SE 3XA3: Module Interface Specification
ScrumBot

Team 304, ScrumBot
Arkin Modi, modia1
Leon So, sol4
Timothy Choy, choyt2

Last Updated: April 6, 2020

Table 1: Revision History

Date	**Developer(s)**	**Change**
March 9, 2020	Timothy Choy	Create template, ScrumBot Module
March 10, 2020	Leon So	MeetingList module
March 11, 2020	Leon So	Meeting, MeetingTypes Modules
March 11, 2020	Leon So	Project, ProjectList, Meeting, MeetingTypes, Generic Dict Modules
March 11, 2020	Timothy Choy	Scrumbot Module
March 12, 2020	Leon So	Rename Meeting Module to Generic Meeting Module
March 12, 2020	Timothy Choy	Generic Meeting Module, Fixed Formatting
March 12, 2020	Arkin Modi	Fix Formatting
March 12, 2020	Leon So	Dict, Meeting, Project, Task, TaskList, Sprint Modules
March 12, 2020	Timothy Choy	Project, Meeting Modules
March 12, 2020	Timothy Choy	Completed ScrumBot module
March 13, 2020	Arkin Modi	Updated ScrumBot Module
April 4, 2020	Leon So	Revision 1 changes
April 5, 2020	Leon So	Revision 1 changes
April 6, 2020	Timothy Choy	Rev1 changes to Scrumbot and FileIO

ScrumBot Module

Module

~~AdminCog, ProjectCog, MeetingCog, MemberCog, Scrumbot, SprintCog, TaskCog~~
Scrumbot, ScrumbotCog

Uses

discord
discord.ext.commands
~~MeetingTypes~~
~~Generic Dictionary~~
~~MeetingList~~
~~Meeting~~
~~TaskList~~
~~Task~~
~~Sprint~~
ProjectList
Project
FileIO

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine Name	**In**	**Out**	**Exceptions**
on_ready			

Table 2: ScrumBot Programs

Routine Name	**In**	**Out**	**Exceptions**
load	commands.Cogs	String	MissingRole, Exception
reload	commands.Cogs	String	MissingRole, Exception
unload	commands.Cogs	String	MissingRole, Exception

Table 3: Admin Programs

Routine Name	**In**	**Out**	**Exceptions**
get_roles		discord.embed	
get_roles	String	discord.embed	

Table 4: Member Programs

Routine Name	**In**	**Out**	**Exceptions**
add_meeting	ℕ, String, Date, Time, MeetingT	String	MissingRole
add_meeting	ℕ, String, Date, Time, MeetingT, String	String	MissingRole
add_project	String	String	MissingRole
add_project	String, String	String	MissingRole
add_rqe	ℕ, String	String	MissingRole
add_sprint	ℕ	String	MissingRole
get_project_desc	ℕ	discord.embed	
get_rqes	ℕ	discord.embed	
get_sprints	ℕ	discord.embed	
list_meetings	ℕ	discord.embed	
list_projects		discord.embed	
rm_last_sprint	ℕ	String	MissingRole
rm_meeting	ℕ, ℕ	String	MissingRole
rm_project	ℕ	String	MissingRole
rm_rqe	ℕ	String	MissingRole
set_project_desc	ℕ, String	String	MissingRole

Table 5: Project Programs

Routine Name	**In**	**Out**	**Exceptions**
get_meeting_desc	ℕ, ℕ	discord.embed	
set_meeting_desc	ℕ, ℕ, String	String	MissingRole

Table 6: Meeting Programs

Routine Name	**In**	**Out**	**Exceptions**
add_feedback	ℕ, ℕ, ℕ, String	String	MissingRole
get_details	ℕ, ℕ, ℕ	discord.embed	
list_feedback	ℕ, ℕ, ℕ	discord.embed	
rm_feedback	ℕ, ℕ, ℕ, ℕ	String	MissingRole
set_details	ℕ, ℕ, String	String	MissingRole

Table 8: Task Programs

Routine Name	In	Out	Exceptions
add_task	$\mathbb{N}$, $\mathbb{N}$, String, Date, Time	String	MissingRole
add_task	$\mathbb{N}$, $\mathbb{N}$, String, Date, Time, String	String	MissingRole
list_tasks	$\mathbb{N}$, $\mathbb{N}$	discord.embed	
rm_task	$\mathbb{N}$, $\mathbb{N}$, $\mathbb{N}$	String	MissingRole

Table 7: Sprint Programs

Semantics

Environment Variables

TOKEN: A string constant containing the information linking ScrumBot to Discord's API

State Variables

startup_extensions: A list of command cogs.

State Invariant

None

Assumptions

This module is the interface between the users and the code. The input taken for the export access programs are text-based commands in Discord, and the output is also in text channels as text output. The methods also keep an output in the terminal as a log of commands used.

The exception to this assumption is the access routine on_ready as the output for the method is found in the terminal only. Due to this, there is no output value in the exported access programs table.

Though it is comprised of multiple files with many different purposes, the main purpose for these files are the same. For this reason, they have been grouped together into one module, with the secret being communication between the client from Discord and the system.

It is assumed that author refers to the user that initiated the command.

Access Routine Semantics

ScrumBot Programs

on_ready()

- transition: waits for ScrumBot to connect to Discord's API servers using the TOKEN, and upon successful connection, will print out the bot's name and id to the terminal. It is a check for successful connection to the server.

Admin Programs

load(*cog*)

- transition: attempts to load a Cog into ScrumBot through the use of the command `load_extension(cog)`
- output: $out := (exc \Rightarrow$ "Error: failed to load" $|| \; cog \mid \neg \; exc \Rightarrow cog \; ||$ "loaded")
- exception:

	$exc :=$
author_role ≠ admin	MissingRole
any issue with loading cog	Exception

reload(*cog*)

- transition: attempts to reload the cog in ScrumBot through unloading the cog then loading the cog
- output: $out := (exc \Rightarrow$ "Error: failed to reload" $|| \; cog \mid \neg \; exc \Rightarrow cog \; ||$ "reloaded")
- exception:

	$exc :=$
author_role ≠ admin	MissingRole
any issue with unloading or loading cog	Exception

unload(*cog*)

- transition: attempts to unload a cog from ScrumBot
- output: $out := (exc \Rightarrow$ "Error: failed to unload" $|| \; cog \mid \neg \; exc \Rightarrow cog \; ||$ "unloaded")
- exception:

	$exc :=$
author_role ≠ admin	MissingRole
any issue with unloading cog	Exception

Member Programs

get_roles()

- transition: finds all roles that are assigned to the author
- output: out := a discord.embed such that it contains the chosen author's name, icon, and a list of their roles

get_roles($name$)

- transition: takes in the name of a Discord user and finds all roles that are assigned to that user
- output: out := a discord.embed such that it contains the chosen user's name, icon, and a list of their roles

Project Programs

add_meeting($n, name, date, time, meeting, desc$)

- transition: ~~new Meeting($name, date, time, meeting$), add it to project n using add_meeting~~
  (proj = get_project(n)) $\Rightarrow$ proj.add_meeting($name, date, time, meeting$) and update file using fileio.write())
- output: out := (exc = MissingRole $\Rightarrow$ "MissingRole: You have insufficient permissions" | exc = TypeError $\Rightarrow$ "Failed to add meeting: meeting type must be GROOMING, STANDUP, RETROSPECTIVE, or SPRINTPLANNING." |
  $\neg$ exc $\Rightarrow$ "Successfully added meeting")
- exception: ~~exc := author_role $\neq$ Scrum Master $\Rightarrow$ MissingRole~~

	exc :=
author_role $\neq$ Scrum Master	MissingRole
$meeting\_type \notin$ {GROOMING, STANDUP, RETROSPECTIVE, SPRINTPLANNING}	TypeError

add_meeting($n, name, date, time, meeting, desc$)

- transition: ~~new Meeting($name, date, time, meeting, desc$), add it to project n using add_meeting~~
  (proj = get_project(n)) $\Rightarrow$ proj.add_meeting($name, date, time, meeting, desc$) and update file using fileio.write()
- output: out := (exc = MissingRole $\Rightarrow$ "MissingRole: You have insufficient permissions" | exc = TypeError $\Rightarrow$ "Failed to add meeting: meeting type must be GROOMING, STANDUP, RETROSPECTIVE, or SPRINTPLANNING." |
  $\neg$ exc $\Rightarrow$ "Successfully added meeting")
- exception: ~~exc := author_role $\neq$ Scrum Master $\Rightarrow$ MissingRole~~

	exc :=
author_role $\neq$ Scrum Master	MissingRole
$meeting\_type \notin$ {GROOMING, STANDUP, RETROSPECTIVE, SPRINTPLANNING}	TypeError

add_project($name$)

- transition: new Project($name$), add project to project_list, and create file using fileio.create()
- output: $out := (exc = \text{MissingRole} \Rightarrow$ "MissingRole: You have insufficient permissions" $| \neg\, exc \Rightarrow$ "Successfully added project")
- exception: $exc :=$ author_role $\neq$ admin $\Rightarrow$ MissingRole

add_project($name, desc$)

- transition: new Project($name, desc$), add project to project_list, and create file using fileio.create()
- output: $out := (exc = \text{MissingRole} \Rightarrow$ "MissingRole: You have insufficient permissions" $| \neg\, exc \Rightarrow$ "Successfully added project")
- exception: $exc :=$ author_role $\neq$ admin $\Rightarrow$ MissingRole

add_rqe(n, s)

- transition: add_rqe(s) in project n and update file using fileio.write()
- output: $out := (exc = \text{MissingRole} \Rightarrow$ "MissingRole: You have insufficient permissions" $| \neg\, exc \Rightarrow$ "Successfully added requirements")
- exception: $exc :=$ author_role $\neq$ Business Analyst $\Rightarrow$ MissingRole

add_sprint(n)

- transition: ~~new Sprint(), then add the sprint to project n using add_sprint(sprint)~~ (proj = get_project(n)) $\Rightarrow$ proj.add_sprint(), and update file using fileio.write()
- output: $out := (exc = \text{MissingRole} \Rightarrow$ "MissingRole: You have insufficient permissions" $| \neg\, exc \Rightarrow$ "Successfully added a sprint")
- exception: $exc :=$ author_role $\neq$ Scrum Master $\Rightarrow$ MissingRole

get_project_desc(n)

- output: $out :=$ A discord.embed such that it contains the project name and description of project n

get_rqes(n)

- output: $out :=$ A discord.embed such that it contains the project name and list of requirements of project n

get_sprints(n)

- output: *out* := A discord.embed such that it contains the project name and list of sprints of project n

list_meetings(n)

- output: *out* := A discord.embed such that it contains the project name and list of meetings of project n, as well as show the times for all the meetings

list_projects()

- output: *out* := A discord.embed containing all the created projects, identified by their name

rm_last_sprint(n)

- transition: removes the last sprint of project n using ~~rm_sprint()~~ proj.rm_sprint(), and update file using fileio.write()

- output: *out* := (*exc* = MissingRole ⇒ "MissingRole: You have insufficient permissions"
  | *exc* = IndexError ⇒ "No sprints found in project."
  | ¬ *exc* ⇒ "Successfully removed a sprint")

- exception: *exc* := author_role ≠ Scrum Master ⇒ MissingRole

rm_meeting(n, m)

- transition: removes a meeting from project n using rm_meeting(m), and update file using fileio.write()

- output: *out* := (*exc* = MissingRole ⇒ "MissingRole: You have insufficient permissions"
  | *exc* = KeyError ⇒ "Meeting not found."
  | ¬ *exc* ⇒ "Successfully removed a meeting")

- exception: *exc* := author_role ≠ Scrum Master ⇒ MissingRole

rm_project(n)

- transition: removes a project by their project id using remove(n), and remove file using fileio.delete()

- output: *out* := (*exc* = MissingRole ⇒ "MissingRole: You have insufficient permissions"
  | *exc* = KeyError ⇒ "Project not found."
  | ¬ *exc* ⇒ "Successfully removed a project")

- exception: *exc* := author_role ≠ admin ⇒ MissingRole

rm_rqe(n, m)

- transition: remove requirement m from project n using rm_rqe(m), and update file using fileio.write()
- output: out := (exc = MissingRole ⇒ "MissingRole: You have insufficient permissions"
  | exc = IndexError ⇒ "Requirement not found."
  | ¬ exc ⇒ "Successfully removed a requirement")
- exception: exc := author_role ≠ Business Analyst ⇒ MissingRole

set_project_desc(n, s)

- transition: set_desc(s) in project n, and update file using fileio.write()
- output: out := (exc = MissingRole ⇒ "MissingRole: You have insufficient permissions"
  | ¬ exc ⇒ "Successfully set description")
- exception: exc := author_role ≠ Business Analyst ∧ author_role ≠ admin ⇒ MissingRole

Meeting Programs

get_meeting_desc(n, m)

- output: out := (exc = KeyError ⇒ "Meeting not found." | ¬ exc ⇒ A discord.embed such that it contains the meeting description of meeting m in project n, showing the meeting name and project name as well.)

set_meeting_desc(n, m, s)

- transition: set_desc(s) for meeting m in project n, and update file using fileio.write()
- output: out := (exc = MissingRole ⇒ "MissingRole: You have insufficient permissions"
  | exc = KeyError ⇒ "Meeting not found."
  | ¬ exc ⇒ "Successfully set description")
- exception: exc := author_role ≠ Scrum Master ⇒ MissingRole

Sprint Programs

add_task(n, ~~m,~~ $name, date, time$)

- transition: ~~new Task($name, date, time$) then add_task(task) in sprint m, project n~~ (proj = get_project(n) ⇒ proj.add_task($name, date + time$, None), and update file using fileio.write()
- output: out := (exc = MissingRole ⇒ "MissingRole: You have insufficient permissions"
  | exc = IndexError ⇒ "Sprint not found."
  | ¬ exc ⇒ "Successfully added task")
- exception: exc := author_role ≠ Scrum Master ⇒ MissingRole

add_task(n, ~~m,~~ $name, date, time, detail$)

- transition: ~~new Task($name, date, time, detail$) then add_task(task) in sprint m, project n~~
  (proj = get_project(n) ⇒ proj.add_task($name, date + time, detail$), and update file using fileio.write()

- output: out := (exc = MissingRole ⇒ "MissingRole: You have insufficient permissions"
  | exc = IndexError ⇒ "Sprint not found."
  | ¬ exc ⇒ "Successfully added task")

- exception: exc := author_role ≠ Scrum Master ∧ author_role ≠ Business Analyst ⇒ MissingRole

list_tasks(n, m)

- output: out := (exc = IndexError ⇒ "Sprint not found."
  ¬ exc ⇒ A discord.embed such that it lists all the tasks of sprint m in project n, as well as their deadlines)

rm_task(n, ~~m,~~ k)

- transition: ~~rm_task(k) in sprint m in project n~~
  (proj = get_project(n)) ⇒ proj.rm_task(k), and update file using fileio.write()

- output: out := (exc = MissingRole ⇒ "MissingRole: You have insufficient permissions"
  | exc = IndexError ⇒ "Sprint not found."
  | exc = KeyError ⇒ "Task not found."
  | ¬ exc ⇒ "Successfully removed task")

- exception: exc := author_role ≠ Scrum Master ∧ author_role ≠ Business Analyst ⇒ MissingRole

Task Programs

add_feedback(n, ~~m,~~ k, s)

- transition: add_feedback(k,s) in task k, ~~sprint m,~~ project n, and update file using fileio.write()

- output: out := (exc = MissingRole ⇒ "MissingRole: You have insufficient permissions"
  | exc = IndexError ⇒ "Sprint not found."
  | exc = KeyError ⇒ "Task not found."
  | ¬ exc ⇒ "Successfully added feedback")

- exception: exc := author_role ≠ Scrum Master ⇒ MissingRole

get_details(n, m, k)

- output: $out :=$ ($exc =$ IndexError $\Rightarrow$ "Sprint not found."
  $|\ \neg\ exc \Rightarrow$ A discord.embed such that it shows the details of task k, in sprint m, in project n, using get_details())

list_feedback(n, m, k)

- output: $out :=$ ($exc =$ IndexError $\Rightarrow$ "Sprint not found."
  $|\ exc =$ KeyError $\Rightarrow$ "Task not found."
  $|\ \neg\ exc \Rightarrow$ A discord.embed such that it lists all the feedback of task k, in sprint m, in project n, using get_feedback())

rm_feedback(n, ~~m,~~ k, x)

- transition: rm_feedback(k, x) for task k ~~in sprint m~~ in project n, and update file using fileio.write()

- output: $out :=$ ($exc =$ IndexError $\Rightarrow$ "Sprint not found."
  $|\ exc =$ KeyError $\Rightarrow$ "Task not found."
  $|\ \neg\ exc \Rightarrow$ "Successfully removed feedback")

- exception: $exc :=$ author_role $\neq$ Scrum Master $\Rightarrow$ MissingRole

set_details(n, ~~m,~~ k, s)

- transition: set_details(k, s) for task k ~~in sprint m~~ in project n, and update file using fileio.write()

- output: $out :=$ ($exc =$ IndexError $\Rightarrow$ "Sprint not found."
  $|\ exc =$ KeyError $\Rightarrow$ "Task not found."
  $|\ \neg\ exc \Rightarrow$ "Successfully set details")

- exception: $exc :=$ author_role $\neq$ Scrum Master $\wedge$ author_role $\neq$ Business Analyst $\Rightarrow$ MissingRole

Local Functions

get_project: $\mathbb{N} \rightarrow$ Project
get_project$(n) \equiv$ project such that project $\in$ project_list $\wedge$ project.key() $= n$

MeetingTypes Module

Module

MeetingTypes

Uses

N/A

Syntax

Exported Constants

N/A

Exported Types

MeetingT = {Grooming, StandUp, Retrospective, SprintPlanning}

Exported Access Programs

None

Semantics

State Variables

None

State Invariant

None

Generic Dictionary Module

Generic Template Module

Dict(T)

Uses

N/A

Syntax

Exported Types

Dict = ?

Exported Constants

None

Exported Access Programs

Routine Name	**In**	**Out**	**Exceptions**
new Dict(T)		Dict(T)	
add	T		
update	$\mathbb{N}$, T		
remove	$\mathbb{N}$		KeyError
get_last_id		$\mathbb{N}$	
to_seq		seq of ($\mathbb{N}$,T)	

Semantics

State Variables

d: seq of ($\mathbb{N}$,T)
c: $\mathbb{N}$

State Invariant

$|d| \geq 0$
$c \geq 0$

Assumptions & Design Decisions

The Dict(T) constructor is called for each object instance before any other access routine is called for that object.

It is assumed that the first term in Dict is referred to as the ”key” and the second term is the ”value”.

Access Routine Semantics

new Dict()

- transition: $d, c := \langle\rangle, 0$
- output: $out :=$ self

add(e)

- transition: $d := d \mid\mid \langle c,\ e\rangle, c := c + 1$

update(id,e)

- transition: $d[id], c := e, id + 1$

remove(id)

- transition: $d := d - \langle id, e\rangle$
- exception: $exc := \langle id, e\rangle \notin d \Rightarrow$ KeyError

get_last_id()

- output: $out := c - 1$

to_seq()

- output: $out := d$ such that $(\forall i \in \mathbb{N} \mid 0 \leq i < |d| \cdot d[i].key() \leq d[i + 1].key())$

MeetingList Module

Template Module

MeetingList is a Dict(Meeting)

Meeting Module

Module

Meeting

Uses

MeetingTypes

Syntax

Exported Constants

None

Exported Types

Meeting = ?

Exported Access Programs

Routine Name	In	Out	Exceptions
new Meeting	String, ~~Date, Time~~ String, MeetingT	Meeting	
new Meeting	String, ~~Date, Time~~ String, MeetingT, String	Meeting	
get_name		String	
get_datetime		String	
~~get_date~~		~~Date~~	
~~get_time~~		~~Time~~	
get_type		MeetingT	
get_desc		String	
set_desc	String		

Semantics

State Variables

name: String
~~*date*: Date~~
~~*time*: Time~~
d: Datetime
type: MeetingT
desc: String

State Invariant

None

Assumptions

- The Meeting constructor is called for each object instance before any other access routine is called for that object.

Access Routine Semantics

new Meeting(n, ~~d, t~~ dt, $type$)

- transition: $name$, ~~$time, date$~~, d, $type$, $desc := n$, ~~d, t~~ $Datetime(dt)$, $type$, None
- output: $out :=$ self

new Meeting(n, ~~d, t~~ dt, $type$, $desc$)

- transition: $name$, ~~$time, date$~~, d, $type$, $desc := n$, ~~d, t~~ $Datetime(dt)$, $type$, $desc$
- output: $out :=$ self

~~get_name()~~

- ~~$output : out := name$~~

~~get_date()~~

- ~~$output : out := date$~~

~~get_time()~~

- ~~$output : out := time$~~

get_datetime()

- output: $out := String(d)$

get_type()

- output: $out := type$

get_desc()

- output: $out :=$ ~~$desc$~~($desc =$ None $\Rightarrow$ "No description" $|$ $desc$)

set_desc(s)

- transition: $desc := s$

TaskList Module

Template Module

TaskList is a Dict(Task)

Task Module

Module

Task

Uses

None

Syntax

Exported Constants

None

Exported Types

Task = ?

Exported Access Programs

Routine Name	In	Out	Exceptions
new Task	String, ~~Date, Time~~ String	Task	
new Task	String, ~~Date, Time~~ String, Details	Task	
get_name		String	
get_deadline		~~(Date, Time)~~ String	
get_details		String	
get_feedback		seq of String	
add_feedback	String		
rm_feedback	$\mathbb{N}$		
set_details	String		

Semantics

State Variables

name: String
deadline: ~~(Date, Time)~~ Datetime
details: String
feedback: seq of String

State Invariant

None

Assumptions

- The Task constructor is called for each object instance before any other access routine is called for that object.

Access Routine Semantics

new Task(s, ~~d, t~~ dt)

- transition: $name, deadline, details := s$, ~~$(d, t)$~~ $Datetime(dt)$, None
- output: $out :=$ self

new Task(s, ~~d, t~~ $dt, details$)

- transition: $name, deadline, details := s$, ~~$(d, t)$~~ $Datetime(dt), details$
- output: $out :=$ self

get_name()

- output: $out := name$

get_deadline()

- output: $out :=$ ~~$deadline$~~$String(deadline)$

get_details()

- output: $out := (details = \text{None} \Rightarrow \text{“No details”} \mid details)$

get_feedback()

- output: $out := feedback$

add_feedback(s)

- transition: $feedback := feedback \mid\mid s$

rm_feedback(s)

- transition: $feedback := feedback - s$

set_details(s)

- transition: $details := s$

Sprint Module

Module

Sprint

Uses

TaskList, Task

Syntax

Exported Constants

None

Exported Types

Sprint = ?

Exported Access Programs

Routine Name	**In**	**Out**	**Exceptions**
new Sprint		Sprint	
new Sprint	Date	Sprint	
get_date		String	
get_feedback	$\mathbb{N}$	String	KeyError
get_last_task_id		$\mathbb{N}$	
get_tasks		seq of Task	
get_task	$\mathbb{N}$	Task	
add_feedback	$\mathbb{N}$, String		
add_task	~~Task~~ String, String		
add_task	String, String, String		
add_task_from_file	$\mathbb{N}$, String, String		
add_task_from_file	$\mathbb{N}$, String, String, String		
rm_feedback	$\mathbb{N}, \mathbb{N}$		KeyError
rm_task	$\mathbb{N}$		
set_details	$\mathbb{N}$, String		KeyError

Semantics

State Variables

$tasks$: TaskList
$date$: Date

State Invariant

None

Assumptions

- The Sprint constructor is called for each object instance before any other access routine is called for that object.

Access Routine Semantics

new Sprint(n)

- transition: $tasks, date := TaskList(), Datetime.today()$
- output: $out :=$ self

new Sprint(n, d)

- transition: $tasks, date := TaskList(), Datetime(d)$
- output: $out :=$ self

get_date()

- output: $out := String(date)$

get_tasks(~~n~~)

- output: $out := tasks.to\_seq()$

get_task(n)

- output: $out := get\_tasks()[n]$

get_feedback(n)

- output: $out := get\_task(n).get\_feedback()$
- exception: $exc := get\_task(n) ==$ None $\Rightarrow$ KeyError

get_last_task_id()

- output: $out := tasks.get\_last\_id()$

add_feedback(n, s)

- transition: $:= get\_tasks(n).add\_feedback(s)$
- exception: $exc := get\_task(n) ==$ None $\Rightarrow$ KeyError

add_task($\sout{task}$$name, deadline$)

- transition: $tasks := tasks.add(\sout{task}new\ Task(name, deadline))$

add_task($name, deadline, details$)

- transition: $tasks := tasks.add(new\ Task(name, deadline, details))$

add_task_from_file($n, name, deadline$)

- transition: $tasks := tasks.update(n, new\ Task(name, deadline))$

add_task_from_file($n, name, deadline, details$)

- transition: $tasks := tasks.update(n, new\ Task(name, deadline, details))$

rm_feedback($task\_id, feedback\_id$)

- transition: $tasks := get\_task(task\_id).rm\_feedback(feedback\_id)$
- exception: $exc := get\_task(task\_id) ==$ None $\Rightarrow$ KeyError

rm_task(n)

- transition: $tasks := tasks.remove(n)$

set_details(n, s)

- transition: $tasks := get\_task(n).set\_details(s)$
- exception: $exc := get\_task(n) ==$ None $\Rightarrow$ KeyError

ProjectList Module

Template Module

ProjectList is a Dict(Project)

Project Module

Module

Project

Uses

Sprint, Tasklist, Task

Syntax

Exported Constants

None

Exported Types

Project = ?

Exported Access Programs

Routine Name	In	Out	Exceptions
new Project	String	Project	
new Project	String, String	Project	
get_desc		String	
get_feedback	$\mathbb{N}, \mathbb{N}$	String	IndexError
get_last_meeting_id		$\mathbb{N}$	
get_meetings		seq of Meeting	
get_meeting_desc	$\mathbb{N}$	String	KeyError
get_meeting_name	$\mathbb{N}$	String	KeyError
get_name		String	
get_rqes		seq of String	
get_sprints		seq of Sprint	
get_task	$\mathbb{N}, \mathbb{N}$	Task	IndexError
get_tasks	$\mathbb{N}$	seq of Task	IndexError
set_desc	String		
add_feedback	$\mathbb{N}$, String		IndexError
add_meeting	~~Meeting~~ String, String, MeetingT		
add_meeting	String, String, MeetingT, String		
add_rqe	String		
add_sprint	~~Sprint~~		
add_sprint_from_file	Date		
add_task	String, String		IndexError
add_task	String, String, String		IndexError
add_task_from_file	$\mathbb{N}$, String, String		IndexError
add_task_from_file	$\mathbb{N}$, String, String, String		IndexError
rm_feedback	$\mathbb{N}, \mathbb{N}$		IndexError
rm_meeting	$\mathbb{N}$		KeyError
rm_rqe	$\mathbb{N}$		IndexError
rm_sprint			IndexError
rm_task	$\mathbb{N}$		IndexError
set_details	$\mathbb{N}$, String		IndexError
set_meeting_desc	$\mathbb{N}$		KeyError
set_meeting_desc	$\mathbb{N}$, String		KeyError
update_meeting	$\mathbb{N}$, String, String, MeetingT		
update_meeting	$\mathbb{N}$, String, String, MeetingT, String		

Semantics

State Variables

$name$: String
$desc$: String
$meetings$: MeetingList
$rqes$: seq of String
$sprints$: seq of Sprint
c: $\mathbb{N}$

State Invariant

$c = |sprints| \ \wedge \ c \geq 0$

Assumptions

- The Project constructor is called for each object instance before any other access routine is called for that object.

Access Routine Semantics

new Project(n)

- transition: $name, desc, rqes, sprints := n,\ \text{None},\ [],\ []$
- output: $out :=$ self

new Project(n, d)

- transition: $name, desc, rqes, sprints := n,\ d,\ [],\ []$
- output: $out :=$ self

get_desc()

- output: $out := (desc = \text{None} \Rightarrow \text{“No description”} \mid desc)$

get_feedback($sprint\_index, task\_index$)

- output: $out := sprints[sprint\_index].get_feedback(task\_index)$
- exception: $exc := sprint\_index > |sprints| - 1 \Rightarrow \text{IndexError}$

get_last_task_id()

- output: $out := sprints[-1].get\_last\_task\_id()$

get_meetings()

- output: $out := meetings.to_seq()$

get_meeting_desc(n)

- output: $out := meetings[n].get\_desc()$
- exception: $exc := meetings[n] == None \Rightarrow$ KeyError

get_meeting_name(n)

- output: $out := meetings[n].get\_name()$
- exception: $exc := meetings[n] == None \Rightarrow$ KeyError

get_name()

- output: $out := name$

get_rqes()

- output: $out := rqes$

get_sprints()

- output: $out := sprints$

get_task($sprint\_index, task\_index$)

- output: $out := sprints[sprint\_index].get\_task(task\_index)$
- exception: $exc := sprint\_index > |sprints| - 1 \Rightarrow$ IndexError

get_tasks(n)

- output: $out := sprints[n].get\_tasks()$
- exception: $exc := n > |sprints| - 1 \Rightarrow$ IndexError

set_desc(s)

- transition: $desc := s$

add_feedback(n, s)

- transition: $sprints := sprints[-1].add\_feedback(n, s)$
- exception: $exc := |sprints| == 0 \Rightarrow$ IndexError

add_meeting($\sout{meeting}name, dt, m\_type$)

- transition: $meetings := meetings.add(\sout{meeting}new\ Meeting(name, dt, m\_type))$

add_meeting$(name, dt, m_type, desc)$

- transition: $meetings := meetings.add(new\ Meeting(name, dt, m\_type, desc))$

add_rqe(s)

- transition: $rqes := rqes \mid\mid s$

add_sprint(~~$sprint$~~)

- transition: $sprints, c := sprints \mid\mid$ ~~$sprint$~~$new\ Sprint()$ $, c + 1$

add_sprint_from_file(d)

- transition: $sprints := sprints \mid\mid new\ Sprint(date)$

add_task$(name, deadline)$

- transition: $sprints := sprints[-1].add\_task(name, deadline)$
- exception: $exc := |sprints| == 0 \Rightarrow$ IndexError

add_task$(name, deadline, details)$

- transition: $sprints := sprints[-1].add\_task(name, deadline, details)$
- exception: $exc := |sprints| == 0 \Rightarrow$ IndexError

add_task_from_file$(n, name, deadline)$

- transition: $sprints := sprints[-1].add\_task\_from\_file(n, name, deadline)$
- exception: $exc := |sprints| == 0 \Rightarrow$ IndexError

add_task_from_file$(n, name, deadline, details)$

- transition: $sprints := sprints[-1].add\_task\_from\_file(n, name, deadline, details)$
- exception: $exc := |sprints| == 0 \Rightarrow$ IndexError

rm_feedback$(task_index, feedback_index)$

- transition: $sprints := sprints[-1].rm\_feedback(task\_index, feedback\_index)$
- exception: $exc := |sprints| == 0 \Rightarrow$ IndexError

rm_meeting(n)

- transition: $meetings := meetings.remove(n)$
- exception: $exc := meetings[n] == None \Rightarrow$ KeyError

rm_rqe(n)

- transition: $rqes := rqes - rqes[n]$

rm_sprint()

- transition: $sprints, c := sprints[0 : |sprints| - 2], c - 1$
- exception: $exc := c = 0 \Rightarrow$ IndexError

rm_task(n)

- transition: $sprints := sprints[-1].rm\_task(n)$
- exception: $exc := |sprints| == 0 \Rightarrow$ IndexError

set_details(n, s)

- transition: $sprints := sprints[-1].set\_details(n, s)$
- exception: $exc := |sprints| == 0 \Rightarrow$ IndexError

set_meeting_desc(n)

- transition: $meetings := meetings[n].set\_desc(None)$
- exception: $exc := meetings[n] == None \Rightarrow$ KeyError

set_meeting_desc(n, s)

- transition: $meetings := meetings[n].set\_desc(s)$
- exception: $exc := meetings[n] == None \Rightarrow$ KeyError

update_meeting($id, name, dt, m\_type$)

- transition: $meetings := meetings.update(id, New\ Meeting(id, name, dt, m\_type)$

update_meeting($id, name, dt, m\_type, desc$)

- transition: $meetings := meetings.update(id, New Meeting(id, name, dt, m\_type, desc)$

File Input/Output Module

Module

FileIO

Uses

Project, ProjectList

Syntax

Exported Constants

N/A

Exported Types

None

Exported Access Programs

Routine Name	In	Out	Exceptions
read		ProjectList	
write	$\mathbb{N}$, String, String		
create	$\mathbb{N}$, String, String		
delete	$\mathbb{N}$		

Semantics

Environment Variables

PATH = the path to the src/data directory

State Variables

None

State Invariant

None

Assumptions

- The FileIO module is a helper module for Scrumbot, dealing with reading and writing to a file. There is assumed that the inputs are strictly defined and given only by Scrumbot.
- It is also assumed that create and delete are only called by add_project() and rm_project respectively.
- The task variable in write() should always be a string that matches the function name given.
- Exceptions are considered unnecessary in this module as all error testing of invalid inputs and cases are tested in the Scrumbot module before running this module.

Access Routine Semantics

read()

- output: a ProjectList created from a list of files under a directory data found in the src folder. Each file would be its individual project, and the data within each folder would contain the information regarding the meetings, sprints, tasks and their components.

write($id, task, info$)

- transition: writes to the specific project file and inserts or removes information to the file based on the task called.

create($id, name, desc$)

- transition: creates a new file in src/data/ where the file name is the id of the project.

delete(id)

- transition: removes a file from the src/data/ directory where the id matches the file name.2