School of Computer Science, The University of Nottingham

G51PGP Programming Paradigms, Spring 2017

Jason Atkin

Object Oriented Coursework 1, Part 1/3

Release date: Monday 13th February 2017

Deadline: Tuesday 21st February 2017, 4pm

This exercise sheet is worth 20% of the first Object Oriented coursework (parts 2 and 3 are worth 40% of that mark each).

This is assessed during the OO parts of the G51PGP labs on Tuesdays. If you do not finish within the first lab after it is released please attempt to complete the exercises in your own time or in the next lab time to get it marked before the end of the lab on the 21st.

Once you have had it marked, please submit a copy of the coursework files to moodle for a permanent record. Your mark will be recorded against the coursework in moodle at some point soon after the lab when all marks are uploaded.

If you get stuck you can ask for help from the tutors during the weekly lab sessions. Note that tutors will only be available to answer questions and assess your solution during the official lab sessions (Tuesdays, 2pm to 4pm, A32) and you should ask OO/Java questions during the OO hour of the labs.

You **may** complete this coursework in pairs if you wish, working together, however you should ensure that you both fully understand what you have done and ensure that you both upload the files to moodle.

You may use any IDE you wish to use for these courseworks. I recommend Eclipse and have tested my own solutions using that. If you prefer another IDE (e.g. Netbeans or IntelliJ) then that is fine. I will use Eclipse in the lectures, it is installed on the lab PCs and virtual desktop, and you can easily download your own copy, so it is easy for you to get access to. It often doesn't even need to be installed, just unzip the download and run it. Eclipse will run on many different operating systems, including Mac, Linux and Windows, see: https://eclipse.org/downloads/. If you are stuck on using Eclipse, please read the 'getting started with Eclipse' document for the previous lab.

Important: Whatever IDE you use, ensure that your new projects will be stored in a safe place which will be backed up, or back up your work regularly. I suggest your home directory.

To complete this coursework, complete the following steps, then ask one of the lab tutors to mark it (see the marking scheme below).

1. Create a new project for this coursework in your chosen IDE. In Eclipse you would just choose to create a new "Java Project". Give the project a meaningful name, e.g. G51PGPOOCW1.
2. Add a new Java class to your project. Create a main() function in it. (Hint: in Eclipse, just tick the checkbox which says "`public static void main( String[] args)`" under the heading "Which method stubs would you like to create?" when you create the class.) Give your class a meaningful name. Remember to make the first letter of a class name a capital.
3. Create static methods for a Java version of the C code sample which is on the following page.
4. Run your program. If it works correctly, when it decodes the string you will end up with the string you started with.
5. Get the result marked by the lab helpers.
6. Upload the Java file(s) for the marked program to moodle.

Hints:

- Look at what we did in the lectures to convert code from C to Java.
- Look up charAt() and setCharAt() for String and StringBuffer
- If you need to change the characters in a String, make it a StringBuffer instead of a String – Strings are immutable (unchangeable) objects
- You will need to cast the result of the calculate back into a character for setAt(). See the example and comments in Lecture OO-3.
- In C, the end of the string is detected with a character 0. How do you detect whether m or p reaches the end (i.e. length) of the String in Java?
- I copy-pasted the C code into the Java file and then just changed each thing in turn until it compiled. This avoids typing it all in.

Marking Scheme: (total 4 marks)

1 mark for each for the following:

- Java program with static methods which works. (1 mark)
- Uses String and/or StringBuffer appropriately. (1 marks)
- Messages are encrypted/decrypted correctly. (1 mark)
- Structure of program matches the structure of C program. (1 mark)

Note: you should all be able to get full marks on this coursework although some of you will find it a lot easier than others. The aim of this coursework is to encourage you to try some Java and to see the similarities with what you already know from C, as well as to learn a bit about Strings and StringBuffers. If you don't finish in the first hour (in the first lab) you can get it marked in the next lab (21st February) so don't worry.

C program to convert to Java static methods:

```
#include <stdio.h>
#include <string.h>

void encrypt(char* message, char* password)
{
	int m = 0, p = 0;
	while (message[m])
	{
		message[m] = 32 + (126 + message[m] + password[p] - (m%95)) % 95;
		m++;
		p++;
		if (password[p] == 0)
			p = 0;
	}
}

void decrypt(char* message, char* password)
{
	int m = 0, p = 0;
	while (message[m])
	{
		message[m] = 32 + (95+message[m] - password[p] + m) % 95;
		m++;
		p++;
		if (password[p] == 0)
			p = 0;
	}
}

void encryptDecrypt(char* word, char* password)
{
	printf("Encrypt word :    '%s' using password '%s'\n", word, password);
	char copyOfWord[128];
	strcpy(copyOfWord, word);
	encrypt(copyOfWord, password);
	printf("Encrypted word is '%s'\n", copyOfWord);
	decrypt(copyOfWord, password);
	printf("Decrypted word is '%s'\n\n", copyOfWord);
}

void main()
{
	encryptDecrypt("Hello world", "password!");
	encryptDecrypt("Hello world", "pass");
	encryptDecrypt("Hello world", "word");
	encryptDecrypt("This is a longer message", "password!");
	encryptDecrypt("This is a longer message", "pass");
	encryptDecrypt("This is a longer message", "word");
	encryptDecrypt("aaaaaaaaaaaaaaaaaaaa", "password!");
	encryptDecrypt("abcdefghijklmnopqrstuvwxyz", "password!");
	encryptDecrypt("AbCdEfGhIjKlMnOpQrStUvWxYz", "password!");
	getchar();
}
```