The University of Melbourne
School of Computing and Information Systems

COMP90041

Programming and Software Development

Second Semester, 2018

Lab 3 — Programming Practice (Week 4)

Introduction to Java programming

This week's exercises practice working with command line arguments, using constants from the library, and writing loops.

Workshop Exercises

These are just for practice, and will not be assessed.

1. Write a program that takes one command line argument, which should be the abbreviation of a compass direction, that is, one of `N`, `S`, `E`, or `W`, The program should print out number of degrees east of north that are specified by that heading. So for example, if the input is `N`, it should print out 0, and if it is `W`, it should print out 270. If the input is not a valid compass direction abbreviation, print out a suitable error message and exit the program.

   Hint: If you are using Java 7 or later, this is easier, as Java 7 introduced switching on strings. Your IDE may allow you to specify which Java version it supports, or you can execute `java -version` on the command line to check the version. If it says version 1.7 or more, you can switch on strings.

   Hint: If `s` is a string, `s.charAt(0)` will return the first character of the string. Every Java version allows you to switch on characters.

   Hint: You can exit any program immediately in an error state with the statement

   ```
   System.exit(1);
   ```

   If you replace the 1 with a 0, the statement is appropriate for exiting immediately in a success state, as would happen if you reached the end of the `main` method normally.

Homework

This will also not be assessed.

2. Write a program to print a 10×10 multiplication table. Print the numbers from 1 to 10 across the top and down the left side, and show the products in the middle. Make sure your columns line up neatly. Print a heading row at the top showing the numbers from 1 to 10, and a leftmost column also showing the numbers from 1 to 10. To make it look nicer, print a `|` character between the columns of each row, and a solid row of `-` character between the rows. To make it look even neater, print a `+` character wherever the vertical and horizontal lines meet.

   The output should look like this:

```
  *  |   1 |   2 |   3 |   4 |   5 |   6 |   7 |   8 |   9 |  10
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
   1 |   1 |   2 |   3 |   4 |   5 |   6 |   7 |   8 |   9 |  10
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
   2 |   2 |   4 |   6 |   8 |  10 |  12 |  14 |  16 |  18 |  20
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
   3 |   3 |   6 |   9 |  12 |  15 |  18 |  21 |  24 |  27 |  30
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
   4 |   4 |   8 |  12 |  16 |  20 |  24 |  28 |  32 |  36 |  40
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
```

```
   5 |   5 |  10 |  15 |  20 |  25 |  30 |  35 |  40 |  45 |  50
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
   6 |   6 |  12 |  18 |  24 |  30 |  36 |  42 |  48 |  54 |  60
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
   7 |   7 |  14 |  21 |  28 |  35 |  42 |  49 |  56 |  63 |  70
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
   8 |   8 |  16 |  24 |  32 |  40 |  48 |  56 |  64 |  72 |  80
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
   9 |   9 |  18 |  27 |  36 |  45 |  54 |  63 |  72 |  81 |  90
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
  10 |  10 |  20 |  30 |  40 |  50 |  60 |  70 |  80 |  90 | 100
```

Hint: You will need to use at least two loops to display the body of the table, one inside the other. The outer loop will iterate over the rows of the table, and the inner loop will iterate over the columns for the current row.

Hint: Start out just printing the body of the table. When you get that looking right, add the | column separators. Then add in the column and row headings, and finally add the horizontal lines separating the rows.

This is often the best way to develop code: implement a little bit at a time, test and debug it, then move onto the next little bit, until the entire thing works as intended.