CS5001 Object-Oriented Modelling, Design and Programming
Practical 0 – Hello World

School of Computer Science
University of St Andrews

Due Friday week 1, weighting 0%
MMS is the definitive source for deadlines and weightings.

This exercise permits you to practice using the School servers and our Module Management System (MMS) for submitting coursework and will introduce you to programming in Java. It will also give you the opportunity to use and familiarise yourself with our automated checking tools (which will be used for many assignments in future). Although this exercise involves submitting an archive file containing your work to MMS, it is not formally assessed and merely provides you with a chance to practice procedures that you will need from now on.

You can discuss this practical, as well as watching a video of me doing the whole thing, in the following thread: https://discourse.cs.st-andrews.ac.uk/t/practical-0-hello-world/901/. This is a demo on Windows, but I'm willing to do a Linux one as well if there's demand.

Task 1: Basic Requirement

For this task you will be using a text editor to write a short program and the Java compiler and Java runtime from a *Terminal Window*, to compile and run your program. You might want to refer to the first exercise sheet here for some of the details. Try to complete the following steps.

- Create a new folder with the name `p0-hello`.
- Open the `p0-hello` folder and similarly to above, create a new folder with the name `src`.
- Open a terminal/command prompt window and navigate to the `src` directory using the `cd` command.

- Create a new file in your **src** directory, called `HelloWorld.java`, and open it with a text editor.
- Type in the `HelloWorld` program from the lectures. You can find the slides here.
- Once you have finished writing the program, save it.
- In the Terminal window, you can compile your program now by typing

```
javac HelloWorld.java
```

  If the compile completed successfully, you should have generated a *class* file called `HelloWorld.class`. This contains the *bytecode* for your program which you can run on the Java virtual machine (JVM).

- Run your program using `java HelloWorld`

If all went well, you should see the output *Hello World* in the terminal window. At this point, you should try running the automated checker on your solution. Details are given in the section below: Running the Automated Checker. At this stage, only the build and first comparison test should pass.

Task 2: Enhancements

Extend your basic HelloWorld program to allow command line arguments to be accepted

- Open HelloWorld.java with the text editor.
- Change the code so that instead of always printing "Hello World", it prints "Hello" plus the first command line argument the program receives e.g.

```
java HelloWorld Bob
```

  will result in "Hello Bob" being displayed in the terminal window. If no argument is given, it should still print "Hello World".

Below are some examples of the expected output for some different command line arguments:

```
java HelloWorld
Hello World
```

```
java HelloWorld Bob
Hello Bob
```

```
java HelloWorld Michael and friends
Hello Michael
```

```
java HelloWorld "Michael and friends"
Hello Michael and friends
```

Hint: what do you think the parameter `String[] args` represents in the `main` method?

Running the Automated Checker

When you submit a practical, it will be checked automatically using the automated checker, `stacscheck`. You can try running this on your program before submitting it, to ensure that nothing has broken accidentally.

You will always upload an archive file of your work to MMS, so you should archive your work first before running `stacscheck` on it. Compress the whole of your p0-hello directory to a .zip file.

In order to run `stacscheck` on your code, you should upload it somewhere in your home directory on the lab computers. Then, access the terminal on the lab computers by using ssh. You can complete these two steps using the following commands from the terminal/command prompt:

```
sftp <yourusername>@<yourusername>.host.cs.st-andrews.ac.uk
ssh <yourusername>@<yourusername>.host.cs.st-andrews.ac.uk
```

Note that you'll need to use your CompSci password here – the same one you use for Discourse. If you're not used to `sftp`, you might prefer a graphical FTP client such as *FileZilla*.

Once you've uploaded your file and logged into the lab PCs via SSH, you can run the automated checking system on your archive by navigating to the archive's location and running the following command:

```
stacscheck --archive p0-hello.zip /cs/studres/CS5001/Practicals/p0-hello/Tests
```

If you have uploaded the original `p0-hello` directory instead of an archive, then you should instead navigate into it and run the following command:

```
stacscheck /cs/studres/CS5001/Practicals/p0-hello/Tests
```

If all goes well, then you should see output like this:

```
Testing CS5001 Practical 0 (Hello World)
- Looking for directory 'src': found in current directory
* BUILD TEST - basic/build : pass
* COMPARISON TEST - basic/Test01_HelloWorld/progRun-expected.out : pass
* COMPARISON TEST - basic/Test02_HelloWorld_Bob/progRun-expected.out : pass
* COMPARISON TEST - basic/Test03_HelloWorld_Michael_and_friends/progRun-expected.out
* COMPARISON TEST - basic/Test04_HelloWorld_MichaelAndFriends/progRun-expected.out
* INFO - basic/TestQ_CheckStyle/infoCheckStyle : pass
--- output ---
Starting audit...
Audit done.
```

If it is not going so well, here are some things to consider

- If the automated checker cannot be started, then you have most likely mis-typed the commands to invoke the checker shown above.
- If the automated checker runs but fails at the build stage, then no other tests can be conducted, so you will have to fix this issue first. Likely reasons for the build failure include:
  - executing the checker from some directory other than your assignment directory or specifying an incorrect path to the tests.
  - your program cannot be compiled as required by the checker and as specified above. Try to modify your program such that it can be compiled using the simple command `javac HelloWorld.java` from within the `p0-hello/src` directory.
- If the automated checker runs and the build succeeds, but all comparison tests fail, it could be that your program cannot be run using the simple command `java HelloWorld` from within the `src` directory in your submission. Make sure you have written a `main` method in your `HelloWorld` class and for the enhancement tasks try to ensure your program uses the `args` command-line arguments that are passed to your `main` method.
- The names of the comparison tests 1 to 4 indicate the command-line arguments being passed to your program. Each test expects your program to produce a specific output as outlined above. If one or more of the tests fail, then it may be that your

program has a bug or is simply not printing out exactly what is expected and shown in the sample execution runs above. Maybe you have included some additional debug messages in your output or additional new lines, these will also cause tests to fail. Try to ensure your output matches the one shown above exactly when executing your program from the command-line.

- You may see a warning about missing Javadoc strings. You should generally include Javadoc strings to explain your code, as discussed in lectures. However, there's no need to worry about it for this practical.
- The final test `TestQ_CheckStyle` runs a program called *Checkstyle* over your source code and uses a style adapted from our St Andrews coding style (informally known as the *Kirby Style*). You may receive a lot of output from the style checker for your program. In order to address these, you can look at the published guide at https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/programming-style.html#programming_style.

  The style recommends the use of spaces as opposed to TABs for indentation. Try to get used to doing this, because it's great for when working with others!
- If nothing is working and you don't know why, please don't suffer in silence, ask us. This is the first time you will have tried to use the automated checking system, so there are bound to be some issues.

Deliverables – Software

Hand in an archive of your work via MMS to the *P0* slot.

Marking

This exercise is not assessed so no marks will be given.

Lateness

This exercise is not assessed so no lateness penalties apply.

Good Academic Practice

The University policy on Good Academic Practice applies:

https://www.st-andrews.ac.uk/students/rules/academicpractice/