

Module Interface Specification for Software Eng 4G06

Team 2, Parnas' Pals

William Lee

Jared Bentvelsen

Bassel Rezkalla

Yuvraj Randhawa

Dimitri Tsampiras

Matthew McCracken

January 18, 2023

1 Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at <https://github.com/dimitritsampiras/olympian/tree/main/docs/SRS>.

Contents

1	Revision History	i
2	Symbols, Abbreviations and Acronyms	ii
3	Introduction	1
4	Notation	1
5	Module Decomposition	1
6	MIS of Project Olympian	3
7	Fitness Unit Of Measurement Type Module	3
7.1	Uses	3
7.2	Syntax	3
7.2.1	Exported Types	3
7.2.2	Exported Access Programs	3
7.3	Semantics	3
7.3.1	State Variables	3
7.3.2	Environment Variables	3
7.3.3	Assumptions	3
7.3.4	Access Routine Semantics	3
7.3.5	Local Functions	3
8	Quantifier Module	4
8.1	Uses	4
8.2	Syntax	4
8.2.1	Exported Constants	4
8.2.2	Exported Access Programs	4
8.3	Semantics	4
8.3.1	State Variables	4
8.3.2	Environment Variables	4
8.3.3	Assumptions	4
8.3.4	Access Routine Semantics	4
8.3.5	Local Functions	5
9	Exercise Module	6
9.1	Uses	6
9.2	Syntax	6
9.2.1	Exported Constants	6
9.2.2	Exported Access Programs	6
9.3	Semantics	6

9.3.1	State Variables	6
9.3.2	Environment Variables	6
9.3.3	Assumptions	6
9.3.4	Access Routine Semantics	6
9.3.5	Local Functions	7
10	Timed Sequence Module	8
10.1	Uses	8
10.2	Syntax	8
10.2.1	Exported Constants	8
10.2.2	Exported Access Programs	8
10.3	Semantics	8
10.3.1	State Variables	8
10.3.2	Environment Variables	8
10.3.3	Assumptions	8
10.3.4	Access Routine Semantics	8
10.3.5	Local Functions	9
11	Database Communicator Module	10
11.1	Uses	10
11.2	Syntax	10
11.2.1	Exported Constants	10
11.2.2	Exported Access Programs	10
11.3	Semantics	10
11.3.1	State Variables	10
11.3.2	Environment Variables	10
11.3.3	Assumptions	10
11.3.4	Access Routine Semantics	10
11.3.5	Local Functions	11
12	Workout Module	12
12.1	Uses	12
12.2	Syntax	12
12.2.1	Exported Constants	12
12.2.2	Exported Access Programs	12
12.3	Semantics	12
12.3.1	State Variables	12
12.3.2	Environment Variables	12
12.3.3	Assumptions	12
12.3.4	Access Routine Semantics	12
12.3.5	Local Functions	13

13 Workout Routine Module	14
13.1 Uses	14
13.2 Syntax	14
13.2.1 Exported Constants	14
13.2.2 Exported Access Programs	14
13.3 Semantics	14
13.3.1 State Variables	14
13.3.2 Environment Variables	14
13.3.3 Assumptions	14
13.3.4 Access Routine Semantics	14
13.3.5 Local Functions	15
14 User Profile Module	16
14.1 Uses	16
14.2 Syntax	16
14.2.1 Exported Constants	16
14.2.2 Exported Access Programs	16
14.3 Semantics	16
14.3.1 State Variables	16
14.3.2 Environment Variables	16
14.3.3 Assumptions	17
14.3.4 Access Routine Semantics	17
14.3.5 Local Functions	18
15 User Login Module	19
15.1 Uses	19
15.2 Syntax	19
15.2.1 Exported Constants	19
15.2.2 Exported Access Programs	19
15.3 Semantics	19
15.3.1 State Variables	19
15.3.2 Environment Variables	19
15.3.3 Assumptions	19
15.3.4 Access Routine Semantics	19
15.3.5 Local Functions	20
16 User Registration Module	21
16.1 Uses	21
16.2 Syntax	21
16.2.1 Exported Constants	21
16.2.2 Exported Access Programs	21
16.3 Semantics	21
16.3.1 State Variables	21

16.3.2	Environment Variables	21
16.3.3	Assumptions	21
16.3.4	Access Routine Semantics	21
16.3.5	Local Functions	22
17	User Fitness Goal Module	22
17.1	Uses	22
17.2	Syntax	22
17.2.1	Exported Constants	22
17.2.2	Exported Access Programs	22
17.3	Semantics	23
17.3.1	State Variables	23
17.3.2	Environment Variables	23
17.3.3	Assumptions	23
17.3.4	Access Routine Semantics	23
17.3.5	Local Functions	24
18	Workout Browsing Module	25
18.1	Uses	25
18.2	Syntax	25
18.2.1	Exported Constants	25
18.2.2	Exported Access Programs	25
18.3	Semantics	25
18.3.1	State Variables	25
18.3.2	Environment Variables	25
18.3.3	Assumptions	25
18.3.4	Access Routine Semantics	25
18.3.5	Local Functions	26
19	Creator Module	27
19.1	Uses	27
19.2	Syntax	27
19.2.1	Exported Constants	27
19.2.2	Exported Access Programs	27
19.3	Semantics	27
19.3.1	State Variables	27
19.3.2	Environment Variables	27
19.3.3	Assumptions	27
19.3.4	Access Routine Semantics	27
19.3.5	Local Functions	27

20 Exercise Creation Module	28
20.1 Uses	28
20.2 Syntax	28
20.2.1 Exported Constants	28
20.2.2 Exported Access Programs	28
20.3 Semantics	28
20.3.1 State Variables	28
20.3.2 Environment Variables	28
20.3.3 Assumptions	28
20.3.4 Access Routine Semantics	28
20.3.5 Local Functions	28
21 Workout Creation Module	29
21.1 Uses	29
21.2 Syntax	29
21.2.1 Exported Constants	29
21.2.2 Exported Access Programs	29
21.3 Semantics	29
21.3.1 State Variables	29
21.3.2 Environment Variables	29
21.3.3 Assumptions	29
21.3.4 Access Routine Semantics	29
21.3.5 Local Functions	29
22 Workout Routine Creation Module	30
22.1 Uses	30
22.2 Syntax	30
22.2.1 Exported Constants	30
22.2.2 Exported Access Programs	30
22.3 Semantics	30
22.3.1 State Variables	30
22.3.2 Environment Variables	30
22.3.3 Assumptions	30
22.3.4 Access Routine Semantics	30
22.3.5 Local Functions	30

3 Introduction

The following document details the Module Interface Specifications for Olympian: a social workout platform that unites beginner, intermediate, and experienced athletes and fitness enthusiasts. Users are able to track, post, interact with, discover, and utilize workouts and programs that are all user generated.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at <https://github.com/dimitritsampirass/olympian>. Note that actual code is contained on the ‘dev’ branch.

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol $:=$ is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | \dots | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by Software Eng 4G06.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	\mathbb{Z}	a number without a fractional component in $(-\infty, \infty)$
natural number	\mathbb{N}	a number without a fractional component in $[1, \infty)$
real	\mathbb{R}	any number in $(-\infty, \infty)$

The specification of Software Eng 4G06 uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, Software Eng 4G06 uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	Level 2
Hardware-Hiding Module	
	exercise
	workout
	workout routine
	user login
	user registration
	user profile
Behaviour-Hiding Module	user fitness goal
	workout browser
	creation
	workout creation
	workout routine creation
	exercise creation
	timed sequence
	DatabaseCommunicator
Software Decision Module	quantifier

Table 1: Module Hierarchy

6 MIS of Project Olympian

7 Finess Unit Of Measurement Type Module

7.1 Uses

7.2 Syntax

7.2.1 Exported Types

FinessUnit = {Set, Rep, Curl, Press, Jump, Step, Stretch, Push, Pull, Second, Minute, Hour, Day...}
//Many more, just stating a few

7.2.2 Exported Access Programs

None

7.3 Semantics

7.3.1 State Variables

None

7.3.2 Environment Variables

None

7.3.3 Assumptions

None

7.3.4 Access Routine Semantics

None

7.3.5 Local Functions

8 Quantifier Module

8.1 Uses

8.2 Syntax

8.2.1 Exported Constants

Quantifier = ?

8.2.2 Exported Access Programs

Name	In	Out	Exceptions
Quantifier	<i>FitnessUnit</i> , \mathbb{Q}	Quantifier	
getUnit		<i>FitnessUnit</i>	
getValue		\mathbb{Q}	

8.3 Semantics

8.3.1 State Variables

unit: *FitnessUnit*

value: \mathbb{Q}

8.3.2 Environment Variables

8.3.3 Assumptions

None

8.3.4 Access Routine Semantics

Quantifier(u, v):

- transition: *unit, value* := *u, v*

- output: *out* := *self*

- exception: None

unit():

- output: *out* := *unit*

- exception: None

value():

- output: *out* := *value*

- exception: None

8.3.5 Local Functions

9 Exercise Module

9.1 Uses

N/A

9.2 Syntax

9.2.1 Exported Constants

Exercise = ?

9.2.2 Exported Access Programs

Name	In	Out	Exceptions
Exercise	Quantifier, String	Exercise	
getQuantifier		Quantifier	
getDescription		String	

9.3 Semantics

9.3.1 State Variables

quantifier: *Quantifier*

description: *String*

9.3.2 Environment Variables

9.3.3 Assumptions

None

9.3.4 Access Routine Semantics

Exercise(q, d):

- transition: *quantifier, description := q, d*
- output: *out := self*
- exception: None

quantifier():

- output: *out := quantifier*
- exception: None

description():

- output: $out := description$
- exception: None

9.3.5 Local Functions

10 Timed Sequence Module

10.1 Uses

10.2 Syntax

10.2.1 Exported Constants

TimedSequence(T) = ?

10.2.2 Exported Access Programs

Name	In	Out	Exceptions
TimedSequence	seq of T, String, Time	TimedSequence	
getSeq		seq of Exercise	
getDescription		String	
getDuration		Time	
add	T		

10.3 Semantics

10.3.1 State Variables

sequence: $T[]$
description: *String*
duration: *Time*

10.3.2 Environment Variables

None

10.3.3 Assumptions

None

10.3.4 Access Routine Semantics

TimedSequence(s, desc, dur):

- transition: $sequence, description, duration := s, desc, dur$
- output: $out := self$
- exception: None

sequence():

- output: *out* := *sequence*

- exception: None

description():

- output: *out* := *description*

- exception: None

duration():

- output: *out* := *duration*

- exception: None

add(t):

- transition: *sequence.put(t)*

- exception: None

10.3.5 Local Functions

11 Database Communicator Module

11.1 Uses

11.2 Syntax

11.2.1 Exported Constants

DatabaseCommunicator = ?

11.2.2 Exported Access Programs

Name	In	Out	Exceptions
DatabaseCommunicator	String	DatabaseCommunicator	
getExercise	String	seq of Exercise	
getWorkout	String	seq of Workout	
getRoutine	String	seq of Routine	
getUser	String	seq of User	
addExercise	Exercise		
addWorkout	Workout		
addRoutine	Routine		
addUser	User		

11.3 Semantics

11.3.1 State Variables

databaseURL: *String*

db: *Database*

11.3.2 Environment Variables

databaseKey: *String*

11.3.3 Assumptions

None

11.3.4 Access Routine Semantics

DatabaseCommunicator(URL):

- transition: $databaseURL, database := URL, database.connect(URL, databaseKey)$
- output: $out := self$
- exception: None

getExercise(query):

- output: *out* := *db.exercises.select(query)*
- exception: None

getWorkout(query):

- output: *out* := *db.workouts.select(query)*
- exception: None

getRoutine(query):

- output: *out* := *db.routines.select(query)*
- exception: None

getUser(query):

- output: *out* := *db.users.select(query)*
- exception: None

addExercise(e):

- output: *out* := *db.exercises.insert(e)*
- exception: None

addWorkout(w):

- output: *out* := *db.workouts.insert(w)*
- exception: None

addRoutine(r):

- output: *out* := *db.routines.insert(r)*
- exception: None

addUser(u):

- output: *out* := *db.users.insert(u)*
- exception: None

11.3.5 Local Functions

12 Workout Module

12.1 Uses

TimedSequence(Exercise)

12.2 Syntax

12.2.1 Exported Constants

Workout = ?

12.2.2 Exported Access Programs

Name	In	Out	Exceptions
Workout	seq of Exercise, String, Time	Workout	
getExercises		seq of Exercise	
getDescription		String	
getDuration		Time	
addExercise	Exercise		

12.3 Semantics

12.3.1 State Variables

exercises: *Exercise*[]

description: *String*

duration: *Time*

12.3.2 Environment Variables

None

12.3.3 Assumptions

None

12.3.4 Access Routine Semantics

Workout(e, desc, dur):

- transition: *exercises, description, duration* := *e, desc, dur*
- output: *out* := *self*

- exception: None

exercise():

- output: *out := exercise*

- exception: None

description():

- output: *out := description*

- exception: None

duration():

- output: *out := duration*

- exception: None

AddExercise(e):

- transition: *exercises.put(e)*

- exception: None

12.3.5 Local Functions

13 Workout Routine Module

13.1 Uses

TimedSequence(Workout)

13.2 Syntax

13.2.1 Exported Constants

Routine = ?

//Note: Same as a Workout Routine, just short version

13.2.2 Exported Access Programs

Name	In	Out	Exceptions
Routine	seq of Workout, Duration, Description	Routine	
getWorkouts		seq of Workout	
getDescription		String	
getDuration		Time	

13.3 Semantics

13.3.1 State Variables

workouts: *Exercise*[]

description: *String*

duration: *Time*

13.3.2 Environment Variables

None

13.3.3 Assumptions

None

13.3.4 Access Routine Semantics

Routine(w, desc, dur):

- transition: *workouts, description, duration* := *w, desc, dur*
- output: *out* := *self*

- exception: None

exercise():

- output: $out := exercise$

- exception: None

description():

- output: $out := description$

- exception: None

duration():

- output: $out := duration$

- exception: None

AddWorkout(w):

- transition: $workouts.put(w)$

- exception: None

13.3.5 Local Functions

14 User Profile Module

14.1 Uses

14.2 Syntax

14.2.1 Exported Constants

User = ?

14.2.2 Exported Access Programs

Name	In	Out	Exceptions
User	String, String, String, String	User	
username		String	
password		String	
email		String	
nickname		String	
fitness Goals		seq of Fitness- Goal	
created Workouts		seq of Workout	
created Routines		seq of Routine	
saved Workouts		seq of Workout	
saved Routines		seq of Routine	

14.3 Semantics

14.3.1 State Variables

username: *String*

password: *String*

email: *String*

nickname: *String*

fitnessGoals: *seq of FitnessGoal*

created Workouts: *Workout*[]

created Routines: *Routine*[]

saved Workouts: *Workout*[]

saved Routines: *Routine*[]

14.3.2 Environment Variables

None

14.3.3 Assumptions

None

14.3.4 Access Routine Semantics

User(username, pass, email, nickname):

- transition: *username, password, email, nickname := username, pass, email, nickname*
- output: *out := self*
- exception: None

username():

- output: *out := username*
- exception: None

email():

- output: *out := email*
- exception: `PermissionException`

password():

- output: *out := password*
- exception: `PermissionException`

nickname():

- output: *out := nickname*
- exception: None

fitnessGoals():

- output: *out := fitnessGoals*
- exception: None

createdWorkouts():

- output: *out := createdWorkouts*
- exception: None

createdRoutines():

- output: *out := createdRoutines*

- exception: None

savedWorkouts():

- output: *out := savedWorkouts*

- exception: None

savedRoutines():

- output: *out := savedRoutines*

- exception: None

createGoal(g):

- transition: *fitnessGoals.put(g)*

- exception: None

createWorkout(workout):

- transition: *createdWorkouts.put(w), savedWorkouts.put(w)*

- exception: None

createRoutine(routine):

- transition: *createdRoutine.put(r), savedRoutines.put(r)*

- exception: None

saveWorkout(w):

- transition: *savedWorkouts.put(w)*

- exception: None

saveRoutine(r):

- transition: *savedRoutines.put(r)*

- exception: None

14.3.5 Local Functions

15 User Login Module

15.1 Uses

15.2 Syntax

15.2.1 Exported Constants

UserLogin = ?

15.2.2 Exported Access Programs

Name	In	Out	Exceptions
UserLogin		UserLogin	
attemptLoginString	String	\mathbb{B}	tooManyAttempts
validateUser	String, String	\mathbb{B}	

15.3 Semantics

15.3.1 State Variables

attempts: \mathbb{Z}

maxAttempts: \mathbb{Z}

userDatabase: *Database*

15.3.2 Environment Variables

None

15.3.3 Assumptions

There exists a stored number for maximum attempts.

15.3.4 Access Routine Semantics

UserLogin(db):

- transition: *userDatabase*, *attempts* := *DatabaseCommunicator.users*, 0
- output: *out* := *self*
- exception: None

attemptLogin(user, pass):

- transition: *attempts* := *attempts* + 1

- output: $out := validateUser(user, pass) \wedge userDatabase.getUser(user) = (pass)$
- exception: $exc := attempts \geq maxAttempts \Rightarrow tooManyAttempts$

15.3.5 Local Functions

$validateUser(user, pass) \rightarrow \mathbb{B}$

- output: $out := userDatabase.validate(user, pass)$
- exception: None

16 User Registration Module

16.1 Uses

16.2 Syntax

16.2.1 Exported Constants

userRegistration = ?

16.2.2 Exported Access Programs

Name	In	Out	Exceptions
userRegistration		userRegistration	
register	String, String, String, String	String	
exists	String, String	\mathbb{B}	
validate	String, String, String, String	\mathbb{B}	invalidPassword, invalidUsername, invalidEmail

16.3 Semantics

16.3.1 State Variables

userDatabase: *Database*

16.3.2 Environment Variables

None

16.3.3 Assumptions

Regex operations are allowed on strings

16.3.4 Access Routine Semantics

userRegistration(db):

- transition: $userDatabase := DatabaseCommunicator.db.users$
- output: $out := self$
- exception: None

register(username, password, email, nickname):

- output: $out := \neg exists(username, email) \wedge validate(username, password, email, nickname)$
- exception: None

16.3.5 Local Functions

$exists(username, email) \rightarrow \mathbb{B}$:

- output: $out := userDatabase.existsUser(username) \vee userDatabase.existsEmail(email)$
- exception: None

$validate(username, email, password, nickname) \rightarrow \mathbb{B}$:

- output: $out := username.matches(\{6, 20\})$
 $\wedge password.matches(\wedge(? = . * \backslash d)(? = . * [a - z])(? = . * [A - Z]).\{6, 20\}\$)$
 $\wedge email.matches(/ \wedge \backslash w + ([\backslash . -]? \backslash w +) * @ \backslash w + ([\backslash . -]? \backslash w +) * (\backslash . \backslash w 2, 3) + \$ /)$
 $\wedge nickname.len() \geq 1$
- exception: $exc := \neg username.matches(\{6, 20\}) \Rightarrow invalidUsername$
- exception: $exc := \neg email.matches(/ \wedge \backslash w + ([\backslash . -]? \backslash w +) * @ \backslash w + ([\backslash . -]? \backslash w +) * (\backslash . \backslash w 2, 3) + \$ /) \Rightarrow invalidEmail$
- exception: $exc := \neg password.matches(\wedge(? = . * \backslash d)(? = . * [a - z])(? = . * [A - Z]).\{6, 20\}\$) \Rightarrow invalidPassword$
- exception: $exc := nickname.len() < 1 \Rightarrow invalidNickname$

17 User Fitness Goal Module

17.1 Uses

17.2 Syntax

17.2.1 Exported Constants

FitnessGoal = ?

17.2.2 Exported Access Programs

Name	In	Out	Exceptions
FitnessGoal	String, Quantifier	FitnessGoal	
quantifier		Quantifier	
description		String	
currentProgress		Quantifier	
status		\mathbb{B}	
progressGoal	Quantifier		

17.3 Semantics

17.3.1 State Variables

quantifier: *Quantifier*

description: *String*

progress: *Quantifier*

17.3.2 Environment Variables

None

17.3.3 Assumptions

None

17.3.4 Access Routine Semantics

FitnessGoal(quant, desc):

- transition: *quantifier, description, progression = quant, desc, Quantifier(quant.unit, 0)*
- output: *out := self*
- exception: None

quantifier():

- output: *out := quantifier*
- exception: None

description():

- output: *out := description*
- exception: None

currentProgress():

- output: *out := progress*
- exception: None

status():

- output: *out := progress.value < quantifier.value*
- exception: None

progressGoal(val):

- transition: *progress.value := val*
- exception: None

17.3.5 Local Functions

18 Workout Browsing Module

18.1 Uses

18.2 Syntax

18.2.1 Exported Constants

Browser = ?

18.2.2 Exported Access Programs

Name	In	Out	Exceptions
Browser		Browser	
workouts	String		
routines	String		

18.3 Semantics

18.3.1 State Variables

displayedRoutines: *Routine*
displayedWorkouts: *Workout*
workoutDatabase: *Database*
routineDatabase: *Database*

18.3.2 Environment Variables

None

18.3.3 Assumptions

None

18.3.4 Access Routine Semantics

Browser(db):

- transition: *workoutDatabase, routineDatabase := DatabaseCommunicator.db.workouts, DatabaseCommunicator.db.routines*
- output: *out := self*
- exception: None

routines(query):

- transition: *displayedRoutines := routineDatabase.select(query)*

- exception: None

workouts(query):

- transition: *displayedWorkouts := workoutDatabase.select(query)*
- exception: None

18.3.5 Local Functions

19 Creator Module

Creator(T) interface

19.1 Uses

19.2 Syntax

19.2.1 Exported Constants

Creator(T) = ?

19.2.2 Exported Access Programs

Name	In	Out	Exceptions
Creator		Creator	
create	Any	T	

19.3 Semantics

19.3.1 State Variables

None

19.3.2 Environment Variables

None

19.3.3 Assumptions

None

19.3.4 Access Routine Semantics

Creator():

- output: $out := self$
- exception: None

create():

- output: $out := T$
- exception: None

19.3.5 Local Functions

None

20 Exercise Creation Module

20.1 Uses

Creator(Exercise)

20.2 Syntax

20.2.1 Exported Constants

ExerciseCreator = ?

20.2.2 Exported Access Programs

Name	In	Out	Exceptions
ExerciseCreator		ExerciseCreator	
create	Any	Exercise	

20.3 Semantics

20.3.1 State Variables

None

20.3.2 Environment Variables

None

20.3.3 Assumptions

None

20.3.4 Access Routine Semantics

exerciseCreator():

- output: $out := self$
- exception: None

create():

- output: $out := Exercise$
- exception: None

20.3.5 Local Functions

21 Workout Creation Module

21.1 Uses

Creator(Workout)

21.2 Syntax

21.2.1 Exported Constants

WorkoutCreator = ?

21.2.2 Exported Access Programs

Name	In	Out	Exceptions
WorkoutCreator		WorkoutCreator	
create	Any	Workout	

21.3 Semantics

21.3.1 State Variables

None

21.3.2 Environment Variables

None

21.3.3 Assumptions

None

21.3.4 Access Routine Semantics

workoutCreator():

- output: $out := self$
- exception: None

create():

- output: $out := Workout$
- exception: None

21.3.5 Local Functions

22 Workout Routine Creation Module

22.1 Uses

Creator(Routine)

22.2 Syntax

22.2.1 Exported Constants

RoutineCreator = ?

22.2.2 Exported Access Programs

Name	In	Out	Exceptions
RoutineCreator		RoutineCreator	
create	Any	Routine	

22.3 Semantics

22.3.1 State Variables

None

22.3.2 Environment Variables

None

22.3.3 Assumptions

None

22.3.4 Access Routine Semantics

RoutineCreator():

- output: $out := self$
- exception: None

create():

- output: $out := Routine$
- exception: None

22.3.5 Local Functions

References

- Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.
- Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL <http://citeseer.ist.psu.edu/428727.html>.