Worksheet 2: Introduction to Programming

Updated: 28th February, 2020

The objectives of this practical are:

- Learn how to write pseudo code
- Learn how to write Java code
- Testing your programs
- Evaluate java expressions

Note: You must submit this practical worksheet **by the start of your next registered practical**, and have it signed off during that session.

Submissions must be done **electronically** through Blackboard. You are only required to submit once for this practical. You must submit everything you have completed in this practical, both in class and at home.

You must submit this Worksheet as a whole, so your weeks work **and** the Assignment Task, additionally if you wish you may include the optional task at the end.

You must also not modify the files in `PDI\P02` until after your submission has been marked. During your practical your tutor will check the modified dates of the files using `ls -l` (el) before marking your work.

To create a gzipped tarball use the following command from your `PDI` folder:

```
[user@pc]$ tar -cvzf <studentID>_P02.tar.gz P02
```

For more information on what each argument of the above command does use:

```
[user@pc]$ man tar
```

To get your **`.bash_history`** into a submittable format, first close all terminals down, using <ctrl>-d. Then open a new terminal, and type this command from anywhere:

```
[user@pc]$ history >~/Documents/PDI/P02/BashHistoryP02.txt
```

Your submission will be marked during your practical session via an interview from your tutor comprised of a few questions.

Note that the questions asked in the interview may cover the entirety of the worksheet, not just the material you submit. Your submitted work will be assessed and marks will be allocated accordingly as per the assignment question.

1. Revision: Pseudo Code

Last week you created a file in your `P01` directory called `Calculator.txt`. This file is a good example of the pseudo code we expect to see in this unit. Feel free to use this file as a reference when writing your own pseudo code.

Can you remember the commands to navigate to that directory and open it in the terminal text editor?

Your tutor will now go through the pseudo code from last week to explain the different parts and what they mean, just to reinforce the concepts.

Its okay if you don't remember fully, now is your chance to ask as many questions as possible before you write your own!

2. Write your own

Now that you have an example of pseudo code, you can now write your own!

Your task is to write a simple program, in pseudo code, that converts a number that a user inputs, from Celsius to Fahrenheit.

Your program should follow these steps:

(a) Ask the user to input the temperature in Celsius (**c**)

(b) Convert it to Fahrenheit (**f**)

(c) Output the result to the user

To assist you in making this program, there is a document on blackboard called **Pseudo Code Style Guide**. This document should be your go to for all things pseudo related. It also shows you the java equivalent for your pseudo code.

If you're having trouble writing this or don't know where to start, ask your tutor for help and they will happily guide you in the right direction.

Note: For your reference the conversion from Celsius to Farenheit is:

$$\frac{\mathbf{c} * 9}{5} + 32 = \mathbf{f}$$

Lets open a file called **`CelsToFaren.txt`**

Design your algorithm within this file, in pseudo code.

Once you have complete your pseudo code, **please get your tutor to have a look at it first before moving onto the next step**.

Note: Fixing problems early can save you a whole lot of time later down the track

3. Java Conversion

Last week you created a file in your `P01` directory called `Calculator.java`. This file is a perfect example of Java code we expect to see in this unit. Feel free to use this file as a reference when writing your own Java code.

Now that you've written the pseudo code for your temperature conversion, you can now convert it to Java.

You are required to comment your code from now on. Here is an example:

```
// This is an In-line comment describing something

/* This is a Block comment that will continue to
   comment until it receives the corresponding */
```

Once again, to assist you with this, there is is a document on blackboard called **`Java Coding Standard`**. This shows you the expected formatting and syntax required for this unit.

Once you have completed this, compile your code and fix any error messages that may appear. If you don't know what they mean, try to look at the numbers and the error text.

For example: Can you tell what is going wrong with this compiler error?

```
MyComplexProgram:171: error: cannot find symbol
        someValue = sc.nextInt();
        ^
    symbol:   variable someValue
    location: class MyComplexProgram
```

If you're still unsure, ask your tutor to walk you through what they are and how to fix them.

> **Note:** A helpful hint, keep a journal of different error messages you see and how you fixed them. This will save you a lot of time in the future when writing other programs.

Feel free to refer back to your first practical if you have forgotten how to compile and run your program.

Once you have fixed your errors, you can now run your program.

> **Warning:** Just because your program compiles, that does not mean it works! This is why we must test it, every time.

If your program works as intended, **please get your tutor to have a look at your java code before moving onto the next step**.

4. Testing your program

For every piece of java code you write, you must also test that it works as intended. There is no such thing as a 'bug' in programming, only mistakes (or features, depending on who you ask).

There is a document on blackboard that shows you the template for testing your programs. Download this document and move it from ~/Downloads/to your ~Documents/PDI/ folder.

Make a copy of this template in your P02 directory. Please use this for every program you write.

Once you have downloaded this document, run your program using the first two example test cases to see if it works. If those tests are positive, fill in the last 2 blank spots with your own test data and check the results of your program with your expected results.

Note: You **must** use the command line and vim to edit this, you may only use a mouse to download the file from Blackboard.

5. Evaluating Java Expressions

Given the following variable declarations and initialisations as below, evaluate the Java expressions. You must show all of your working out. Make sure you keep in mind the order of operations (BIMDAS/PEDMAS).

Store your answers in a file called **JavaExpressions.txt**

```
String marvel = "Avengers";
double x = 314.219, y = 240.213;
int mcu = 1999, prime = 614, cap = 2;
```

(a) `(int)x - mcu / 100`

(b) `marvel + prime`

(c) `(double)((prime - (int) y)/2)`

(d) `(mcu / 100 * cap ) * cap - prime / 10`

(e) `( mcu % 10 != 0 && ( cap > 10 || y > 100 ))`

6. Assignment Task

Your last task for this practical is to complete the following task **in your own time**.

> **Warning:** This question goes towards your portfolio/assignment mark and thus any collusion will be dealt with as per university policy.

Create a Sub Directory in your `P02` directory called **`AssignmentTask`** and move into it.

Your task is to create an `ASCII` converter, you will need to gather an uppercase character input from the user, and convert it to the `ASCII` value it represents. Then you will need to display that back to the user, as well as the lowercase equivalent.

You must design this first in pseudocode, then convert it to Java and then test it with another copy of the test file.

Name your files **`CharConverter.txt`**, **`CharConverter.java`** and **`CharConverter_Test.txt`** accordingly.

Open a file called **`WhyIsThisImportant.txt`** and answer the question: Why would we need to use `ASCII` values? Think about the difference between `Character`'s and `Integer`'s.

Hint: You may assume that all inputs are valid uppercase characters.

For example:

```
Please enter an uppercase character: A
The ASCII value of 'A' is: 65
The lowercase value of 'A' is: 'a'
```

> **Note:** `ASCII` is a code for representing English characters as numbers, each letter of english alphabets is assigned a number ranging from 0 to 127. For example, the `ASCII` code for uppercase P is 80.
>
> You may need to have a look at the difference between uppercase and lowercase characters in order to do the conversion.
>
> An `ASCII` table has been included for you

You will need to read the lecture notes and attend this weeks lecture in order to work out how this can be achieved.

If you are having trouble feel free to attend the Senior Tutor, they will be more than willing to guide you in the right direction

ASCII Table

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
48	30	110000	60	0	96	60	1100000	140	`
49	31	110001	61	1	97	61	1100001	141	a
50	32	110010	62	2	98	62	1100010	142	b
51	33	110011	63	3	99	63	1100011	143	c
52	34	110100	64	4	100	64	1100100	144	d
53	35	110101	65	5	101	65	1100101	145	e
54	36	110110	66	6	102	66	1100110	146	f
55	37	110111	67	7	103	67	1100111	147	g
56	38	111000	70	8	104	68	1101000	150	h
57	39	111001	71	9	105	69	1101001	151	i
58	3A	111010	72	:	106	6A	1101010	152	j
59	3B	111011	73	;	107	6B	1101011	153	k
60	3C	111100	74	<	108	6C	1101100	154	l
61	3D	111101	75	=	109	6D	1101101	155	m
62	3E	111110	76	>	110	6E	1101110	156	n
63	3F	111111	77	?	111	6F	1101111	157	o
64	40	1000000	100	@	112	70	1110000	160	p
65	41	1000001	101	A	113	71	1110001	161	q
66	42	1000010	102	B	114	72	1110010	162	r
67	43	1000011	103	C	115	73	1110011	163	s
68	44	1000100	104	D	116	74	1110100	164	t
69	45	1000101	105	E	117	75	1110101	165	u
70	46	1000110	106	F	118	76	1110110	166	v
71	47	1000111	107	G	119	77	1110111	167	w
72	48	1001000	110	H	120	78	1111000	170	x
73	49	1001001	111	I	121	79	1111001	171	y
74	4A	1001010	112	J	122	7A	1111010	172	z
75	4B	1001011	113	K	123	7B	1111011	173	{
76	4C	1001100	114	L	124	7C	1111100	174	|
77	4D	1001101	115	M	125	7D	1111101	175	}
78	4E	1001110	116	N	126	7E	1111110	176	~
79	4F	1001111	117	O	127	7F	1111111	177	*[DEL]*
80	50	1010000	120	P					
81	51	1010001	121	Q					
82	52	1010010	122	R					
83	53	1010011	123	S					
84	54	1010100	124	T					
85	55	1010101	125	U					
86	56	1010110	126	V					
87	57	1010111	127	W					
88	58	1011000	130	X					
89	59	1011001	131	Y					
90	5A	1011010	132	Z					
91	5B	1011011	133	[					
92	5C	1011100	134	\					
93	5D	1011101	135	]					
94	5E	1011110	136	^					
95	5F	1011111	137	_					

7. (Optional) Date Divider

Note: This question is optional, but very good practice for you to reinforce the concepts up to now.

Your task is to create a program called **SplitMyDate** in pseudo code to convert an input from the user (**DDMMYYYY**) to their respective **day**, **month** and **year** variables. With the output formatted as below.

INPUT: 20022020

```
It is day 20 of month 2, in the year 2020.
9 years ago was day 20 of month 2, in the year 2011.
```

Hint: Think of how you may be able to use **MOD** to achieve this.

You may assume that the user will only ever enter it in the correct format. e.g., 8 characters and each value is within its bounds.

Once you have written it in pseudocode, convert it to Java and test it **fully**.

The output for your Java program would look something like this:

```
[user@pc]$ java SplitMyDate
Please enter the date (DDMMYYYY): 20022020
It is day 20 of month 2, in the year 2020.
9 years previous was day 20 of month 2, in the year 2011.
```

End of Worksheet