UNIVERSITY OF PORTSMOUTH

OS Week 1 - Lab

Lab Books and Java Reprise

Introduction

Welcome to Operating Systems and Internetworking, your second-year "architectures" module. This half of the module will be preoccupied with the underlying concepts of operating systems and the computers they run on.

Part of the assessment of the module is based on a lab book kept in practical sessions, recording experiments you do and answering various questions and exercises. This will help you to pass a set of requested quizzes. You will be asked to maintain this lab book in the form of a Moodle Wiki. In this first lab session we just want you to get experience using this format, and recap on previous experience you may have running Java in NetBeans.

This is not a programming module, but in the Operating Systems labs, we will often use programs written in Java to illustrate concepts introduced in the lectures. The few parts of Java that are essential to these labs will be reviewed in these first two weeks of labs. Moreover, all essential experiments in the weekly lab scripts can be completed without actually *writing* Java programs. Java will be used especially to illustrate ideas about *concurrency*.

You will however find there are exercises accompanying the labs, and in these, we *do* typically ask you to write simple programs. If you attempt these and record the results in your lab book, discussed below, it should help your understanding. The final exam does *not* require specific Java programming skills.

For the experiments in this first lab - apart from reviewing the Wiki editor used for writing lab books - you only have to run a "Hello world" program using NetBeans, the preferred development environment for this teaching block.

The Lab Book

30% of the credit for this module comes from its lab books (or logbook). In your lab book, you will keep a record of your experiences in these labs. The lab book will be assessed by the staff team as we go – This will help you with the quizzes and may serve as one of your revision aids when you come to sit the final exam (which accounts for 70% of your grade).

You should plan to complete your lab book entry for any given week before the

following week's timetabled practical session. Expect this to require some work outside the lab sessions themselves. Each lab has the following general achievements:

Keeping a systematic record of your activities and understanding from the lab sessions - in particular in relation to singled out *experiments*. Answering certain *reflective questions* that will be highlighted in the lab scripts.
Attempting *exercises* set at the end of the lab scripts, and recording your results

So you will encounter examples of "experiments", "reflective questions" and "exercises" in this simple first lab script below.

Under the Operating Systems theme, you will have 3 quizzes. We will consider the best 2. Most of the quizzes' questions will be similar to your lab books' questions.

We ask you all to maintain your lab books in the individual Wikis you will find on the Moodle page for the module. Begin your Wiki for the Operating Systems theme now.

The Week 1 Lab Script

Click on the activity "**Operating Systems Lab Book Wiki**" near the top of the Moodle site for the module. This is your portal to store all your lab book files.

In this first week make sure you experiment with at least the following:

Experiment 1

From the Lab Book Wiki menu choose "Files". Click on "Edit wiki files" and upload at least one file of some external type, e.g. Word, PDF, PowerPoint, etc.

Tips for editing your lab book Wiki:

- Use Chrome or Firefox rather than Microsoft browsers.
- Always click the "Save" button at the bottom of the editing page before leaving the edit page! (Save regularly!)
- Complete your weekly lab book using Microsoft Word and make sure it is organized then upload it to your Wiki portal.

Java "Hello World" using NetBeans

We need to make sure that you can run a Java program, preferably in Netbeans, on whatever platform you find yourself working on.

Experiment 2

If you are using a university computer, or otherwise have access to AppsAnywhere, start it, log in if necessary, and search for and launch "NetBeans".

Otherwise, make sure you have a current version of NetBeans installed from: https://netbeans.org/, and that you can run the main IDE.

Experiment 3

Now follow the instructions at http://netbeans.org/kb/docs/java/quickstart.html. In "Setting Up the Project", you can leave the project location as `N:\` if you are on a lab computer or choose your own preferred location for NetBeans projects if you are not. Either way, NetBeans should create a project folder called "HelloWorldApp" in the destination folder.

Every Java program consists of one or more modules called *classes*. In this case, we have a single class called

`HelloWorldApp`.

Classes contain executable instructions inside named sections called *methods*. In our case we have a single method called `main` - it is the "main method" of our program (and `HelloWorldApp` is the "main class").

In our case, there is a single statement in the method that prints out the string "Hello World" using the method `println`. The prefix `System.out` identifies a particular "output stream", which in our case just refers to the NetBeans console output.

The class `HelloWorldApp` is created in a package called `helloworldapp` (all lower case), so there is a `package` declaration near the top of the program.

In Java, a *package* is a group of related classes that have a common prefix added to their name. Source files (and class files) in a package are usually stored together in a folder named after the package. Using packages is considered good practice, but since our application only has a single class it is slightly redundant.

You should compile and run the program now.

Reflective Question

What in your opinion are the relative merits or demerits of Java as compared to other computer languages you have used? What about the Netbeans development environment - how does it compare with other environments you may have encountered? This week you need only answer briefly.

Exercises

1. Write a Java program that chooses a random integer between 0 and five thousand and prints it out (here is one way to do this - not necessarily the simplest).

2. Extend that to a main program that chooses a random number, *n*, between 0 and five thousand, prints the value of *n* in a message, then runs `Thread.sleep(n)` before printing some other message. You will need to change the header of the `main()` method to:

```
public static void main(String [] args) throws Exception
```

Run the program a few times - can you see the variation in behaviour?

In later labs, we will use approaches like this to simulate non-deterministic delays imposed by schedulers.

3. The project we created in Experiment 2 was a *Java application*, meaning it starts with a single static `main()` method. Java applications can also be run as standalone programs - for example from the Windows Command Prompt. Let's experiment with that.

Start a Command Prompt and, if you are working on a lab computer, go to the N-drive by issuing the command:

```
N:
```

if necessary. Otherwise just use the `cd` command to go to your top level Netbeans projects folder, if it is not your home directory.

Change to the `classes` subfolder of your NetBeans project, which, if you followed the defaults, could be done by issuing the command:

```
cd HelloWorldApp\target\classes
```


Now you should be able to run the `HelloWorldApp` program in the Command Prompt by issuing this command:

```
java helloworldapp.HelloWorldApp
```

(java [Pakagename].class name)

If successful it will respond with a Hello World message.

If the java command isn't recognized, try first issuing a command like this:

```
set PATH=%PATH%;"C:\Program Files (x86)\Java\jre1.8.0_211\bin\"
```

You should be able to just copy/paste commands like this into the Command Prompt, but you will have to look in your "Program Files", folder and edit in the exact version of Java.

The java command above is actually running a Java class file in a subdirectory, `helloworldapp\HelloWorldApp.class`. If the command still fails, check this file exists.

In a Java application *command line arguments* are accessible through the `args` parameter of `main()` method. For example if the command was:

```
java classname arg-0 arg-1 ...
```

the argument string *arg-0* would be accessible in the program as `args [0]`, the string *arg-1* would be accessible as `args [1]`, and so on.

By using two elements of the `args` array, write a new application project called just `Hello` such that the command:

```
java hello.Hello Tamer Elboghdadly
```

prints out:

```
Hello Tamer Elboghdadly!
```

Before running Hello from the command prompt you will have to click the "build" icon in NetBeans - it looks like a hammer - to create the Hello class file, which this time will probably be under N:\Hello\build\classes. (If you wanted to run programs using `args` from *within* Netbeans, you would have to find the project's Run properties and set the "command line" arguments there.)

SPECIAL TOPIC: Hello World in C using Code::Blocks

This is optional, but in operating systems, it is useful to have some appreciation of the C language!

If you are working on a home computer, you may have to change "N:\" in these instructions to some other folder name. You may also have to manually install Code::Blocks and GCC.

1. Go to AppsAnywhere and search for the Code::Blocks IDE. Start it running.
2. Hopefully the GCC compiler will be detected - click through the screen concerning compiler detection if it appears.
3. Click on "File" -> "New" -> "Project", and select "Console Application".
4. Select the language as "C".
5. Choose a Project title (I used "wk1egs"). You can set "Folder to create the project in" as "N:\".
6. Accept defaults on the next screen, and click "Finish".
7. On the file chooser at the left of the screen, go to "Sources" and double click on "main.c".
8. You should find a ready-made "Hello World" program, which you can edit if you wish to.
9. When you are ready, click the run button (green triangle) to optionally build and run the project.

If you want to run your program directly from a Command Prompt:

1. Start a Command Prompt running and navigate to the folder "N:\ <project-name>" using commands similar to the ones described in exercise 3 above.
2. Change folder to "bin\Debug".
3. Run "<project-name>.exe" as a command (it should be sufficient just to type the project name at the Command Prompt, as long as it is a single "word").

Can you adapt your program to take your name as an argument to the Hello World program and print it, as we did for Java in exercise 3? There are some relevant examples here.

