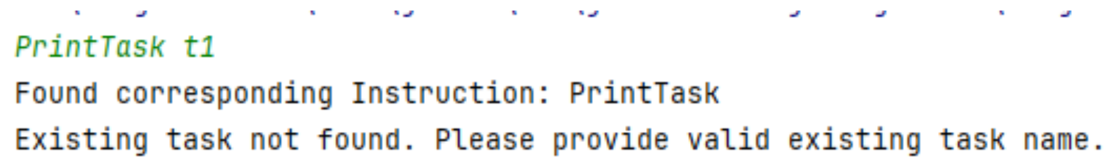
```
PrintTask t4
Found corresponding Instruction: PrintTask

--------Information for t4--------
Task Name: t4
Description: ddd
Duration: 9.0
Type: Composite
Direct Prerequisite: t1
Indirect Prerequisites:
Subtasks: t3 t2
```

Create and print a composite task with name t4.

TroubleShooting:

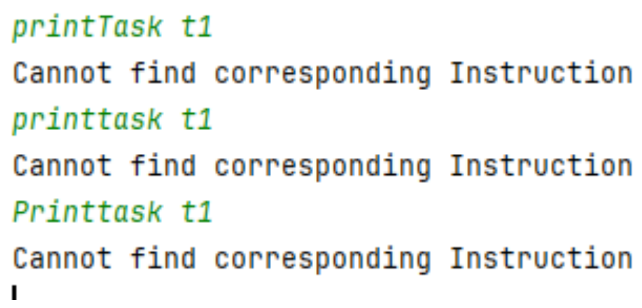
Some errors occur when you don’t have the correct command and capital letters for the command. For this command, the P and T should all be capitalized for successfully execute. At the same time, remember to execute the command with an existing task’s name or it will return an error.

Some mistake errors when having wrong commands.



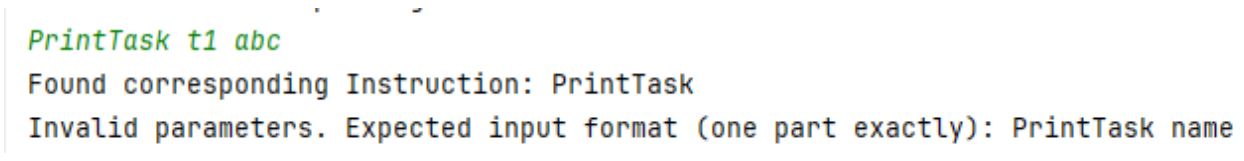
```
PrintTask t1
Found corresponding Instruction: PrintTask
Existing task not found. Please provide valid existing task name.
```

Error occurs when accesses to print a not existing task.

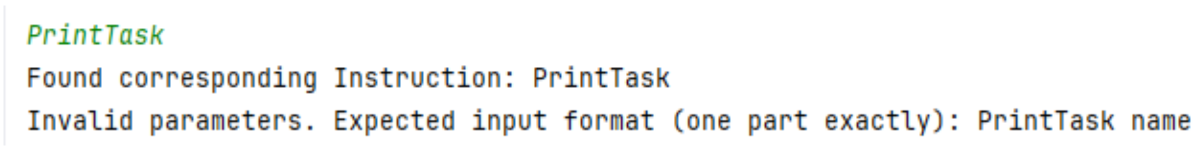


```
printTask t1
Cannot find corresponding Instruction
printtask t1
Cannot find corresponding Instruction
Printtask t1
Cannot find corresponding Instruction
```

Without P&&T capitalized when writing command.



```
PrintTask t1 abc
Found corresponding Instruction: PrintTask
Invalid parameters. Expected input format (one part exactly): PrintTask name
```



```
PrintTask
Found corresponding Instruction: PrintTask
Invalid parameters. Expected input format (one part exactly): PrintTask name
```

Error when input is not equal to one part (Taskname).

REQ6: PrintAllTasks

Description:

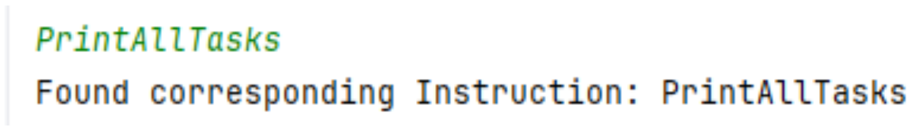
The functionalities for this command: Print the information of all existing tasks.

It can be used by calling the command: PrintAllTasks

Example: PrintAllTasks

Notes:

Remember to CreatePrimitive/Composite Tasks before you print the tasks, or it show nothing and return false

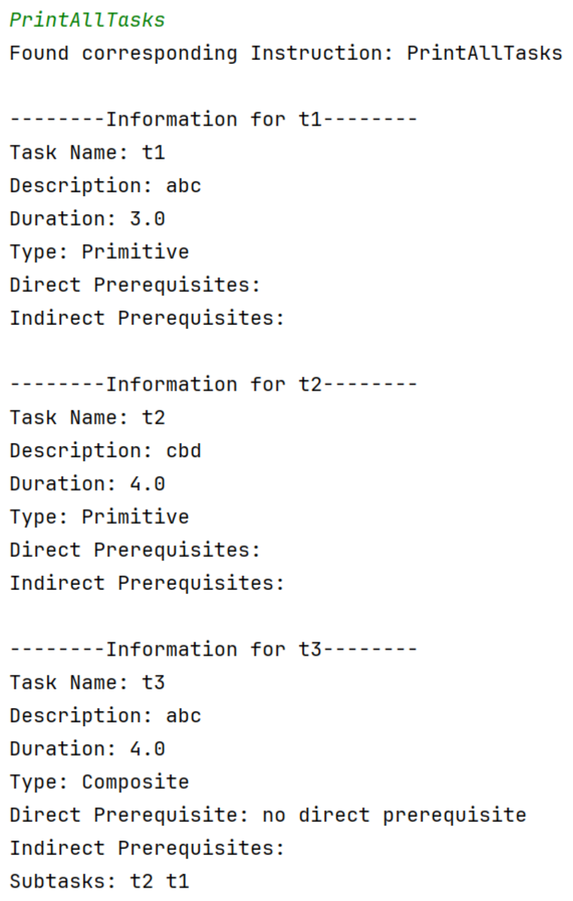


```
PrintAllTasks
Found corresponding Instruction: PrintAllTasks
```

Example Output if no tasks but asked to PrintAllTasks(No outputs will be printed)

Step by step Instructions:

Once you have successfully created many primitive or composite tasks, and you would like to check what are the information for all the task created, you can choose to use the PrintAllTasks command to check the information for all the tasks you have created.



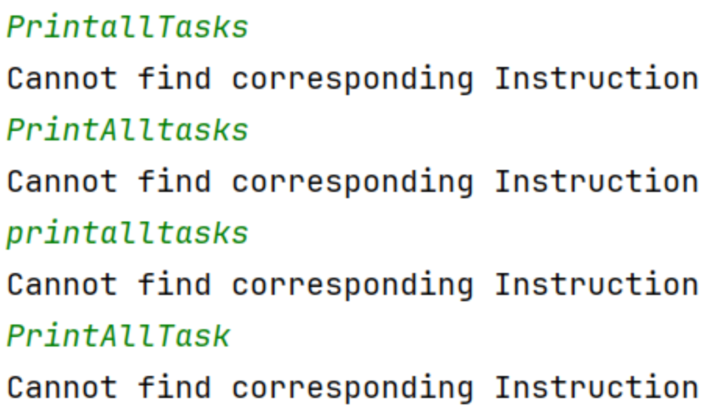
```
PrintAllTasks
Found corresponding Instruction: PrintAllTasks

--------Information for t1--------
Task Name: t1
Description: abc
Duration: 3.0
Type: Primitive
Direct Prerequisites:
Indirect Prerequisites:

--------Information for t2--------
Task Name: t2
Description: cbd
Duration: 4.0
Type: Primitive
Direct Prerequisites:
Indirect Prerequisites:

--------Information for t3--------
Task Name: t3
Description: abc
Duration: 4.0
Type: Composite
Direct Prerequisite: no direct prerequisite
Indirect Prerequisites:
Subtasks: t2 t1
```

Example Output when created primitive task t1, t2 and composite task t3

Troubleshooting:

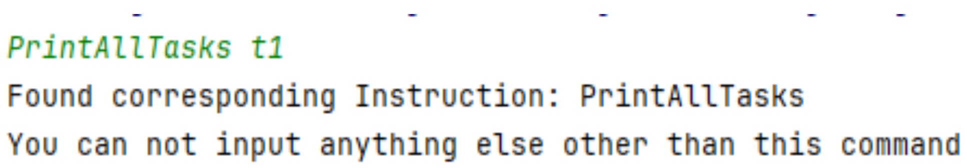
Some errors occur when you don’t have the correct command and capital letters for the command, for this command, the P, A, T should all be capitalized for successfully execute. At the same time, remember to add a s after PrintAllTask as the command is called PrintAllTasks.

Some mistake errors when having wrong commands.



```
PrintallTasks
Cannot find corresponding Instruction
PrintAlltasks
Cannot find corresponding Instruction
printalltasks
Cannot find corresponding Instruction
PrintAllTask
Cannot find corresponding Instruction
```

Without P&&A&&T capitalized or Without s when writing command



```
PrintAllTasks t1
Found corresponding Instruction: PrintAllTasks
You can not input anything else other than this command
```

Error when input anything else than the command

REQ7: ReportDuration

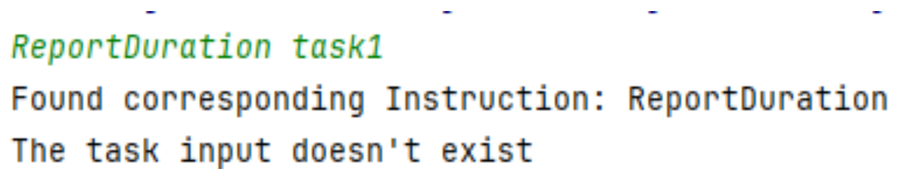
Description:

The functionalities for this command: Report the duration of a task

It can be used by calling the command: ReportDuration Taskname

Example: ReportDuration task1

Notes: Please be reminded that the duration of a composite task is the minimum number of hours needed to finish all its sub-tasks. Suppose that a composite task t0 has three subtasks t1, t2, and t3, with their durations being 1 hour, 2 hours, and 2 hours, respectively. Further suppose that both t1 and t2 are prerequisites for t3 and there is no prerequisite relation between t1 and t2. The duration of task t0 is then 4 hours. Also, please create a primitive or composite task before calling the ReportDuration command.



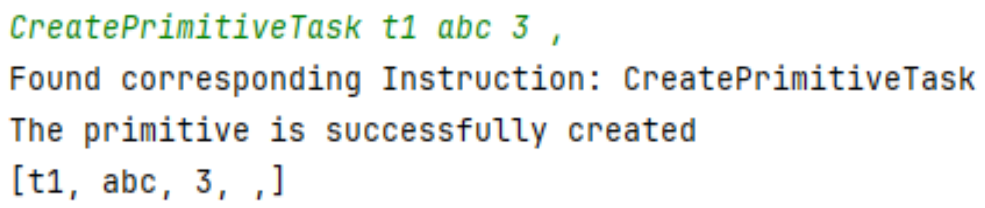
```
ReportDuration task1
Found corresponding Instruction: ReportDuration
The task input doesn't exist
```

When required to report duration of a not existing task.

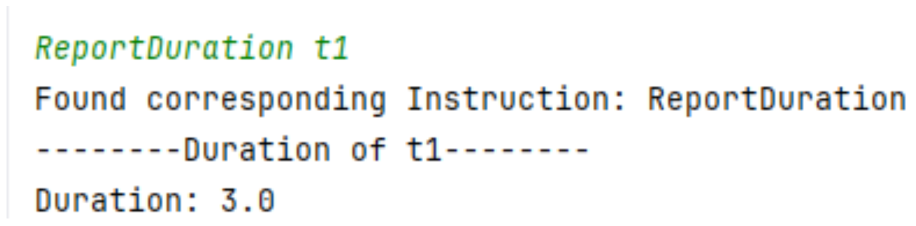
Step by step Instructions:

Once you have successfully created a primitive or composite task and you would like to check the duration time of a certain task. You can call the method ReportDuartion taskname to print out the duration time for the certain task.

Some examples of reporting earliest finish time of a task.

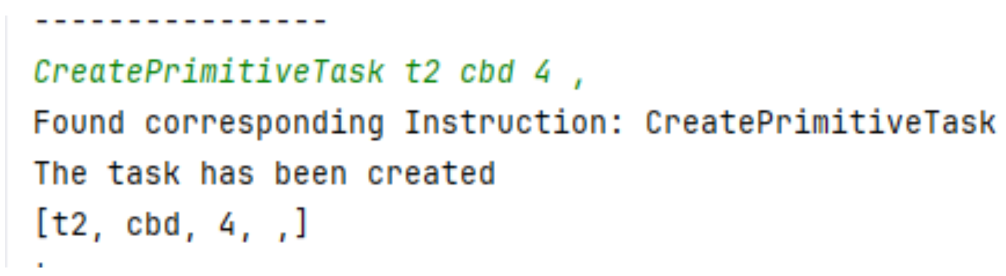


```
CreatePrimitiveTask t1 abc 3 ,
Found corresponding Instruction: CreatePrimitiveTask
The primitive is successfully created
[t1, abc, 3, ,]
```

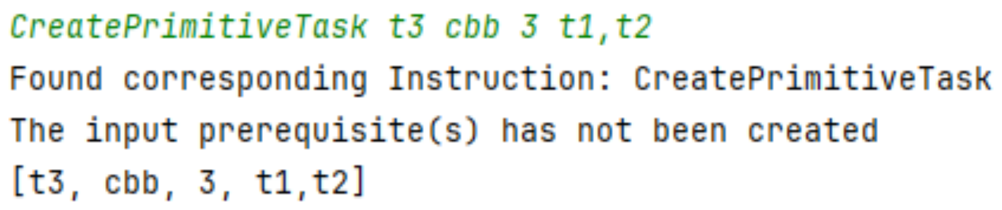


```
ReportDuration t1
Found corresponding Instruction: ReportDuration
--------Duration of t1--------
Duration: 3.0
```

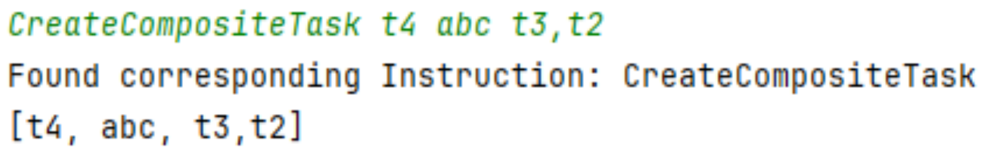
Example of reporting duration time of task t1(primitive task).



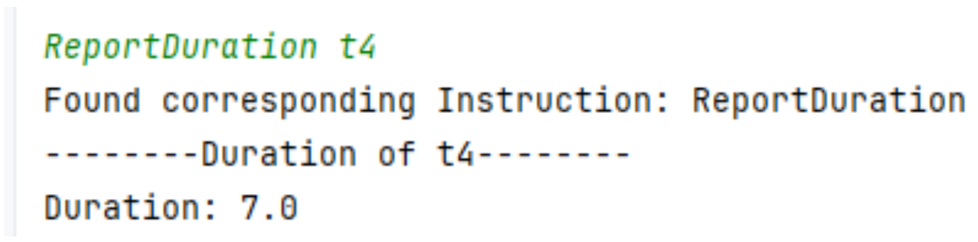
```
----------------
CreatePrimitiveTask t2 cbd 4 ,
Found corresponding Instruction: CreatePrimitiveTask
The task has been created
[t2, cbd, 4, ,]
```



```
CreatePrimitiveTask t3 cbb 3 t1,t2
Found corresponding Instruction: CreatePrimitiveTask
The input prerequisite(s) has not been created
[t3, cbb, 3, t1,t2]
```



```
CreateCompositeTask t4 abc t3,t2
Found corresponding Instruction: CreateCompositeTask
[t4, abc, t3,t2]
```



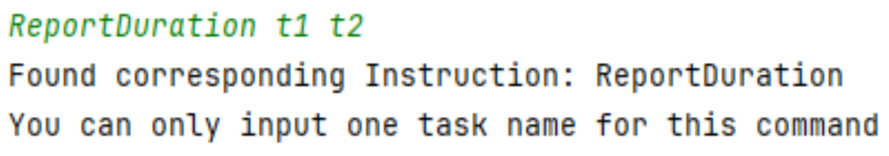
```
ReportDuration t4
Found corresponding Instruction: ReportDuration
--------Duration of t4--------
Duration: 7.0
```

Example of reporting duration time of task t4(composite tasks).

Troubleshooting:

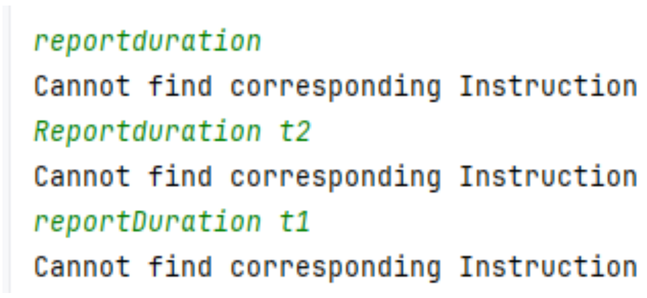
For some trouble errors, remember to capitalize R and D. Also remember that only one task can be reported of its duration in one time if two or more inputs are entered, the method will return an error.

Some mistake errors when having wrong commands.



```
ReportDuration t1 t2
Found corresponding Instruction: ReportDuration
You can only input one task name for this command
```

Error when input more than one task name to report.



```
reportduration
Cannot find corresponding Instruction
Reportduration t2
Cannot find corresponding Instruction
reportDuration t1
Cannot find corresponding Instruction
```

Without R&&D capitalized when writing command

REQ8: ReportEarliestFinishTime

Description:

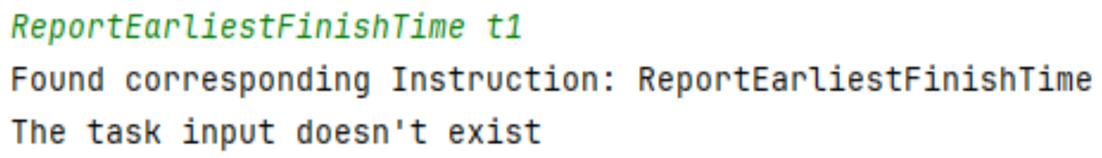
The functionalities for this command: Report the earliest finish time of a task.

It can be used by calling the command: ReportEarliestFinishTime name.

Example: ReportEarliestFinishTime task1

Notes: The earliest finish time of a task is calculated as the sum of 1) the earliest finish time of all its prerequisites and 2) the duration of itself.

Also, please create a primitive or composite task before calling the ReportEarliestFinishTime command.



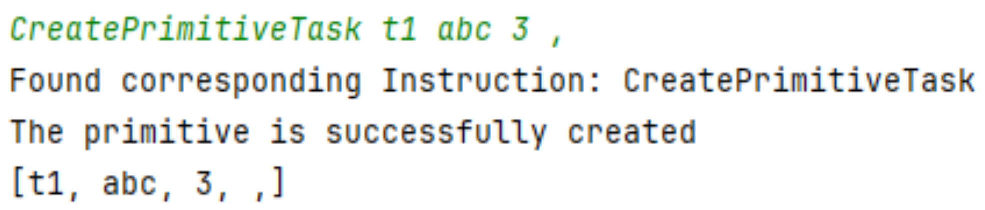
```
ReportEarliestFinishTime t1
Found corresponding Instruction: ReportEarliestFinishTime
The task input doesn't exist
```

When required to report earliest finish time of a not existing task.

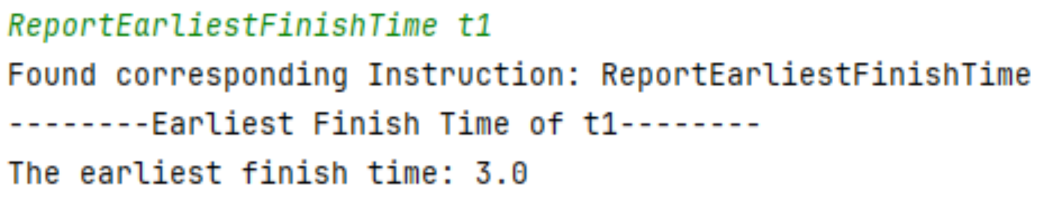
Step by step Instructions:

Once you have successfully created a primitive or composite task and you would like to check the earliest finish time of a certain task. You can call the method ReportEarliestFinishTime taskname to print out the earliest finish time for the certain task.

Some examples of reporting earliest finish time of a task

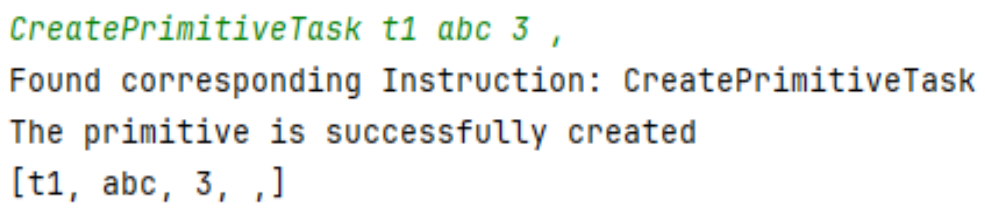


```
CreatePrimitiveTask t1 abc 3 ,
Found corresponding Instruction: CreatePrimitiveTask
The primitive is successfully created
[t1, abc, 3, ,]
```

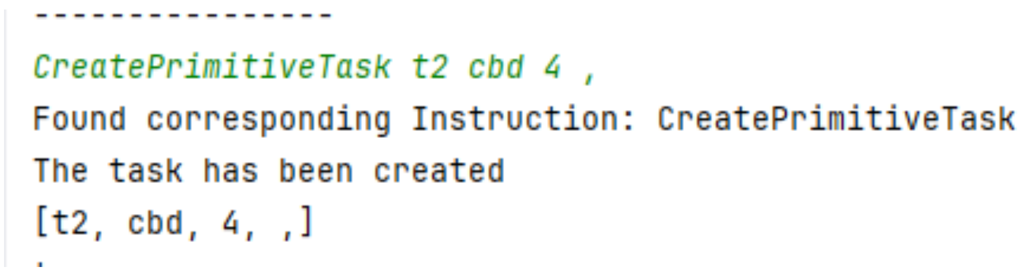


```
ReportEarliestFinishTime t1
Found corresponding Instruction: ReportEarliestFinishTime
--------Earliest Finish Time of t1--------
The earliest finish time: 3.0
```

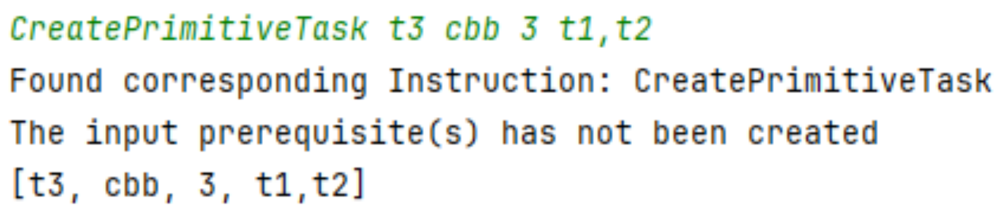
Example of reporting duration time of task t1(primitive task)



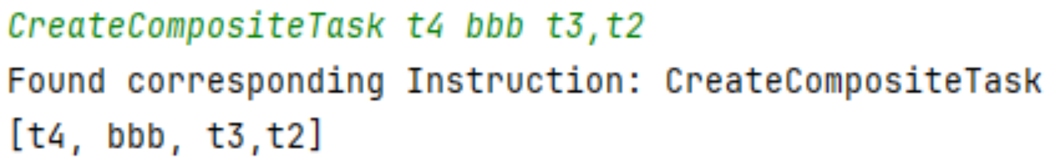
```
CreatePrimitiveTask t1 abc 3 ,
Found corresponding Instruction: CreatePrimitiveTask
The primitive is successfully created
[t1, abc, 3, ,]
```



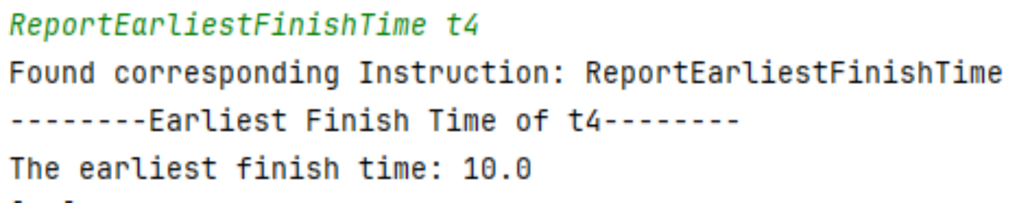
```
----------------
CreatePrimitiveTask t2 cbd 4 ,
Found corresponding Instruction: CreatePrimitiveTask
The task has been created
[t2, cbd, 4, ,]
```



```
CreatePrimitiveTask t3 cbb 3 t1,t2
Found corresponding Instruction: CreatePrimitiveTask
The input prerequisite(s) has not been created
[t3, cbb, 3, t1,t2]
```



```
CreateCompositeTask t4 bbb t3,t2
Found corresponding Instruction: CreateCompositeTask
[t4, bbb, t3,t2]
```



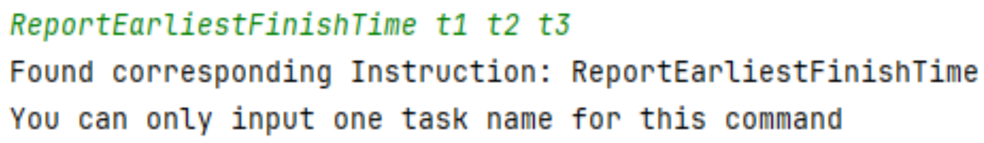
```
ReportEarliestFinishTime t4
Found corresponding Instruction: ReportEarliestFinishTime
--------Earliest Finish Time of t4--------
The earliest finish time: 10.0
```

Example of reporting earliest finish time of task t4(composite tasks)

Troubleshooting:

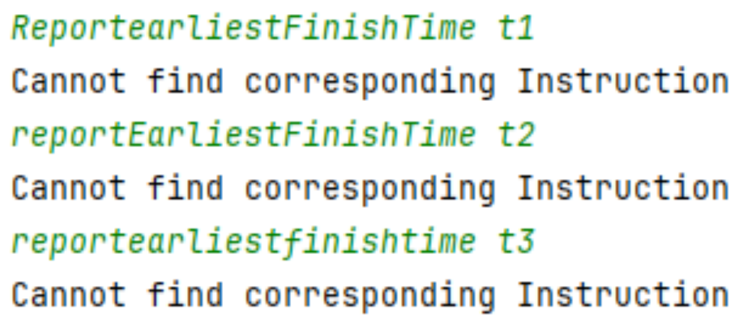
For some trouble errors, remember to capitalize R, E, F, and T. Also remember that only one task can be reported of its earliest finish time in one time if two or more inputs are entered, the method will return an error.

Some mistake errors when having wrong commands.



```
ReportEarliestFinishTime t1 t2 t3
Found corresponding Instruction: ReportEarliestFinishTime
You can only input one task name for this command
```

Error when input more than one task name to report.



```
ReportearliestFinishTime t1
Cannot find corresponding Instruction
reportEarliestFinishTime t2
Cannot find corresponding Instruction
reportearliestfinishtime t3
Cannot find corresponding Instruction
```

Without R&&E&&F&&T capitalized when writing command

REQ9: DefineBasicCriterion

Description:

The functionalities for this command: Define a basic task selection criterion.

It can be used by calling the command: DefineBasicCriterion name property op value.

Example:

DefineBasicCriterion criterion1 duration > 0.1

DefineBasicCriterion criterion2 name/description contains “a”

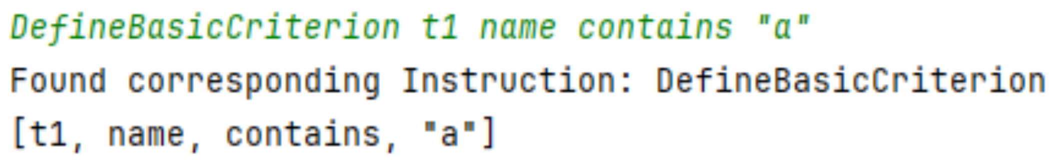
Notes:

name1 is a unique name for the new basic criterion. The construction of criterion names follows the same rules as task names. property is either name, description, duration, or prerequisites. If property is name or description, op must be contains and value must be a string in double quotes; If property is duration, op can be >, <, >=, <=, ==, or !=, and value must be a real value. If property is prerequisites or subtasks, op must be contains, and value must be a list of comma-separated task names

Step by step Instructions:

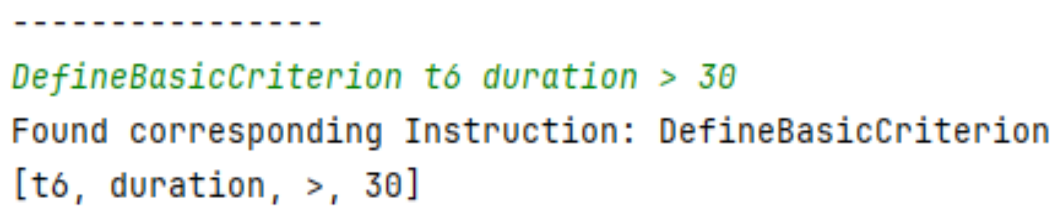
When you would like to create a basic criterion, you can call the DefineBasicCriterion command to create one basic criterion. You would need to follow the rules based on descriptions mentioned in notes and remember that the name can't be same as any other criterion defined in the past. If all notes and conditions are cleared, you can call and define an own basic criterion using the command DefineBasicCriterion name property op value.

Some examples of creating basic criterion



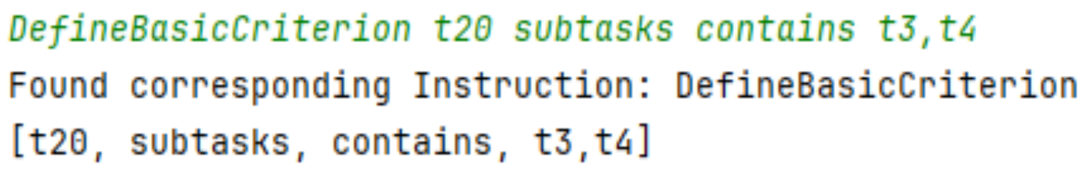
```
DefineBasicCriterion t1 name contains "a"
Found corresponding Instruction: DefineBasicCriterion
[t1, name, contains, "a"]
```

Creating a criterion named t1 requiring name contains "a"



```
----------------
DefineBasicCriterion t6 duration > 30
Found corresponding Instruction: DefineBasicCriterion
[t6, duration, >, 30]
```

Creating a criterion named t6 requiring duration larger than 30



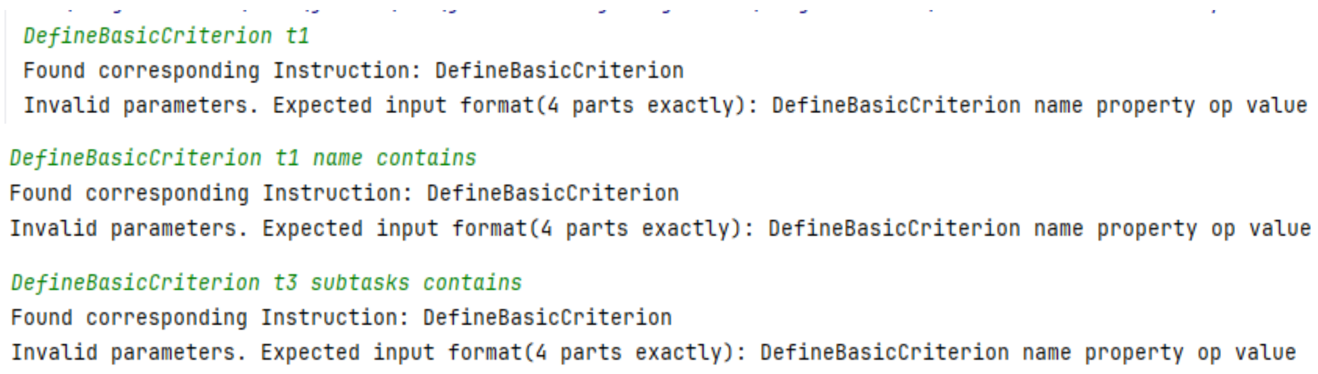
```
DefineBasicCriterion t20 subtasks contains t3,t4
Found corresponding Instruction: DefineBasicCriterion
[t20, subtasks, contains, t3,t4]
```

Creating a criterion name t20 requiring subtasks contains t3 and t4

Troubleshooting:

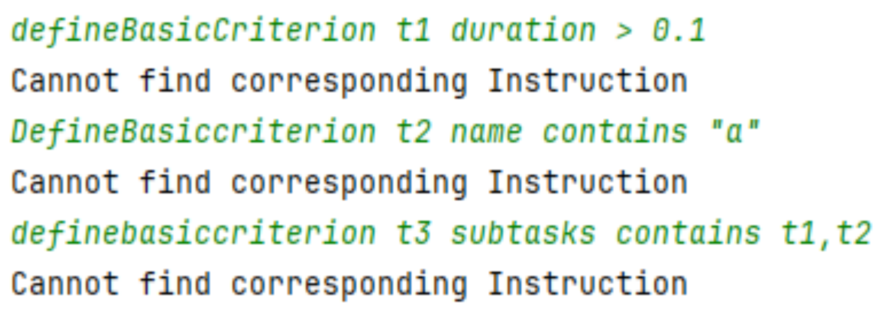
For some trouble errors, remember to capitalize D, B, and C when calling the command. Also remember that all the inputs entered must have exactly four parts after the command and should be in the order name, property, op, value if the value is not valid or in wrong order, it will return an error.

Some mistake errors when having wrong commands.



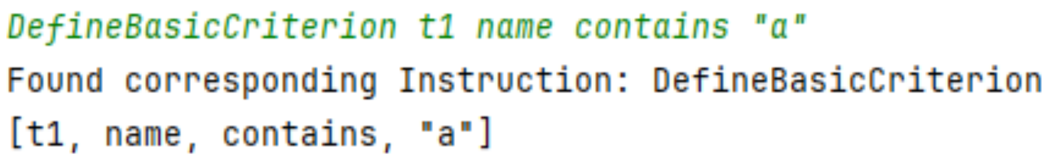
```
DefineBasicCriterion t1
Found corresponding Instruction: DefineBasicCriterion
Invalid parameters. Expected input format(4 parts exactly): DefineBasicCriterion name property op value
DefineBasicCriterion t1 name contains
Found corresponding Instruction: DefineBasicCriterion
Invalid parameters. Expected input format(4 parts exactly): DefineBasicCriterion name property op value
DefineBasicCriterion t3 subtasks contains
Found corresponding Instruction: DefineBasicCriterion
Invalid parameters. Expected input format(4 parts exactly): DefineBasicCriterion name property op value
```

Error when input not equal to four sessions (name property op value)

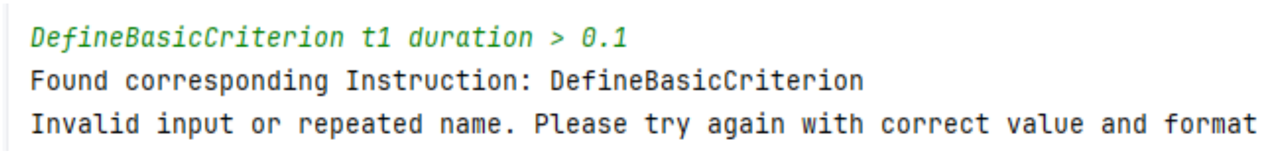


```
defineBasicCriterion t1 duration > 0.1
Cannot find corresponding Instruction
DefineBasiccriterion t2 name contains "a"
Cannot find corresponding Instruction
definebasiccriterion t3 subtasks contains t1,t2
Cannot find corresponding Instruction
```

Without D&&B&&C capitalized when writing command

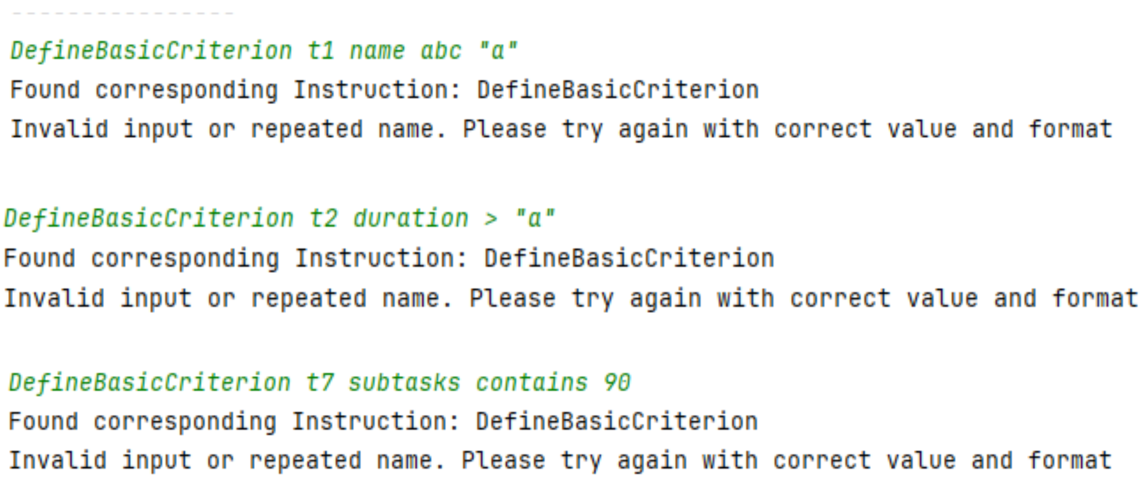


```
DefineBasicCriterion t1 name contains "a"
Found corresponding Instruction: DefineBasicCriterion
[t1, name, contains, "a"]
```



```
DefineBasicCriterion t1 duration > 0.1
Found corresponding Instruction: DefineBasicCriterion
Invalid input or repeated name. Please try again with correct value and format
```

Error when repeated name for new criterion (Already have an existing criterion named by the name for new criterion)



```
DefineBasicCriterion t1 name abc "a"
Found corresponding Instruction: DefineBasicCriterion
Invalid input or repeated name. Please try again with correct value and format

DefineBasicCriterion t2 duration > "a"
Found corresponding Instruction: DefineBasicCriterion
Invalid input or repeated name. Please try again with correct value and format

DefineBasicCriterion t7 subtasks contains 90
Found corresponding Instruction: DefineBasicCriterion
Invalid input or repeated name. Please try again with correct value and format
```

Error when values or properties entered aren't valid.

REQ10: IsPrimitive (defined in REQ13 Search for criterion searching)

Description:

IsPrimitive is a built-in criterion where users are not expected to manually create such a criterion. If you wish to filter all tasks that is of the primitive type, simply use Search IsPrimitive command to find them out.

Troubleshooting:

The system does not allow users to create a criterion named IsPrimitive, and it sends message on the screen in such situation.

REQ11: Define Binary Criterion/Define Negated Criterion

Description:

The functionalities for these two commands: To create a negated criterion or a binary criterion. Check notes for further explanation on what are exactly the criterion and how it is different from basic criterion.

Example:

Creating Negated Criterion command: DefineNegatedCriterion name1 name2

Example: DefineNegatedCriterion NegatedCriterion1 BasicCriterion1

Create Binary Criterion command: DefineBinaryCriterion name1 name2 logicOp name3.

Example: DefineBinaryCriterion BinaryCriterion1 BasicCriterion2 || BasicCriterion3

Notes:

The new criterion constructed using DefineNegatedCriterion is the negation of an existing criterion named name2. The new criterion constructed using DefineBinaryCriterion is name2 logicOp name3, where name2 and name3 are two existing criteria, while logicOp is either && or ||.

The logical operators, negation, and (&&), or (||) have the conventional precedence, associativity, and semantics.

Step by step Instructions:

First, before using the commands, you have to decide what kind of criterion you want to create.

For negated criterions, use the command DefineNegatedCriterion.

For creating negated criterions, remember to use an existing criterion to creates its negated criterion version. If the basic criterion entered not exist, it will return error. Also, the two names can't be repeated, if two of them are the same, it will return error.

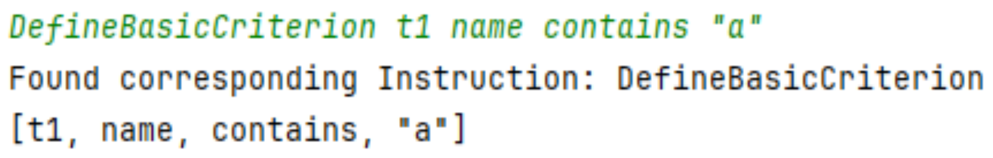
For binary criterions, use the command DefineBinaryCriterion.

For creating binary criterions, remember to use two existing criterions with logicOp (only && || allowed). If the two basic criterion s entered not exist or the logicOp isn't && or ||, it will return error. Also, the three names can't be repeated, if any two of them are the same, it will return error.

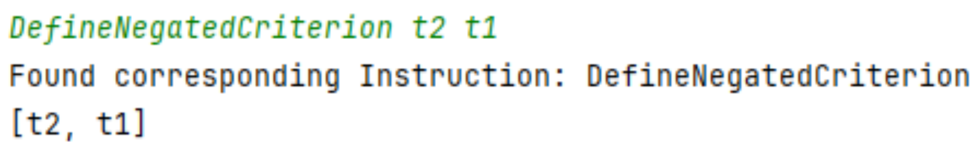
If conditions valid, you can create a negated criterion by: DefineNegatedCriterion newname existingcriterionname

If conditions valid, you can create a binary criterion by: DefineBinaryCriterion newname existingcriterionname1 (&& or ||) existingcriterionname2.

Some examples of creating negated/binary criterions

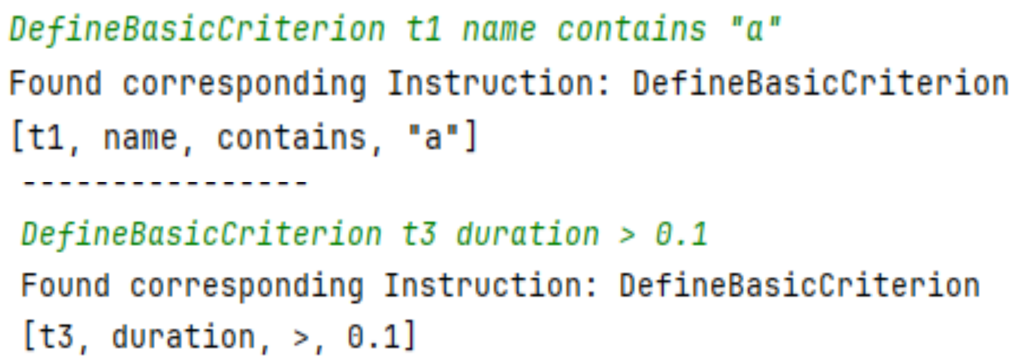


```
DefineBasicCriterion t1 name contains "a"
Found corresponding Instruction: DefineBasicCriterion
[t1, name, contains, "a"]
```

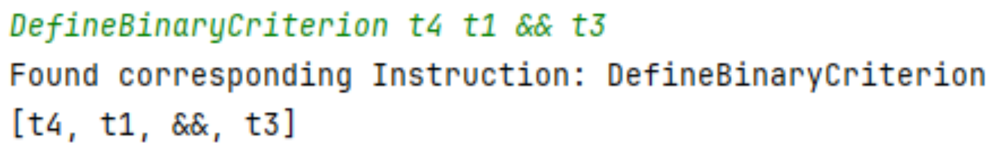


```
DefineNegatedCriterion t2 t1
Found corresponding Instruction: DefineNegatedCriterion
[t2, t1]
```

Defining a negated criterion named t2 being a negated version of basic criterion t1



```
DefineBasicCriterion t1 name contains "a"
Found corresponding Instruction: DefineBasicCriterion
[t1, name, contains, "a"]
 ----------------
 DefineBasicCriterion t3 duration > 0.1
 Found corresponding Instruction: DefineBasicCriterion
 [t3, duration, >, 0.1]
```



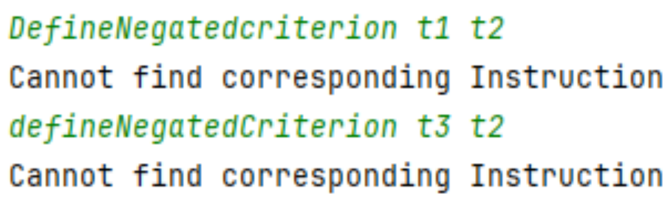
```
DefineBinaryCriterion t4 t1 && t3
Found corresponding Instruction: DefineBinaryCriterion
[t4, t1, &&, t3]
```

Defining a binary criterion name t4 having two basic criterions t1 and t3 connected by logicOp &&

Troubleshooting:

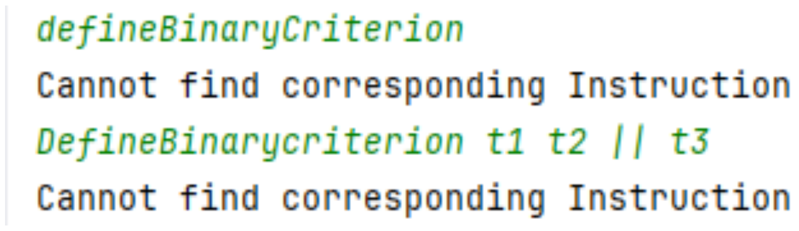
For some trouble errors, remember to capitalize D, B, and C when calling the command DefineBinaryCriterion. Remember to capitalize D, N, and C when calling the command DefineNegatedCriterion. Also remember that all the inputs entered must have exactly four parts after the command and should be in the order name1, name2 , logicOp, name3 for binary criterion. All the inputs entered must have exactly two parts after the command and should be in the order name1, name2 for negated criterion. If the value is not valid or in wrong order, it will return an error.

Some mistake errors when having wrong commands.



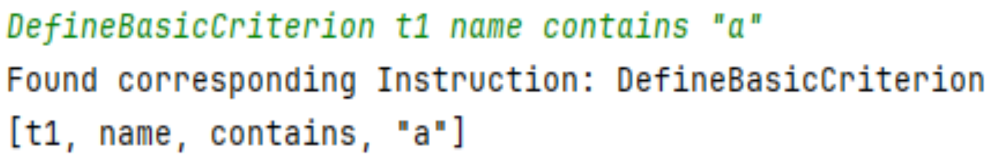
```
DefineNegatedcriterion t1 t2
Cannot find corresponding Instruction
defineNegatedCriterion t3 t2
Cannot find corresponding Instruction
```

Without D&&N&&C capitalized when writing command (Negated Criterion)

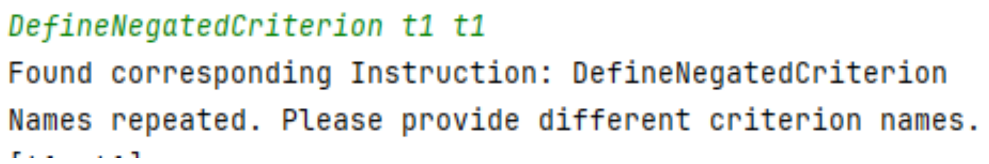


```
defineBinaryCriterion
Cannot find corresponding Instruction
DefineBinarycriterion t1 t2 || t3
Cannot find corresponding Instruction
```

Without D&&B&&C capitalized when writing command (Binary Criterion)

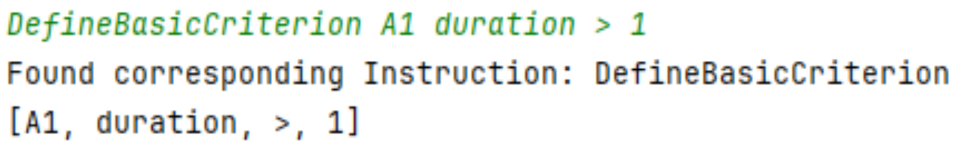


```
DefineBasicCriterion t1 name contains "a"
Found corresponding Instruction: DefineBasicCriterion
[t1, name, contains, "a"]
```

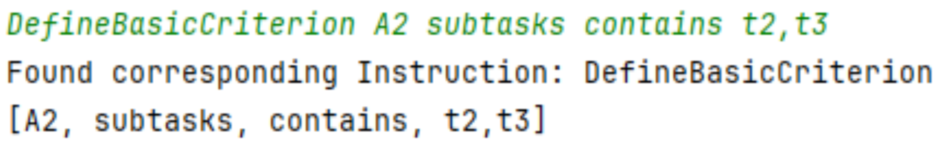


```
DefineNegatedCriterion t1 t1
Found corresponding Instruction: DefineNegatedCriterion
Names repeated. Please provide different criterion names.
```

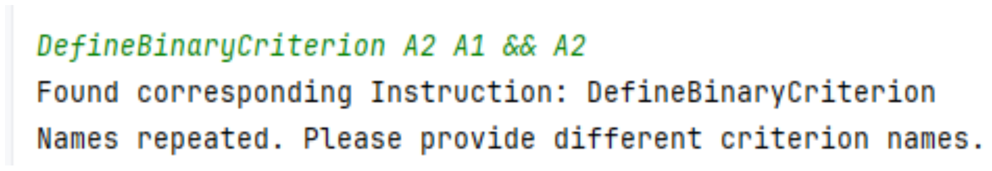
(Negated Criterion)



```
DefineBasicCriterion A1 duration > 1
Found corresponding Instruction: DefineBasicCriterion
[A1, duration, >, 1]
```



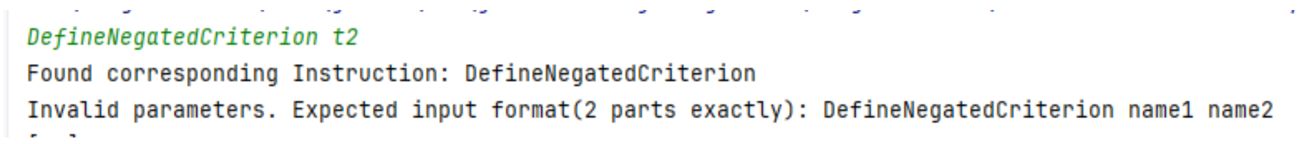
```
DefineBasicCriterion A2 subtasks contains t2,t3
Found corresponding Instruction: DefineBasicCriterion
[A2, subtasks, contains, t2,t3]
```



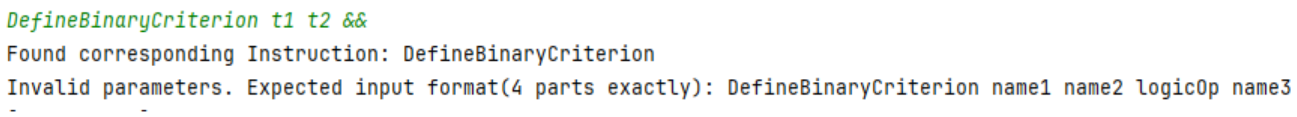
```
DefineBinaryCriterion A2 A1 && A2
Found corresponding Instruction: DefineBinaryCriterion
Names repeated. Please provide different criterion names.
```

(Binary Criterion)

Error when repeated name for new criterion (Already have an existing criterion named by the name for new criterion)

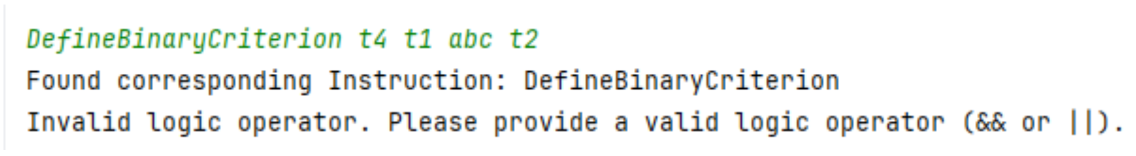


```
DefineNegatedCriterion t2
Found corresponding Instruction: DefineNegatedCriterion
Invalid parameters. Expected input format(2 parts exactly): DefineNegatedCriterion name1 name2
```



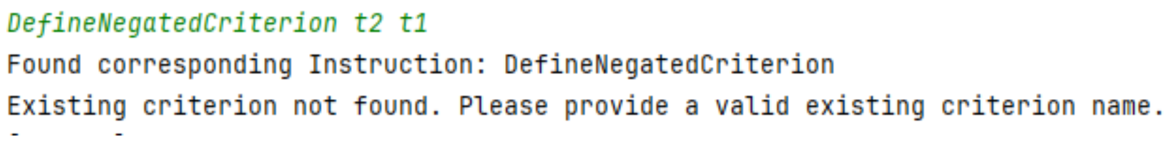
```
DefineBinaryCriterion t1 t2 &&
Found corresponding Instruction: DefineBinaryCriterion
Invalid parameters. Expected input format(4 parts exactly): DefineBinaryCriterion name1 name2 logicOp name3
```

Error when input number of parts not according to command requirements

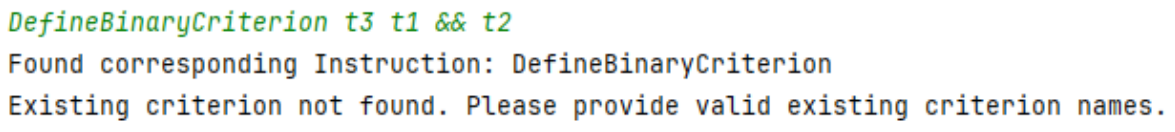


```
DefineBinaryCriterion t4 t1 abc t2
Found corresponding Instruction: DefineBinaryCriterion
Invalid logic operator. Please provide a valid logic operator (&& or ||).
```

Error when logicOp entered for binary criterion isn't valid.



```
DefineNegatedCriterion t2 t1
Found corresponding Instruction: DefineNegatedCriterion
Existing criterion not found. Please provide a valid existing criterion name.
```



```
DefineBinaryCriterion t3 t1 && t2
Found corresponding Instruction: DefineBinaryCriterion
Existing criterion not found. Please provide valid existing criterion names.
```

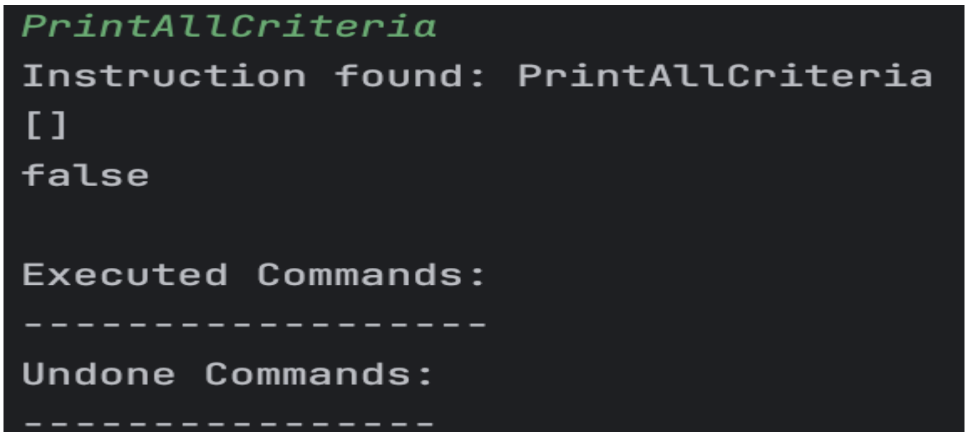
Error when name for existing tasks entered doesn't exist.

REQ12: Print All Criteria

Command description: The function of this command is to print all the criteria successfully defined. The printed information of a criterion includes property name, op, value, logicOp, and IsPrimitive.

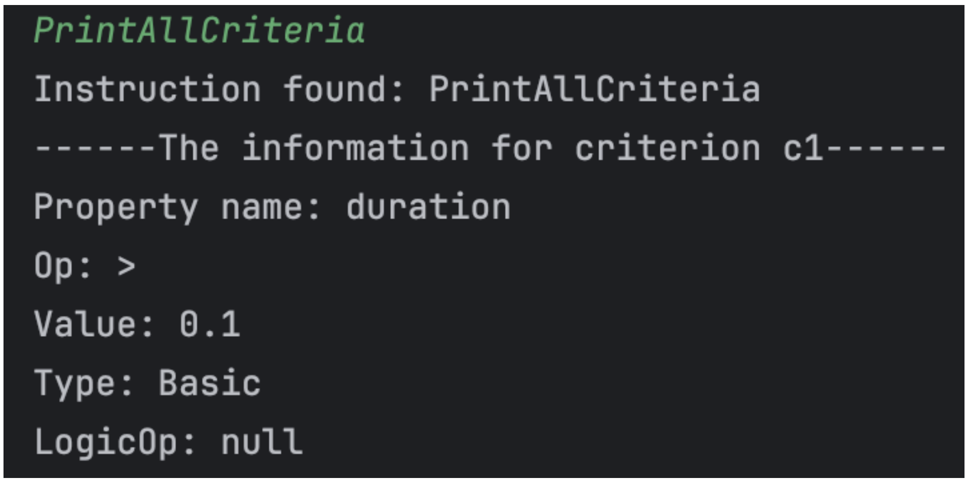
Example: PrintAllCriteria

Notes: If you enter the command “PrintAllCriteria” before defining any criterion, the system prints an empty “[ ]” and a “false” message.



```
PrintAllCriteria
Instruction found: PrintAllCriteria
[]
false

Executed Commands:
------------------
Undone Commands:
----------------
```

If you have already defined a criterion or several criteria, the output will be like this:



```
PrintAllCriteria
Instruction found: PrintAllCriteria
------The information for criterion c1------
Property name: duration
Op: >
Value: 0.1
Type: Basic
LogicOp: null
```

Step-by-step instructions: First, make sure that you have already defined at least one criterion using the commands mentioned above (i.e, REQ 9 and RED11). Then, enter the command PrintAllCriteria (pay attention to the upper-case letters), and you will see the result.

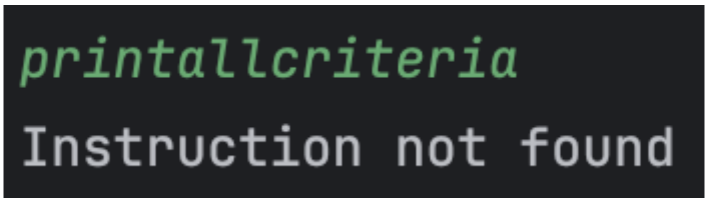
Troubleshooting: Below are some common issues that you may encounter when using this command.

1) No criterion defined:

If you enter this command before defining/creating any valid criterion as per REQ9 and REQ11, the command prints an empty “[ ]” for you and a “false” message as shown in the screenshot above. To solve this problem, you just need to define a valid criterion or several criteria. After that, the issue will be tackled.

2) Invalid command input (Instruction not found):

If you enter this command without caring the upper-case and lower-case of letters, the system will not find corresponding instruction. The result is shown in this screenshot:



```
printallcriteria
Instruction not found
```

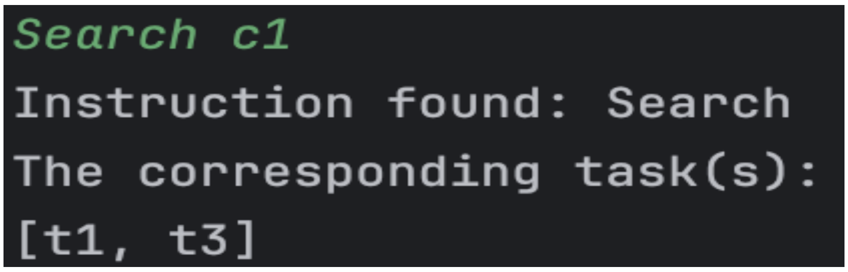
To solve it, please re-enter the command with correct spelling.

REQ13: Search name

Command description: The function of this command is to list the names of all tasks that satisfy the criterion based on user’s input (user’s input should be the name of a criterion).

Example 1: Search c1

By entering this command, the system will list the names of all tasks that satisfy the criterion named “c1”. The expected output is shown in the screenshot below.

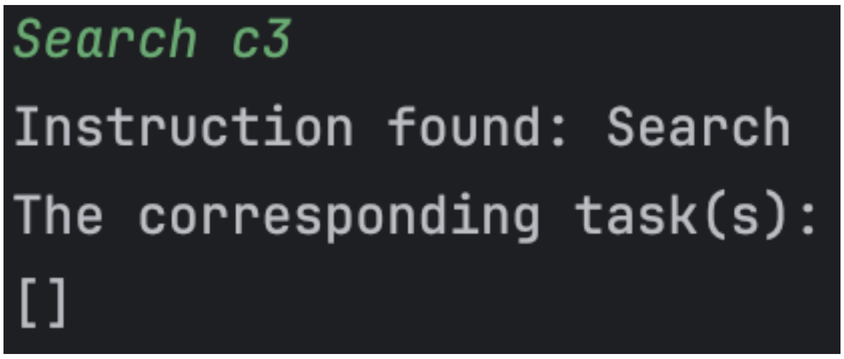


```
Search c1
Instruction found: Search
The corresponding task(s):
[t1, t3]
```

(t1 and t3 are two tasks)

Example 2: Search c3

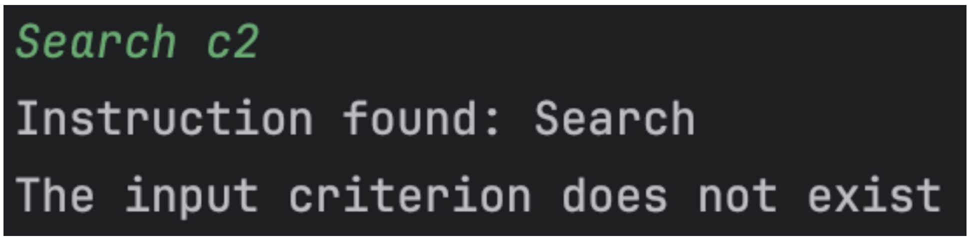
If no tasks satisfy the input criterion, you will see an empty “[ ]”. This is not an error. It indicates that no tasks were found based on the input criterion.



```
Search c3
Instruction found: Search
The corresponding task(s):
[]
```

(No tasks were found)

Step-by-step instructions: To realize this command's function, you need to have at least one criterion defined in the system. If there is no criterion defined, this command cannot function well by nature.



```
Search c2
Instruction found: Search
The input criterion does not exist
```

(c2 is an undefined criterion)

After you have defined at least one criterion, you can input the name of an existing criterion. Make sure that you type the "Search" and your criterion name correctly with one space in between.

Troubleshooting: You may encounter some common issues shown below.

1) Undefined criterion

   You are trying to search tasks based on an undefined criterion, which is naturally an error. Defining a criterion using the commands as per REQ9 and REQ11 solves this issue.

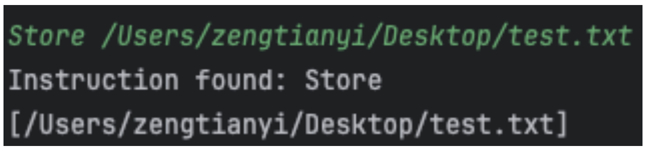
2) Error in the input format (Instruction not found)

   Upper-case letters, lower-case letters, space, and spelling of your defined criterion name are of importance. Double check your input before pressing the return button.

REQ14: Store path

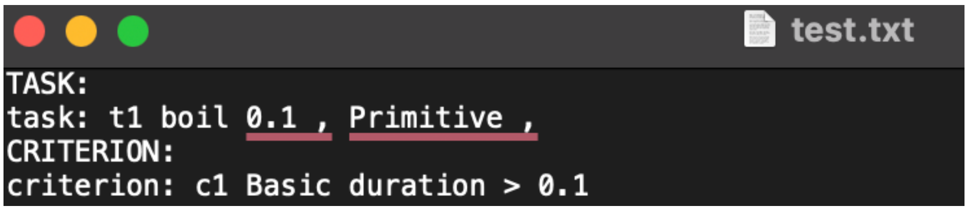
Command description: The function of this command is to store all defined tasks and criteria into a file at a given location.

Example: Store /Users/zengtianyi/Desktop/test.txt



```
Store /Users/zengtianyi/Desktop/test.txt
Instruction found: Store
[/Users/zengtianyi/Desktop/test.txt]
```

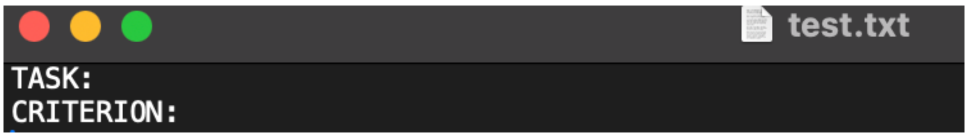
And the result after successful execution of this command:



```
test.txt
TASK:
task: t1 boil 0.1 , Primitive ,
CRITERION:
criterion: c1 Basic duration > 0.1
```

(In this case, the system has only one task and one criterion)

Besides, if there is no task or criterion defined, this command will still work but you will see nothing in the file. Here is the screenshot:



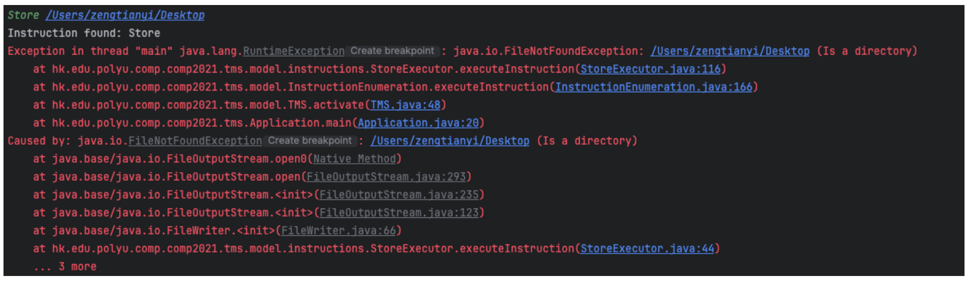
```
test.txt
TASK:
CRITERION:
```

Notes: To realize this function, it is better that a user create a file before entering the command. However, if the user doesn't create a file in advance, the command will help him create a file, but the user should provide a name of the file used to store data (e.g., data.txt) and its path. The best format of the file is .txt. because only .txt files are discovered to function well in storing the data and checking what has/have been stored. Although files of other formats may be compatible with the storing function (e.g., Docx), you may not be able to see what has/have been stored in the file as issues may occur when you try to open that file.

Step-by-step instructions: First, create a file (.txt) to store the tasks and criteria on your computer in advance. If you forget this, make sure that you provide a name in the format of xxx.txt in the “path” parameter when using the command. Then, enter the command “Store path” where the path is the path of your file on your computer used to store the tasks and criteria or to be created to store tasks or criteria. Make sure there is one space in between.

Troubleshooting: You may encounter some common issues when trying to use this command.

1) Failure to find the file: You provided an incorrect path of your file to be used for storage. The result is shown in the screenshot:



```
Store /Users/zengtianyi/Desktop
Instruction found: Store
Exception in thread "main" java.lang.RuntimeException Create breakpoint : java.io.FileNotFoundException: /Users/zengtianyi/Desktop (Is a directory)
    at hk.edu.polyu.comp.comp2021.tms.model.instructions.StoreExecutor.executeInstruction(StoreExecutor.java:116)
    at hk.edu.polyu.comp.comp2021.tms.model.InstructionEnumeration.executeInstruction(InstructionEnumeration.java:166)
    at hk.edu.polyu.comp.comp2021.tms.model.TMS.activate(TMS.java:48)
    at hk.edu.polyu.comp.comp2021.tms.Application.main(Application.java:20)
Caused by: java.io.FileNotFoundException Create breakpoint : /Users/zengtianyi/Desktop (Is a directory)
    at java.base/java.io.FileOutputStream.open0(Native Method)
    at java.base/java.io.FileOutputStream.open(FileOutputStream.java:293)
    at java.base/java.io.FileOutputStream.<init>(FileOutputStream.java:235)
    at java.base/java.io.FileOutputStream.<init>(FileOutputStream.java:123)
    at java.base/java.io.FileWriter.<init>(FileWriter.java:66)
    at hk.edu.polyu.comp.comp2021.tms.model.instructions.StoreExecutor.executeInstruction(StoreExecutor.java:44)
    ... 3 more
```

So, make sure that your path is detailed and correct like the example given above.

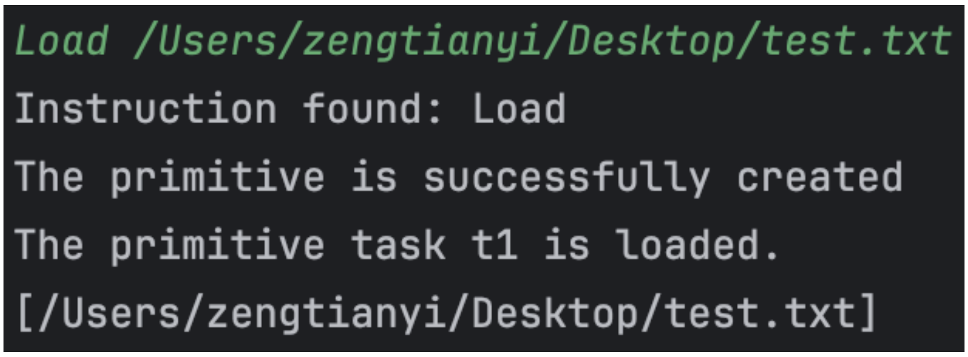
2) Format error (Instruction not found): As stated previously, you may get error by ignoring the upper-case and lower-case situations. Also, the space between Store and the path is important. Always double check your input before pressing the return button.

REQ15: Load path

Command description: The function of this command is to load the tasks and criteria stored in the file at a given path into the system and execute the task creation and criterion definition according to the file.

Example: Load /Users/zengtianyi/Desktop/test.txt

The result after the command execution:



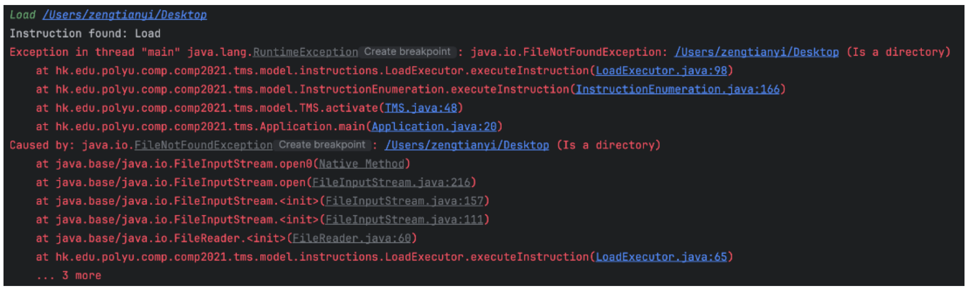
```
Load /Users/zengtianyi/Desktop/test.txt
Instruction found: Load
The primitive is successfully created
The primitive task t1 is loaded.
[/Users/zengtianyi/Desktop/test.txt]
```

(There is one task named t1 and no criteria in the file)

Step-by-step instructions: First, you need to have a file (the best format is .txt). Then, execute the Store command mentioned above to make this file have all defined tasks and criteria. In fact, the data stored in the file is commands of creating tasks and defining criteria (see the second screenshot in REQ14). After having such a file, enter this command: Store path, where path is the location of the file you wish to load.

Troubleshooting: You may encounter some common issues when executing this command.

1) Failure to find the file: You provided an incorrect path of your file to be used for loading. The result is shown below:



```
Load /Users/zengtianyi/Desktop
Instruction found: Load
Exception in thread "main" java.lang.RuntimeException Create breakpoint : java.io.FileNotFoundException: /Users/zengtianyi/Desktop (Is a directory)
    at hk.edu.polyu.comp.comp2021.tms.model.instructions.LoadExecutor.executeInstruction(LoadExecutor.java:98)
    at hk.edu.polyu.comp.comp2021.tms.model.InstructionEnumeration.executeInstruction(InstructionEnumeration.java:166)
    at hk.edu.polyu.comp.comp2021.tms.model.TMS.activate(TMS.java:48)
    at hk.edu.polyu.comp.comp2021.tms.Application.main(Application.java:20)
Caused by: java.io.FileNotFoundException Create breakpoint : /Users/zengtianyi/Desktop (Is a directory)
    at java.base/java.io.FileInputStream.open0(Native Method)
    at java.base/java.io.FileInputStream.open(FileInputStream.java:216)
    at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
    at java.base/java.io.FileInputStream.<init>(FileInputStream.java:111)
    at java.base/java.io.FileReader.<init>(FileReader.java:60)
    at hk.edu.polyu.comp.comp2021.tms.model.instructions.LoadExecutor.executeInstruction(LoadExecutor.java:65)
    ... 3 more
```

So,

make sure that your path is detailed and correct like the example given above.

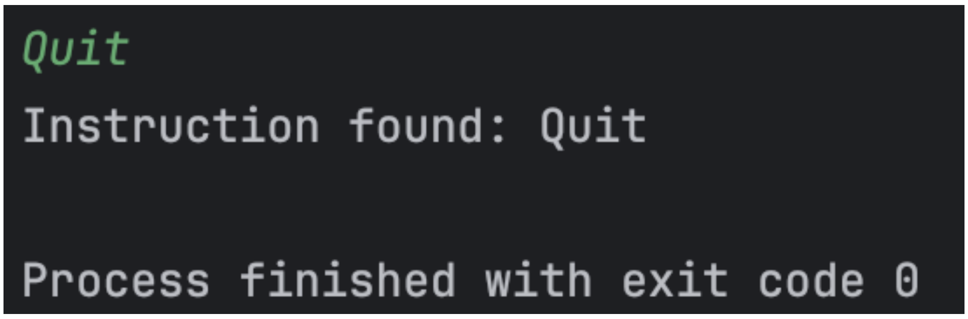
2) Format error (Instruction not found): As stated previously, you may get error by ignoring the upper-case and lower-case situations. Also, the space between Load and the path is important. Always double check your input before pressing the return button.

REQ16: Quit

Command description: The function of this command is to terminate the execution of the TMS.

Example: Quit

The result is shown in the screenshot:



```
Quit
Instruction found: Quit

Process finished with exit code 0
```

Step-by-step instructions: Just simply enter “Quit” and press return.

Troubleshooting: You may encounter this issue.

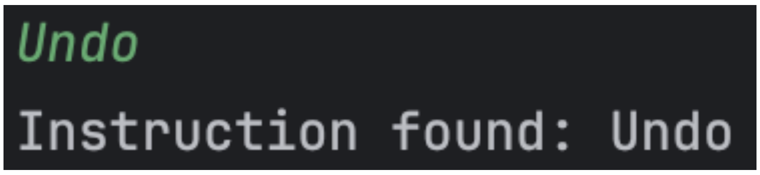
1) Format error (Instruction not found): As stated previously, you may get error by ignoring the upper-case and lower-case situations. Always double check your input before pressing the return button.

BON2: Undo and Redo

Command description: This requirement consists of two functions: undo and redo. "Undo" retrieves the last valid command (i.e., all commands except PrintTask, PrintAllTasks, ReportDuration, ReportEarliestFinishTime, PrintAllCriteria, Search, Store, Load, and Quit) and deletes the action made by that command. "Redo" retrieves the last valid command that was "undone" by the "Undo" command and re-execute it.

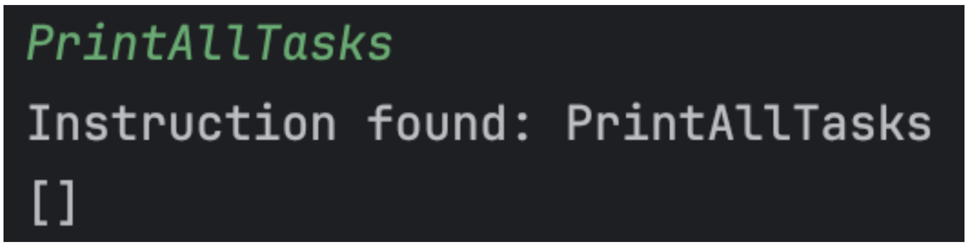
Example: Given that before calling Undo, the last command is: "CreatePrimitiveTask t1 boil 0.1 ," (The system just has this task)

Now, enter Undo, and the result is:



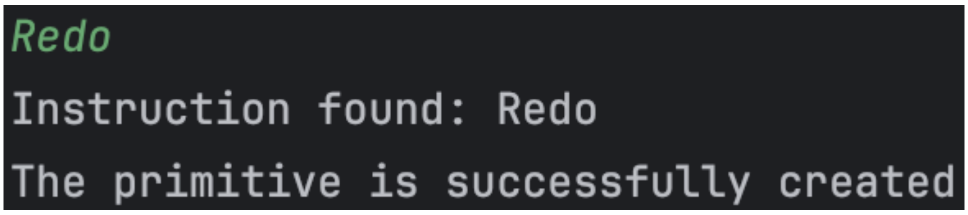
```
Undo
Instruction found: Undo
```

After this, enter "PrintAllTasks" to check whether the "Undo" was successful. The result is:



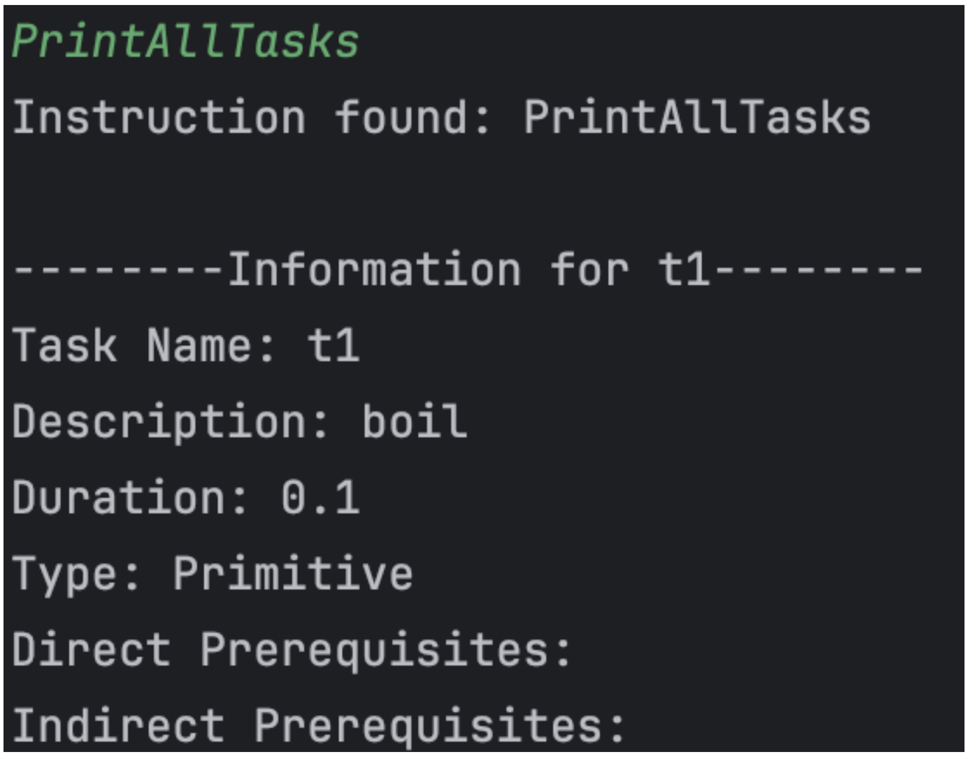
```
PrintAllTasks
Instruction found: PrintAllTasks
[]
```

This indicates that the “Undo” has been successfully executed. After that, enter “Redo” to re-execute the CreatePrimitiveTask command. The result is shown:



```
Redo
Instruction found: Redo
The primitive is successfully created
```

Now, enter “PrintAllTasks” to check whether the task (t1) has been re-created. The result is shown:



```
PrintAllTasks
Instruction found: PrintAllTasks

--------Information for t1--------
Task Name: t1
Description: boil
Duration: 0.1
Type: Primitive
Direct Prerequisites:
Indirect Prerequisites:
```

It indicates that “Redo” was successful.

Step-by-step instructions: To implement undo and redo, make sure there is at least one valid command (see Command description) that has been successfully executed. Also, you need to care about the order of Undo and Redo. If Redo is called before any execution of Undo, you get nothing as Redo re-executes the command “undone” by Undo. After being clear about the logic, it is safe to use these two commands.

Troubleshooting: You may encounter some common issues:

1) Format error (Instruction not found): As stated previously, you may get error by ignoring the upper-case and lower-case situations. Always double check your input before pressing the return button.

Additional Resources

Common FAQs

Where can I start the whole system? (In case you didn’t see the line mentioned in the introduction)

Run application.java then you can start the whole system.

Why some of my names for tasks doesn’t work?

Check the rules in command for tasks names.

Why my duration for tasks can’t be negative?

Check the rules in command for duration of tasks.

If you have any questions that are not included in the FAQ session

Here is the contact information for technical support.

On System introduction/usage contact ruijie.wang@connect.polyu.hk

On REQ 1-11 contact jinshun.zhu@connect.polyu.hk

On REQ 12-16/BON2 contact tianyi.zeng@connect.polyu.hk