CM500288 Principles of Programming Coursework 1: SRPN

1 Introduction

This document provides the specification for the first Principles of Programming coursework.

You can use any environment for the development of your code, but we must be able to compile and run your code using Java 11 without requiring the installation of additional libraries, modules or other programs.

2 Learning Objectives

On completion of this unit, you will be able to:

1. Describe the design of a computer program separately from its implementation.
2. Explain the basic concepts of procedural and object orientated programming in the design and implementation of computer programs.
3. Explain debugging and testing methods and how they contribute to robust code
4. Design, construct and evaluate simple data structures and algorithms.
5. Plan, organise and implement program code to support reuse and maintainability of the software (project).

At the end of this coursework you will be able to design and write a medium-sized program using the appropriate procedural software techniques of data encapsulation and decomposition. You will also be able to understand, construct and use key data structures in a program.

3 Saturated Reverse Polish Notation (SRPN) Calculator

Whilst performing some maintenance on a legacy system you find that it makes use of a program called **SRPN**. **SRPN** is not documented and no one seems to know who wrote it, so your boss tells you to rewrite it in Java[1].

SRPN is a reverse polish notation calculator with the extra feature that all arithmetic is saturated i.e. when it reaches the maximum value that can be stored in a variable, it stays at the maximum rather than wrapping around.

Your task is to write a program, which matches the functionality of *SRPN* as closely as possible, flaws and all. Note that this includes *not* adding or enhancing existing features. We do not expect everyone to implement the calculator perfectly.

It is important that your solution works!

You have been provided with a repl.it link which allows you to run and interact with the SRPN program. You should start by typing in the tests on the following pages, observe the output and the implement your code to replicate this functionality.

We have also provided you with a second repl.it link with starting code. It contains two files (Main.java and SRPN.java). Please leave Main.java as it is. You should edit the SRPN.java file to write your solution.

[1] But we have only just started learning Java!? Please don't worry, this problem can be solved procedurally, the structure will be similar to what you've been learning over the last few weeks. We do not expect submissions to follow an Object Oriented (OO) Programming approach. We expect to see submissions consisting of Main.java and SRPN.java and we expect to see your code broken down into methods (functions) as this is good coding practice no matter the paradigm.

1. The program must be able to input at least two numbers and perform one operation correctly and output

```
Example Input:

10
2
+
=

Example Input:

11
3
-
=

Example Input:

9
4
*
=

Example Input:

11
3
/
=

Example Input:

11
3
%
=
```

2. The program must be able to handle multiple numbers and multiple operations

```
Example Input:

3
3
*
4
4
*
+
=

Example Input:

1234
2345
3456
d
+
d
+
d
=
```

3. The program must be able to correctly handle saturation

```
Example Input:

2147483647
1
+
=

Example Input:

-2147483647
1
-
=
20
-
=

Example Input:

100000
0
-
d
*
=
```

4. The program includes the less obvious features of SRPN. These include but are not limited to...

```
Example Input:

1
+

Example Input:

10
5
-5
+
/

Example Input:

11+1+1+d

Example Input:

# This is a comment #
1 2 + # And so is this #
d

Example Input:

3 3 ^ 3 ^ 3 ^=

Example Input:

r r r r r r r r r r r r r r r r r r r r r d r r r d
```

3.1 Investigating and getting started with SRPN

This is an investigative assignment. Once you have tried the inputs described above, you should spend some time trying other inputs into the calculator. The legacy SRPN calculator has a lot of functionality, some of it obscure. We are looking for you to recreate the same calculator, not to fix its flaws or to add new functionality.

It is important that your outputs are the same as the SRPN calculator. For example, SRPN's "Stack overflow." is not the same as "stack overflown" – note the spelling error, lack of capitalisation and missing full stop.

We will run the above tests, as well as others that are similar or more obscure. You are not expected to find and implement every feature. Start by implementing functionality to meet the tests under point 1. Then those under point 2, etc. Add bits of functionality at a time, continually testing that your solution works before adding more.

It is possible to spend many hours working on this assignment to attempt to replicate it perfectly – do not do this. You should manage your time to implement the core functionality (i.e. Tests similar to those in 1, 2 and 3) with some additional features if possible (i.e. tests similar to those in 4), then dedicate time to commenting and code quality.

4 Submission Instructions

Your source (`.java`) files must be submitted in a single zip, with all files in the root of the zip. You must not include subfolders within the zip file. There is no specific requirement for naming your zip file, simply name it something sensible.

Your .zip file must be submitted to the Engage assignment page by the submission deadline shown. Submissions received after this deadline will be capped at 40% if received within 5 working days. Any submissions received after 5 working days will marked at 0%. If you have a valid reason for an extension, you must submit an extension request by following the appropriate process outlined on this page `https://engage.bath.ac.uk/learn/mod/page/view.php?id=33003` – unit leaders cannot grant extensions.

You should leave yourself time to download your file from Engage, extract it, and check that you have attached the correct file, with the content that you want to be marked. **You** are responsible for checking that you are submitting the correct material to the correct assignment.

Please check Engage for the submission deadline.

5 Assessment

5.1 Conditions

The coursework will be conducted individually. Attention is drawn to the University rules on plagiarism. While software reuse (with correct referencing of the source) is permitted, we will only be able to assess your contribution.

5.2 Marking

Key considerations for marking the code will be: compiling, replicating original SRPN functionality. You stand to gain marks for well-structured and documented code. Marks are also awarded for well-named variables or methods and consistent formatting and indentation.

5.3 Marks Table

Below is a breakdown of marks, with descriptions on how each criteria is met.

Criteria	Max Score	Description
Running with expected input	max 60	Code replicates the functionality of the SRPN program to read input and produce expected outputs. This includes handling incorrect user inputs.
Code quality, design, formatting and commenting	max 40	Code makes proper use of coding techniques, such as data encapsulation and generalisation, use of functions and return statements, and robustness. Appropriate variable and function names. Code is well-structured and indented. Appropriate comments are used to document functions.

In cases where it is not clear from the assignment how it should be marked, you may be called on to explain or demonstrate your program. Such cases include but are not limited to suspected plagiarism.