


KIT101 Programming Fundamentals

PP Task 1.2 Hello, World!

Overview


- Purpose:** Gain some more familiarity with the DrJava development environment.
- Task:** Implement and modify a Hello World program, the first program every new programmer writes.
- Learning Outcomes:** 1 
- Time:** Complete this task before or during your first tutorial, in Week 2.
- Resources:**
- Introductory Programming Notes:
 - 02 Computers & Programming Languages

Submission Details

Upload the following to the MyLO submission folder for this task:

- Your HelloWorld.java source file
- A screenshot of the output from your program when it is run (visible in the bottom section of DrJava)

Assessment Criteria

- A  Completed submission will:
- Exhibit good code style by matching the provided code's layout and indentation
 - Have your name in the comment at the top, after the text `@author`
 - Show (in the screenshot) that the code compiles and runs
 - Display your customised message

Instructions

Work through the following steps to create the classical [‘Hello World’ program](#).

1. Open the Windows **Start menu** and start typing DrJava. When the search box displays a match press Enter.

If the program prompts you to download a newer version just click OK to dismiss the message. You are using the most recent version already.

DrJava provides a text editor and an interface to the Java Development Kit's (JDK's) commands `javac` (which compiles Java source code) and `java` (which executes Java programs). Since a Java source file is just text, you could use any text editor you like, such as Notepad, Sublime Text, Emacs, Vim, etc.

By default, DrJava starts with a new blank document.

2. Type (or paste) in the following program, remembering to save it. If you use copy-paste and the indentation is lost then use the tab key on the affected lines to make it match what you see below.

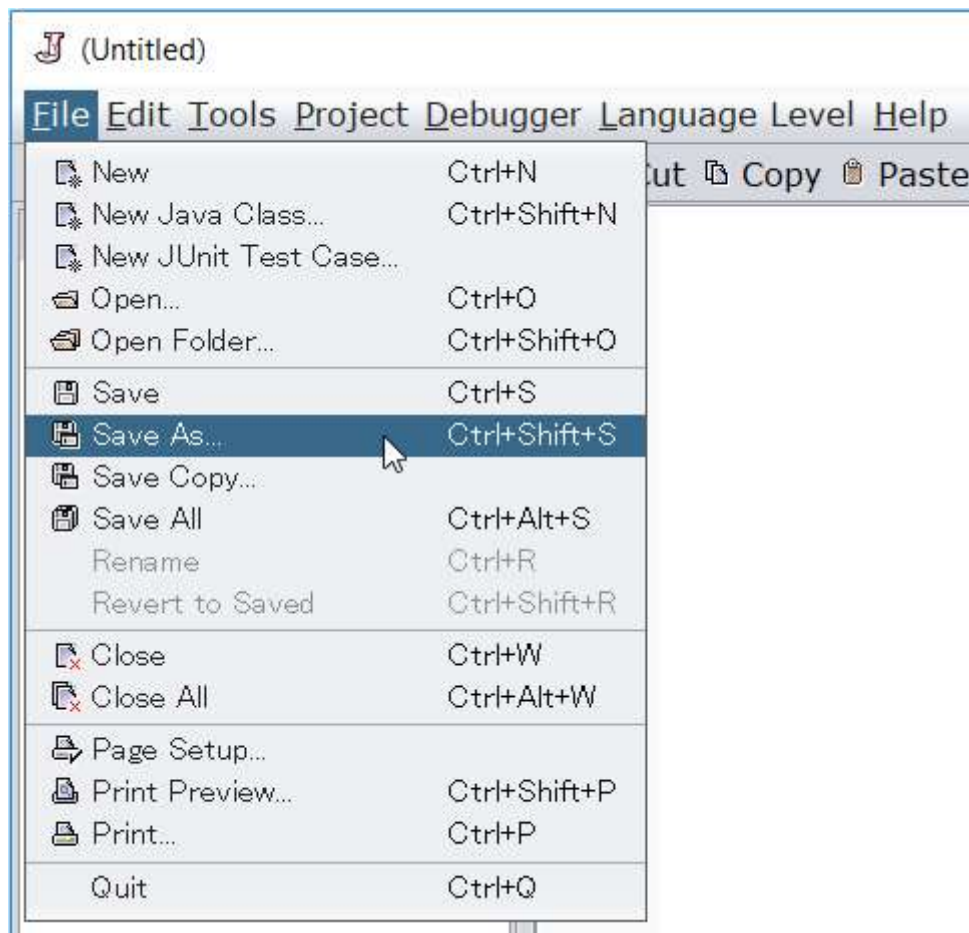
```
/**
 * KIT101 1.2PP Hello World
 * @author YOUR NAME
 */
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Note: Code editors (and many of the notes you will receive in this unit) use colour to indicate the meaning of different parts of code. This is called *syntax highlighting*. You do not have to—indeed, you cannot—type green or blue text. The editor will do that for you. As you learn what the colours represent you will be able to spot some syntax errors because the colours will look ‘wrong’.

3. Replace the text `YOUR NAME` with your name.

Tip: Java comments beginning with `/**` can be used to produce documentation (and so are called “javadoc comments”). `@author` is one of a number of special entries in these comments, and indicates the author of a program.

4. Save the file by choosing **Save** or **Save As...** from the **File** menu as shown below.



Save the file as "HelloWorld.java" (the .java extension will be added for you if you forget) somewhere in your personal folder (P:). DrJava should have correctly filled in the name by reading the text after **public class** in your source code, but make sure that the H and W are capitalised. You'll learn why this is important soon.

In future, save your file as soon as you've created the outer structure of the program (the code between **public class** and the final closing brace `}`).

5. Click **Compile** on the toolbar. Behind the scenes, DrJava is using the `javac` command to compile your program to machine-executable bytecode. Note the text in the Compiler Output window at the bottom of the screen; it should only say "Compilation completed." since there are no errors in the code. If there are error messages displayed then you will need to return to the editor and fix the typing mistakes.
6. In an Explorer window navigate to the location you saved your .java file. Are there any new files now? What do you think they contain?

7. Return to DrJava. To execute (run) your program click **Run** on the toolbar. Is the output shown in the window at the bottom what you expect?

Activity: Explore options for creating a new program

Experiment (for just a few minutes) with the File menu command **New Java Class...** There are a lot of options you can ignore for the moment, but try the following two:

1. Select **New Java Class...** and enter a name containing no spaces (but not HelloWorld) in the Class Name field, then click OK. *What text is entered for you in the editor window?*
2. Select **New Java Class...** again, enter another Class Name, but also select "Include main method". *What text appears in the editor now? Does it look similar to anything else you've seen today?*

The various options in the new class dialog window are actually just ways of having DrJava write some parts of the code for you, to save time and reduce the chance of errors. Before proceeding close the two new files you just created.

8. Now customise the message. Go back to editing your HelloWorld program. Find a short poem or piece of text you like that spans multiple lines. Write statements using `System.out.println()` to print each line in order.

Change the order of the `println()` statements, recompile and run your program. Is the output what you expected? Now that the displayed message is different from the original text you chose, would this be classified as a syntactic, semantic or runtime error?

Restore the order of the lines to what it should be.

9. **Compile and run the program, and take a screenshot** of its output. **Submit your source code and the screenshot** to the appropriate submission folder on MyLO.

Activity: *After you have submitted,* introduce some syntax errors

Introduce some errors and recompile the program. Try each of the following one at a time: introduce the error, try compiling, look at the error message(s) produced, then undo the change. Suggested errors:

- delete the `}` at the end of the file;
- misspell `"main"` as `"man"`;
- after `public class`, misspell `"HelloWorld"` as `"helloworld"`;
- delete a semicolon or two;

- delete the closing double quote (") on one of the `println()` statements; and
- misspell "args" as "double" (which is a reserved word). What happens if you change it to "arg" or "fred"?

It is important to become familiar with the error messages (some are not very intuitive) so that in the future you can deducing the likely cause of mistakes. Note that the error may be identified at the wrong location (the error may be earlier but cause the compiler to fail to understand something later), or cause a different error to be indicated. You need to think about what the compiler is telling you and some interpretation may be required.

You do not need to submit this practice.