Computer/Engineering Programming 1 – Practical Class 3

Variables, Assignment, Expressions and I/O

Aims and Objectives

This laboratory has been designed to help you to

- make use of variables to store, retrieve and update values,
- construct arithmetic expressions which make use of variables, literal values and operators such as integer "division" (/) and "remainder" (%),
- to create and use an input stream to read lines of text from the keyboard,
- convert a string of digits to an `int` value.

Getting Started

By default Java and *jGrasp* expects the name of the class containing the main method to be the same as the name of the file containing the class. To avoid problems it is advisable to adhere to this convention. For example,

- Class name: `MyClass`
- Source file name: `MyClass.java`
- Class file (containing bytecode generated by the compiler): `MyClass.class`

On with the prac. . .

1. Create a directory called prac2 (in your cp1 directory) and make it your current working directory.
2. Copy all the Java files contained in the Week 2 area on FLO to your current working directory (prac2).

During the next stage you will need to create a file to put your Java source code in. You can either create the file from scratch or use a copy of an existing file and modify it. If you use the latter approach ensure the file name and class names match (as described above). Refer back to practical 1 for information on using *jGrasp* and compiling and running Java applications.

Task 1: Using Variables

Create a Nassi-Shneidermann diagram (using Structorizer), which solves the following problem:

1. Declare an `int` variable called `total` with an initial value of 0
2. Declare an `int` variable called `val1` which has no initial value
3. Declare an `int` variable called `val2` which has no initial value
4. Assign 26 to the variable `val1`
5. Assign a value 1 bigger than the value of val1 to the variable `val2` (you should not use the literal value 27).
6. Assign the sum of `val1` and `val2` to the variable `total`

7. Print out the values of `val1`, `val2` and `total` so that each is separated by a comma

Once you have completed the diagram and are satisfied that it does solve the problem, write a Java application which translates the solution into code.

Once you have successfully compiled your program, on running it your output should be similar to the following:

```
26, 27, 53
```

__________________ **Checkpoint 6** __________________
Show the program source code and the output to a demonstrator

Task 2: Reading, Storing and Outputting Strings

Open the file `Echo.java` (it will have been copied earlier and so should be in your current working directory) from *jGrasp* and then compile and run it. It reads a line of input and echoes it back to the screen.

Modify the program so that it prompts for, and reads, two separate lines of input and outputs the contents of the lines in reverse order, on the same line, separated by a comma. Here is how your modified program should behave (user input is in **bold**):

```
  Enter a line:
  first
  Enter another line:
  second
  second, first
```

__________________ **Checkpoint 7** __________________
Show the program source code and the output to a demonstrator

Task 3: String to Integer Conversion

Make a copy of the file from the previous checkpoint and use this as your starting point and rename it: `Number.java`

Return to the *jGrasp* window and open the file. Change the name of the class to `Number`. Modify the program so that rather than printing out the two strings which have been read in, it treats them as being two halves of a single integer. Your program should print out the value of that integer plus 1. Here is how your modified program should behave (user input is in **bold**):

```
  Enter a line:
  23
  Enter another line:
  99
  Number = 2400
```

The complete number is `2399` and adding 1 gives `2400`.

Hint: make use of the `Integer.parseInt()` method (see page 90 of the textbook).

__________________ **Checkpoint 8** __________________
Show the program source code and the output to a demonstrator

Task 4: Integer and Floating Point Division

Create a Nassi-Shneidermann diagram (using Structorizer), which solves the following problem:

1. prompts for and reads in an integer (on the same line)
2. Outputs:
   i) the value of the number divided by 10 as a floating point value
   ii) just the fraction part of the above number
   iii) the remainder when the number is divided by 10
   iv) the number of times 10 divides the integer

Once you have completed the diagram and are satisfied that it does solve the problem, write a program which translates the solution into code.

Here is how your program should behave (user input is in **bold**):

```
Enter an integer: 634
63.4
0.4
4
63
```

__________________ **Checkpoint 9** __________________
Show the diagram, program source code and the output to a demonstrator

Task 5: Variables and Division Expressions

Your task is to develop software for a machine which dispenses change. It accepts an amount in cents and outputs the ways that amount can be made up from 1, 5 and 20 cent coins (ignoring the fact that we no longer have 1 cent coins) using as many of the larger valued coins as possible.

Here is how your program should behave (user input is in **bold**):

```
Enter an amount of cents in the range 0 to 100: 77
Dispensing...
  3 20c coin(s)
  3 5c coin(s)
  2 1c coin(s)
```

Hint: 77 % 20 is 17 and 77 / 20 is 3.

Before writing any code ensure that you have created a design (Nassi-Shneidermann diagram) for the solution.

__________________ **Checkpoint 10** __________________
Show the diagram, program source code and the output to a demonstrator