

# University of Canberra Software Technology 2 2023 Semester 1

## Assignment 2

This version: Updated 13/4/2023 – 9:12am

### Contents

Authors notes: .....	4
TASK1 – Detail .....	5
TASK2 – Detail .....	6
TASK3 – Detail .....	7
TASK4 – Detail .....	9
TASK5 – Detail .....	11
TASK 6 - detail .....	13
Submission: .....	13
SAIL2023 – Language Details.....	15
Helpful Hints.....	18
Getting Started .....	19

This assignment is worth 30 Marks towards your final assessment.

DUE: Midnight 5<sup>th</sup> of May (Midnight Friday last week of term) (please watch announcements for possible extensions).

The idea is to build a language interpreter – albeit a very simple one.

I have named this trivial computing language SAIL2023 standing for Simple Assignment Interpreted Language 2023

This assignment is 5 different tasks which build on each other:

- Task 1 (11 marks) The first task is to create a simple interpreter with a small list of instructions (see detail in Task 1 section).
- Task 2 (4 marks) The second task is to add two new keywords (“multiply” and “divide”).
- Task 3 (4 marks) The 3<sup>rd</sup> task is to expand the language to include negative numbers, to allow “println” and “print” to print integers, and to clear the screen. In addition, undefined variables are now disallowed.
- Task 4 (3 marks) This is basically the DI level version – you need to add an “if” and “endif” statement with the following comparison operators “<”, “>”, “==”.
- Task 5 (3 marks) The 5<sup>th</sup> task is really optional for those wanting DI /HD grades. It’s to add a while and endwhile loop construct.
- Task 6 (5 marks) All students should do a Reflection report.

Note that each task has mandatory testing. This can be included in the reflection report.

Additional Mandatory Requirements:

- 1) The SAIL2023 program should be read into an array, ArrayList or other suitable structure and be executed from there.
- 2) I expect the code to compile in BlueJ; if it does not compile and run in that environment, you will not receive a good grade. I have been known to reduce marks to less than 50% for just this alone. I also expect to see screen shots of your mandatory testing; again I have been known to reduce marks to less than 50% for failing to provide evidence of your testing.
- 3) You must deal with lines starting with “/” as comments and ignore them. After reading in the lines you should give a count of lines read and non-comment lines read.

- 4) In Task3 and above Variables are created by the set command and if used in any other statement before being 'set' an undefined variable error should be generated.
- 5) Blank lines should be ignored (but they should be counted in the lines read in counter – see 3 above)
- 6) Variables must be stored in a suitable Java 'Dictionary Like' Class. I used Hashtable, but others can be used including parallel arrays.
- 7) You need a menu – I give you that code.
- 8) You need to print the name of the file executing – when you run it.
- 9) For tasks 1 to 3 whitespace is just space character(s) (to make it easier). For tasks 4 and 5 whitespace must include tab to allow formatting.

NOTE1: Tasks 4 and 5 are notably more difficult for just 6 marks, consider carefully if you wish to attempt them, less help will be given by tutors for these tasks.

NOTE2: You cannot get more than 30 for the assignment.

NOTE3: In all testing I (and the tutors) will always use lower case variables. This makes it slightly easier.

NOTE4: In my answer I do a 'syntax' check as I read in each line, this was not a good idea, though tokenising it may be a good idea (my option T1 in the supplied menu file was used to test the tokenizer and syntax checker).

NOTE5: tasks should be done in order Task1, Task2, then any others in any order, and finally Task6.

NOTE6: You need to clear the system between file reads, so you can run multiple files from the menu. Specifically you need to be able to run one file after the other without re-starting the program.

NOTE7: You should be able to pass the assignment just doing Task 1, Task 2 and Task 6 reasonably well and submitting working BlueJ code. If you are worried, pick one or two Task 3 options to get a couple of extra marks, two of them are supposed to be easy.

NOTE8: I provide a simple menu class and main program to help you – note that you do not need all the menu options provided – just delete or comment out the ones of no use to you.

#### Authors notes:

This is a step by step assignment designed to be done in order; Task1, Task2 .. After that you can pick and choose the tasks you do. Task3 is 4 sub parts you can do some and not others.

For the curious: This language would become 'Turing-complete' after Task 4, but not useful until Task 5.

In my answer, options E, E2, E3, E4, E5 are all identical and run the same method. The idea was that students would do E, for Task 1, E2 for Task 2, E3 for Task3 ... etc. But in my answer, I did all the tasks in one option. Students should code/use only the menu options that they need.

## TASK1 – Detail

The first task is to create a simple interpreter with a small list of instructions specifically:

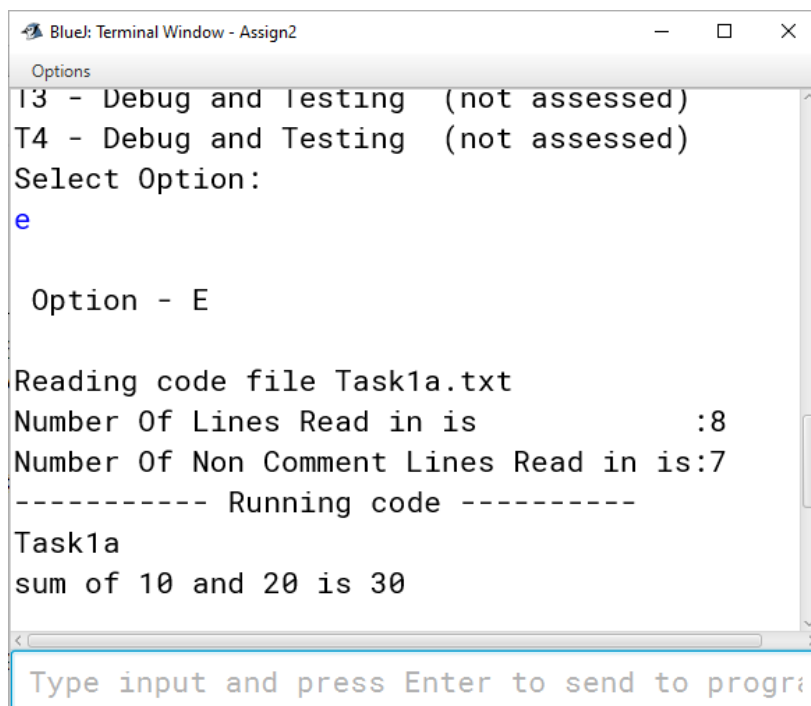
set <variable> to <integer>	
add <variable or integer> to <variable>	
subtract <variable or integer> from <variable>	
print <variable, literal>	*1 - Prints without a new line
println <variable, literal>	*1 - Prints then does a new line
End	*2 - Stops the program
/	Any line starting with / is a comment line
Blank line	Treated as a comment

Also Fill out the W option to show your name and student number (failure to do this will cost 1 mark).

Task 1 includes the basic menu, students can write their own – but I do provide a template.

The programs should display a count of the lines read in and a count of the non-comment (and non-blank) lines read in (see example below).

Sample run of Task1a.txt



```
BlueJ: Terminal Window - Assign2
Options
T3 - Debug and Testing (not assessed)
T4 - Debug and Testing (not assessed)
Select Option:
e

Option - E

Reading code file Task1a.txt
Number Of Lines Read in is      :8
Number Of Non Comment Lines Read in is:7
----- Running code -----
Task1a
sum of 10 and 20 is 30

Type input and press Enter to send to progr
```

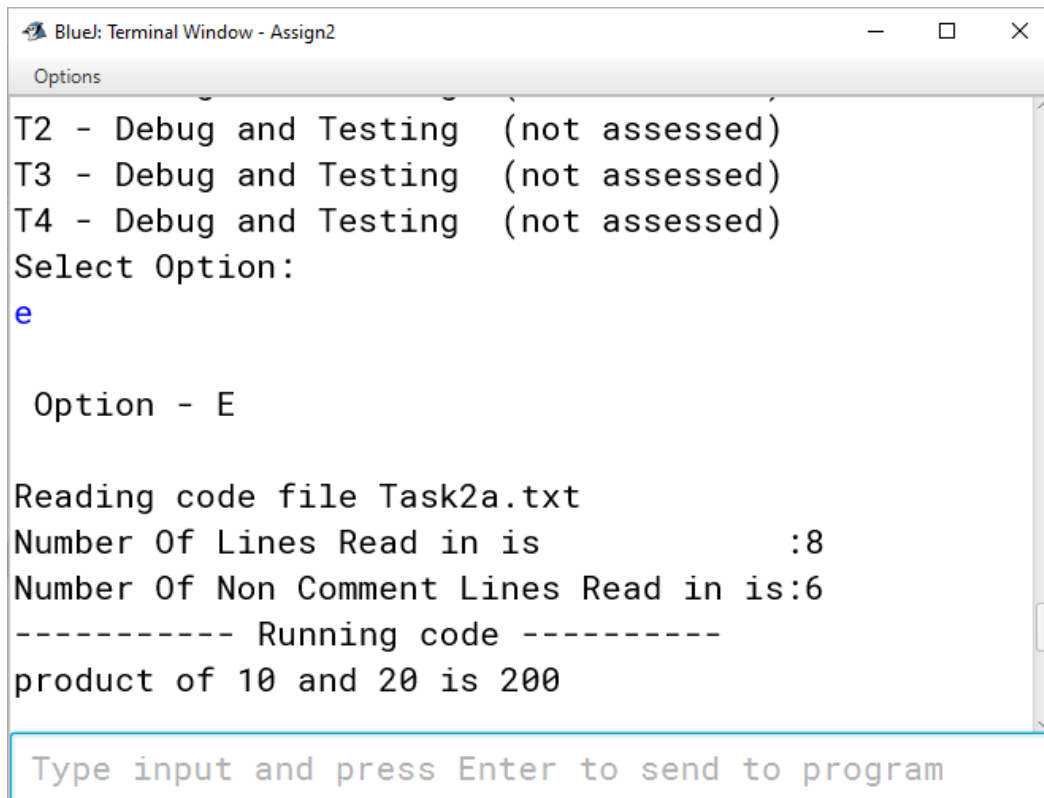
## TASK2 – Detail

The second task is to add two new keywords (“multiply” and “divide”), these are like the add and subtract commands except the variable is the 2nd token not the fourth token.

Obviously, they multiply and divide rather than add and subtract.

Statement	Notes
<b>Task 2</b>	
multiply <variable> by <variable or integer>	
divide <variable> by <variable or integer>	

Sample run of task2a.txt



```
Blue: Terminal Window - Assign2
Options
T2 - Debug and Testing (not assessed)
T3 - Debug and Testing (not assessed)
T4 - Debug and Testing (not assessed)
Select Option:
e

Option - E

Reading code file Task2a.txt
Number Of Lines Read in is          :8
Number Of Non Comment Lines Read in is:6
----- Running code -----
product of 10 and 20 is 200

Type input and press Enter to send to program
```

## TASK3 – Detail

The 3rd task is to expand the language to include 4 new features:

- a) negative numbers,
- b) allow “println” and “print” to print integers
- c) allow println and print to clear the screen using the cls keyword
- d) Also undefined variables are disallowed

You can pick and choose to do all or none of the above and still do Task4 and task5, basically each of the above is worth 1 mark.

To clarify d) above, the set command defines the variable.

So the sequence below should fail

```
add 3 to fred
println fred
```

The sequence below should succeed.

```
set fred to 0
add 3 to fred
println fred
```

Statement	Notes
<b>Task 3</b>	
print <variable, literal, cls or integer>	
println <variable, literal, cls or integer>	

Note: ‘print cls’ should clear the screen (same as System.out.print(“\f”);

Note: ‘print 3’ should print the integer 3 (same as System.out.print(3));

Note: ‘print -3’ should print the integer -3 (same as System.out.print(-3)); // if you do part a

Note: Negative numbers should also be available to the add, subtract, multiply and divide commands  
eg ‘add -3 to fred’

### Testing Task3

NOTE: Just test the Task3M files you implemented

NOTE: You can modify task3a.txt and Task3b.txt to remove features you did not implement for your testing.

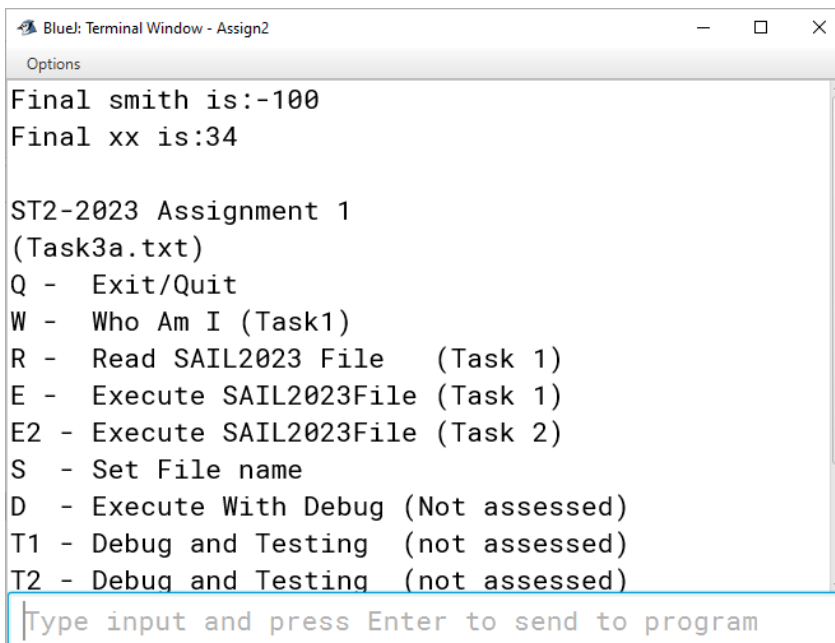
Task3Ma.txt tests negative numbers

Task3Mb.txt tests integer print

Task3Mc.txt tests print cls

Task3Md.txt tests undefined variables

Sample run of task3a.txt – note the brutal effect of the print cls command.



```
BlueJ: Terminal Window - Assign2
Options
Final smith is:-100
Final xx is:34

ST2-2023 Assignment 1
(Task3a.txt)
Q - Exit/Quit
W - Who Am I (Task1)
R - Read SAIL2023 File (Task 1)
E - Execute SAIL2023File (Task 1)
E2 - Execute SAIL2023File (Task 2)
S - Set File name
D - Execute With Debug (Not assessed)
T1 - Debug and Testing (not assessed)
T2 - Debug and Testing (not assessed)

Type input and press Enter to send to program
```



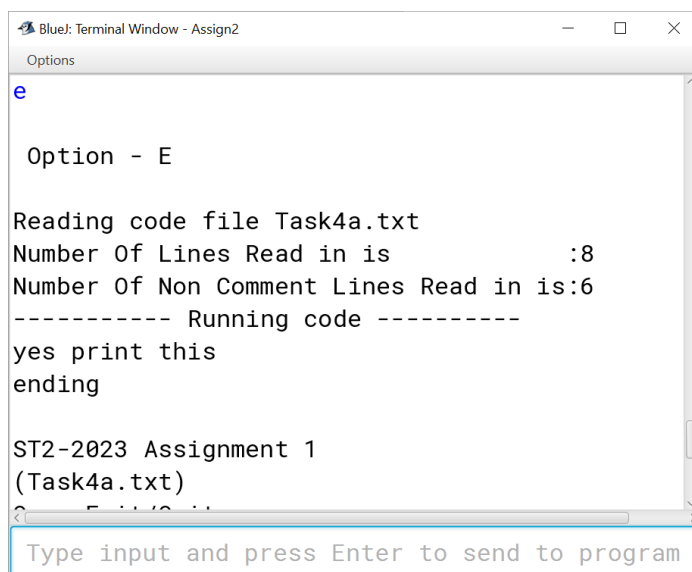
## TASK4 – Detail

This is basically the DI/HD level version – you need to add an “if” and “endif” statement with the following comparison operators “<”, “>”, “==”.

Statement	Notes
<b>Task 4</b>	
If <variable or integer> <operator> <variable or integer> then	*3 *4
endif	

HINT- test the condition if it’s false skip forward to endif – I don’t ever nest ifs in testing so that will work.

Example run of if (task4a.txt):



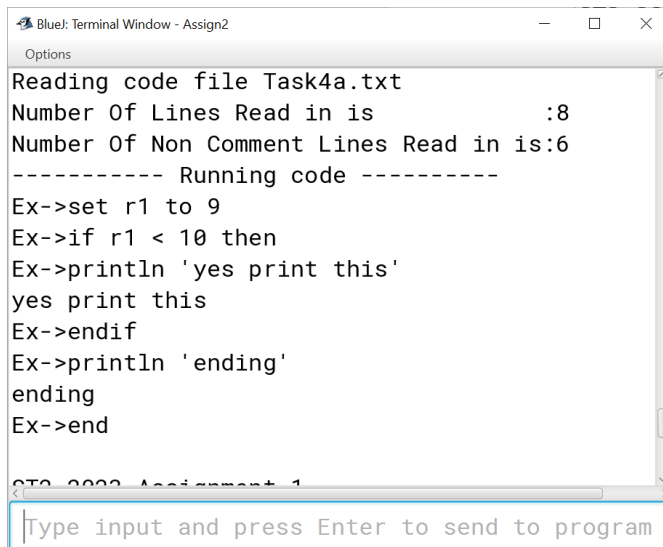
```
BlueJ: Terminal Window - Assign2
Options
e

Option - E

Reading code file Task4a.txt
Number Of Lines Read in is      :8
Number Of Non Comment Lines Read in is:6
----- Running code -----
yes print this
ending

ST2-2023 Assignment 1
(Task4a.txt)
C:\Program Files\Java\jdk-17\bin\java.exe -ea -Xmx1024m -Djava.awt.headless=true -jar C:\Program Files\Java\jdk-17\bin\bluej.jar C:\Program Files\Java\jdk-17\bin\bluej.jar C:\Program Files\Java\jdk-17\bin\bluej.jar
Type input and press Enter to send to program
```

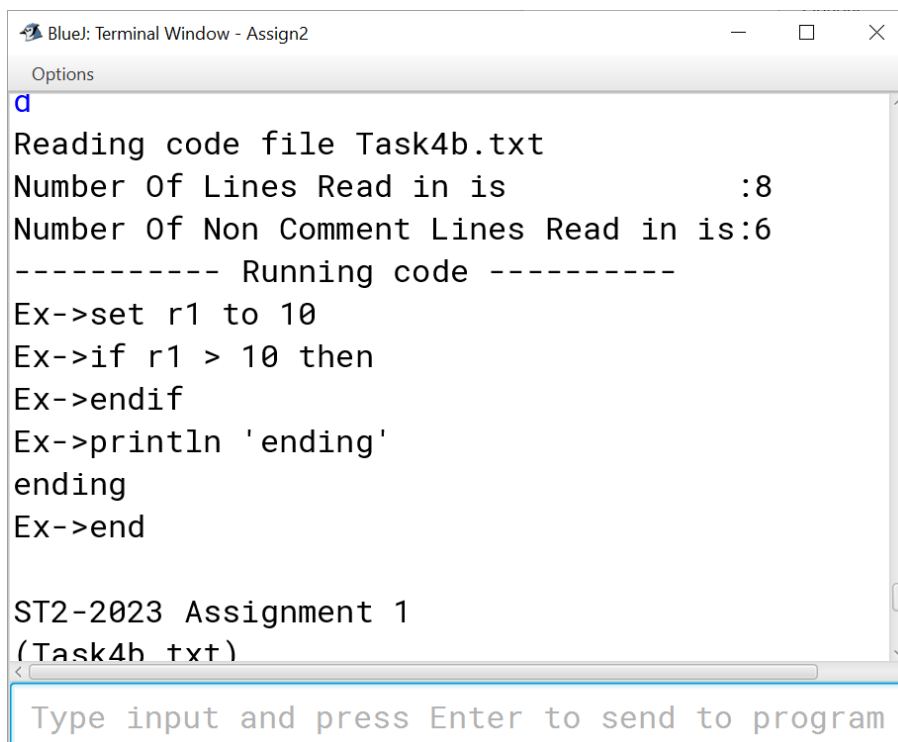
I think its helpful to see that with the ‘D’ or debug option, so here it is run with debug.



```
BlueJ: Terminal Window - Assign2
Options
Reading code file Task4a.txt
Number Of Lines Read in is      :8
Number Of Non Comment Lines Read in is:6
----- Running code -----
Ex->set r1 to 9
Ex->if r1 < 10 then
Ex->println 'yes print this'
yes print this
Ex->endif
Ex->println 'ending'
ending
Ex->end

ST2-2023 Assignment 1
Type input and press Enter to send to program
```

Finally a 3<sup>rd</sup> run with debug so you can see it skip forward:



```
BlueJ: Terminal Window - Assign2
Options
d
Reading code file Task4b.txt
Number Of Lines Read in is      :8
Number Of Non Comment Lines Read in is:6
----- Running code -----
Ex->set r1 to 10
Ex->if r1 > 10 then
Ex->endif
Ex->println 'ending'
ending
Ex->end

ST2-2023 Assignment 1
(Task4b.txt)
Type input and press Enter to send to program
```

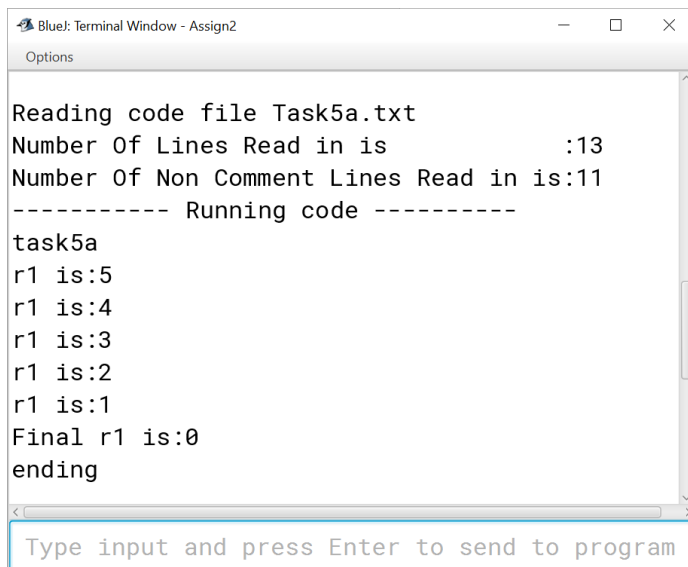
## TASK5 – Detail

You need to add while and endwhile statements to the interpreter:

Statement	Notes
<b>Task 5</b>	
While <variable or integer> <operator> <variable or integer> do	'do' is not optional Same operators as 'if' statement *4
endwhile	

Note this is a loop, and this part is for HD students so not so many hints

Sample run:



```
Blue: Terminal Window - Assign2
Options
Reading code file Task5a.txt
Number Of Lines Read in is      :13
Number Of Non Comment Lines Read in is:11
----- Running code -----
task5a
r1 is:5
r1 is:4
r1 is:3
r1 is:2
r1 is:1
Final r1 is:0
ending
Type input and press Enter to send to program
```

Sample Run with debug : (ug so long – so only partial)

```
Blue! Terminal Window - Assign2
Options
Number Of Lines Read in is :13
Number Of Non Comment Lines Read in is:11
----- Running code -----
Ex->println 'task5a'
task5a
Ex->set r1 to 5
Ex->while r1 > 0 do
Ex->print 'r1 is:'
r1 is:Ex->println r1
5
Ex->subtract 1 from r1
Ex->endwhile
Ex->while r1 > 0 do
Ex->print 'r1 is:'
r1 is:Ex->println r1
4
Ex->subtract 1 from r1
Ex->endwhile
Ex->while r1 > 0 do
Ex->print 'r1 is:'
r1 is:Ex->println r1
3
Ex->subtract 1 from r1
Type input and press Enter to send to program
```

## TASK 6 - detail

The reflection report:

You should indicate the following in this report:

1. The problems you faced and how you got around them. Please address:
  - How you chose to store the program.
  - What you learned.
  - What you found difficult.
  - What you would do differently.
2. You must identify any source code used that is not your own (except that code that I have supplied);
3. Remember the majority of the program should be your own work.

The reflection report should be a few paragraphs only, you may have some code in it that makes it longer than this; but try to keep it to a minimum. The report can contain screenshots of mandatory testing, or they can be in separate document(s).

Submission:

Submitted on Canvas as a single zip file that contains:

- Your reflection report.
- Your source code as a BlueJ solution.
- Screen shots of mandatory testing (which can be in the reflection report or as separate document(s)).

Your submission report should also contain screen shots (like the ones in here) for the following file runs. Just leave out the ones for tasks you did not do. Failure to do the mandatory testing will result in a very low grade often near 0:

<b>Mandatory Test File</b>	
Task1Ma.txt	Mandatory
Task1Mb.txt	Mandatory
Task1Mc.txt	Mandatory
Task2Ma.txt	Only if you attempted task2
Task3Ma.txt	Only do if you implement the tested feature
Task3Mb.txt	Only do if you implement the tested feature
Task3Mc.txt	Only do if you implement the tested feature
Task3Md.txt	Only do if you implement the tested feature
Task4Ma.txt	Only if you attempted task4
Task4Mb.txt	Only if you attempted task4
Task5Ma.txt	Only if you attempted task5
Task5Mb.txt	Only if you attempted task5
NOTE:	Some are designed to test your error handling and deliberately fail

## SAIL2023 – Language Details

The SAIL2023 language definition and sample code

### Simple Assignment Interpreted Language 2023

To make life easy all variables are integers in this simple language

Variable names start with a letter (a to z) and are lower case they can be up to 10 characters in length.

Integers are defined as one or more digits (in range 0-9). For Tasks 1 and 2 only positive integers are used. In Tasks 3 integers are defined as a leading ‘-’ sign followed by one or more digits (in range 0-9).

String constants are bounded by a single quote and used exclusively in print and println statements. For example, print ‘Hello All’

The words to, from, into, by, do, then and cls are keywords that are not statement leaders.

## List of statements

Statement	Notes
<b>Task 1</b>	
set <variable> to <variable or integer>	
add <variable or integer> to <variable>	
subtract <variable or integer> from <variable>	
print <variable or literal>	*1 - Prints without a new line
println <variable or literal>	*1 - Prints then does a new line
End	*2 - Stops the program
/	Any line starting with / is a comment
Blank line	Treated as a comment
<b>Task 2</b>	
multiply <variable> by <variable or integer>	
divide <variable> by <variable or integer>	
<b>Task 3</b>	
print <variable, literal, cls or integer>	
println <variable, literal, cls or integer>	
<b>Task 4</b>	
If <variable or integer> <operator> <variable or integer> then	*3 *4
endif	
<b>Task 5</b>	
While <variable or integer> <operator> <variable or integer> do	*4 *2
endwhile	

\*1 – Print and println have the same syntax, note that for Tasks 1 and 2 the syntax is only print <variable or stringLiteral>. Note also that cls is a keyword.

\*2 the “do” is not optional. Operators are “<”, “>” and “==”

\*3 the “then” is not optional. Operators are “<”, “>” and “==”

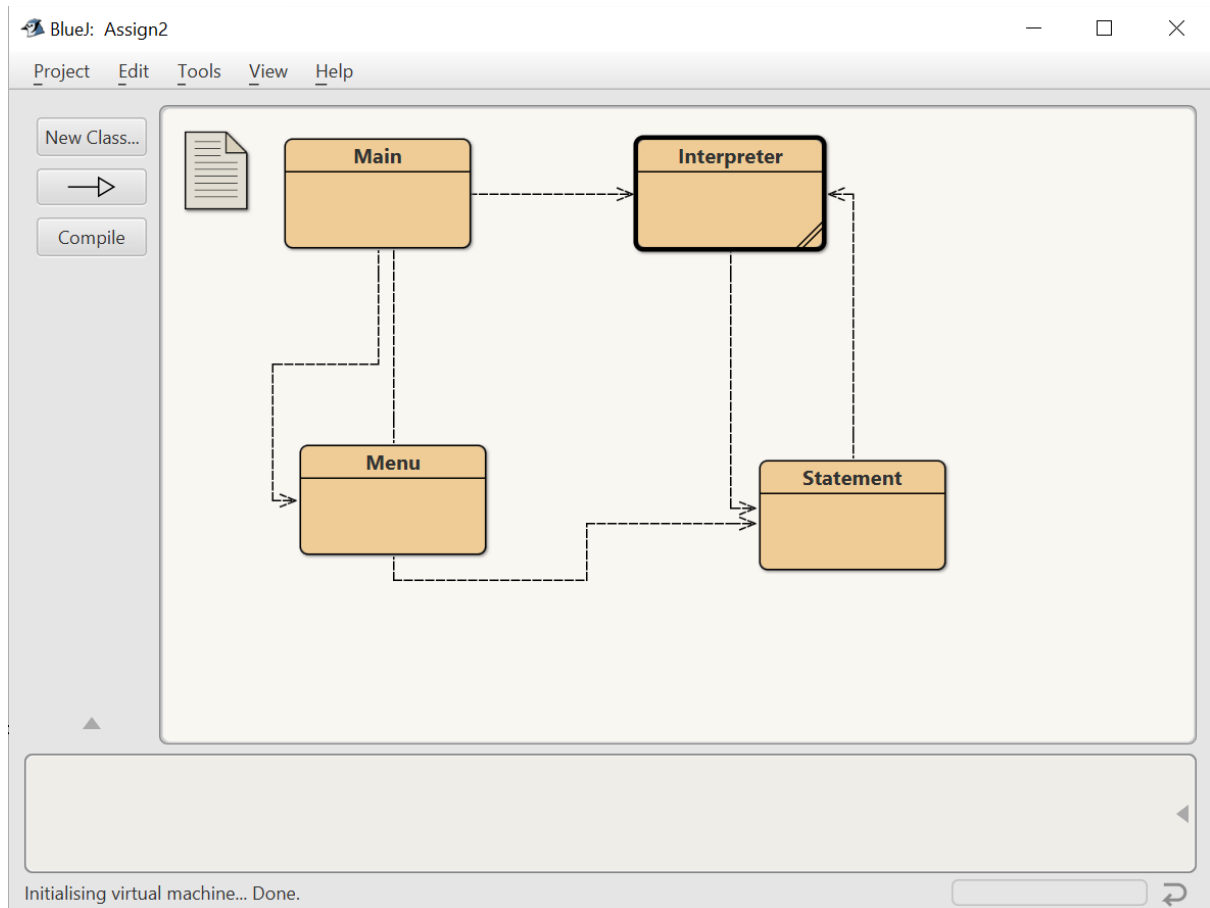
\*4 there is no need to test or implement nested if or while, it’s difficult enough without nesting.



\*5 the minor keywords 'then', 'do', 'from', 'to', and 'into' are not optional and should be checked. If you think about it a statement 'add 3 from ss' is basically just wrong.

## Helpful Hints

These are the classes I used in my sample answer; there are many, many ways to do it so this is an FYI – not even a suggestion. Software Engineering students should be able to build different and probably better class arrangements.



In my solution I broke it into two tasks – syntax checking then running – this was a bad idea and cost me two extra hours of work, I recommend you do syntax check with execution of the code.

A statement in my solution was a tokenised form of the line :

e.g – the line ‘add’ 3 to jimmy’ would become the tokens

token-0 = add

token-1 = 3

token-2 = to

token-3 = jimmy

Note print statements have 2 tokens

If and while statements have 5 tokens

End, endif and endwhile have just 1 token

I had particular difficulty in parsing the == operator so if you are attempting task 4 and 5 then give thought to testing the two lines

‘If a==b then’

‘If a == b then’

### Getting Started

I am providing 3 files for you to get started-

Main

Menu (version 1 – Clean) – my code removed

Menu (version 2 – Dirty) – some of my code remaining – this is not a file you should use – it’s just to give you an idea of what I did for each option, it would be unwise for a student who is a weak programmer to copy paste this in and use it wholesale – tempting though that is. (Having said that you can use it – I don’t mind it’s not cheating IMHO).

## Menu Description:

Note only code the ones you need a minimum would be options, E,W,Q and S (though I recommend D to help with debugging)

Q	Exit/Quit	
W	Who Am I (Task1)	Who are you – student number and name
R	Read SAIL2023 File (not actually assessed and is optional)	Just reads and tokenises the file
E	Execute SAIL2023File (Task 1)	How to run
E2	Execute SAIL2023File (Task 2)	How to run task2 if different from task 1
S	Set File name	Should be obvious, enter a zero to input mandatory file names for testing
D	Execute With Debug (not assessed)	See my demonstration in lecture – 11/4/2023 to understand this option
T1	Debug and Testing (not assessed)	Testing code for the tokeniser
T2	Debug and Testing (not assessed)	Testing HashTable – repeated in lecture – 11/4/2023
T3	Debug and Testing (not assessed)	A simple set of tests for early interpreter development
T4	Debug and Testing (not assessed)	Just shows all variables