Principles of Programming
Coursework Part 1: Java SRPN

1 Introduction

The coursework of this unit consists of three parts: the lab sheets and two larger Java courseworks. This document provides the specification for the first larger coursework: **SRPN**.

You can use any Integrated Development Environment (IDE) for the development of your scripts, but we must be able to compile your code on linux.bath.ac.uk without requiring the installation of libraries, modules or other programs.

Questions regarding the coursework can always be posted on the Moodle Forum and/or the mailing list.

2 Learning Objectives

At the end of this part of the coursework you will be able to design and write medium-sized program using the appropriate procedural software techniques of data encapsulation and decomposition.

3 SPRN

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

Your task is to write a program, which matches the functionality of **SRPN** as closely as possible. Note that this means **not** adding to or enhancing existing features. The output of your program should be identical to SRPN when given the same inputs. **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.

The **SRPN** program can be downloaded from the unit's Moodle pages. Furthermore, your have been provided with a marking script which gives you the opportunity to test your code and get an indication of the marks you will received. Our marking script will be the same but will use different test-cases. The marking script zip file contains

1. **srpn.o** - the legacy program you are replicating. To run from the command line on linux.bath.ac.uk, use ./srpn.o
2. **mark-srpn** - a script which compares the output from srpn.o to your implementation based on the inputs defined in the files found in the folders t-XXX. Note: the main method for your code should be in the file SRPN.java in the folder SRPN. To run this script from the command line on linux.bath.ac.uk, navigate to the srpn folder and type ./mark-srpn
3. **t-XXX** - each folder contains example input and is used by the marking script to test your code. Note: our marking script will be the same but will use different test-cases.
4. **SRPN** - folder where your SRPN.java class with the main method should be placed. Note: you can use as many other classes as you want to define and should all the placed in this folder.
5. **SRPN/SPRN.java** - java class with the main method, code needed to read input from the command line and the processCommand method. To start this coursework you might start by modifying this method.

The tutors in the lab can also help you make sure that your program is recognised by our marking script.

Your program will be tested on the following inputs and others that are similar.

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

```
Input:

10
2
+
=
```

```
Input:

11
3
-
=
```

```
Input:

9
4
*
=
```

```
Input:

11
3
/
=
```

```
Input:

11
3
%
=
```

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

```
Input:

3
3
*
4
4
*
+
=
```

```
Input:

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

3. The program must be able to correctly handle saturation

```
Input:

2147483647
1
+
=
```

```
Input:

-2147483647
1
-
=
20
-
=
```

```
Input:

100000
0
-
d
*
=
```

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

```
Input:

1
+

Input:

10
5
-5
+
/

Input:

11+1+1+d

Input:

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

Input:

3 3 ^ 3 ^ 3 ^=

Input:

r r 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
```

4 Assessment

4.1 Conditions

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

4.2 Marking

Key issues for marking the code will be: compiling, running with expected input, robustness (handling incorrect user input), module design, proper use of data encapsulation and decomposition, and the algorithms being used. You stand to lose marks for repeated or badly structured or documented code. Marks will also be deducted for ill-named variables or functions or for inconsistent indentation.

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.

5 Submission Instructions

The deadline for this coursework is **Friday 23rd November at 5pm**. You should upload your solution before the deadline via Moodle.

Your program solution should be a zip file that can be extracted to a directory with the name SRPN-yourusername. The directory should contain the files necessary to compile and run your code SRPN.java with any additional class files needed to allow your program to compile and run. Before you upload your solution, make sure that the zip file contains all necessary files and creates the correct directory. Please download the file from Moodle and double check that you have attached the right 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.

6 Feedback

Individual detailed feedback will be provided via Moodle within three weeks of the submission deadline. Clarification of the feedback can be obtained from the tutors during the labs.