

# ORIENTATION MANUAL

Computer Science 1016S

## 1. Introduction

This is a self-paced tutorial on the procedure for setting up and using a Windows computer to complete Computer Science 1 practical assignments and submit them online using the *Amathuba* learning management system.

There are screen snapshots along the way as well as explanations of what you are doing. It is recommended that you follow the tutorial in sequence from beginning to end. However, do feel free to deviate from the procedure to explore the functions available but try not to get lost – if necessary, ask for assistance from a tutor.

*Microsoft Windows 10* is the name of the operating system used for this tutorial; however, it should generally still be applicable to earlier versions of Windows. An operating system (OS) is a special software application that manages the resources of your computer. Major functions provided by an operating system include the ability to organise data and execute applications, both of which we will do in the following sections.

## 2. Policy on Academic Dishonesty for Computer Program Submissions

The first thing you will do is locate the CSC1016S course notes for students, read the Computer Science department policy on academic dishonesty for computer program submissions, and then complete and submit the policy acceptance form.

Execute a Web browser application such as *Google Chrome* from the desktop or from the *Windows* menu. Enter the URL '<http://amathuba.uct.ac.za/>' to go to the *Amathuba* website.

Enter your username and password and you will be able to log in. Then click on the widget on the left under the 2023 tab that corresponds to “CSC1016S 2023 | Computer Science 1016” and you will arrive at the front page of the website.

The notes to students are under *Content* tab of the website. Click on *Lesson 1 – Introduction* in the left-hand menu bar. An additional list of files and folders will be displayed under *Lesson 1* on the left panel. Click on the *Computer Science 1016S – 2023 Notes to Students* folder, then the document that we want bears the name '*csc1016s\_2023\_notesv1.pdf*' which will be displayed on the main panel.

Click on the other links and familiarise yourself with the contents. In particular, read through the policy document that appears immediately after the *Notes to students* folder.

Now you need to complete and submit the policy acceptance form. You'll find this in the assignments section of the site. Click on *Activities* from the menu items on top of the page then *Assignments*.

In the table that appears in the main pane you will see an entry titled *Assignment 1: Orientation Academic Dishonesty for Computer Program Submissions Policy Acceptance*. Click on it.

At the top of the page is the following text:

Please read the attached "Policy on Academic Dishonesty for Computer Program Submissions" and then accept the conditions by filling in your name, student number and date into the text-box below and then click "Submit".

\*\*\*\*\*

## Academic Dishonesty for Computer Program Submissions

### Policy Acceptance

I hereby acknowledge that I have read and understood the policy of the Department of Computer Science regarding academic dishonesty of computer program submissions. I will adhere to this policy and the general policies of the university referred to therein.

\*\*\*\*\*

At the bottom of the page is a text box. Enter your name, student number and date e.g.

Lebeko Poulo,

PLXLEB003

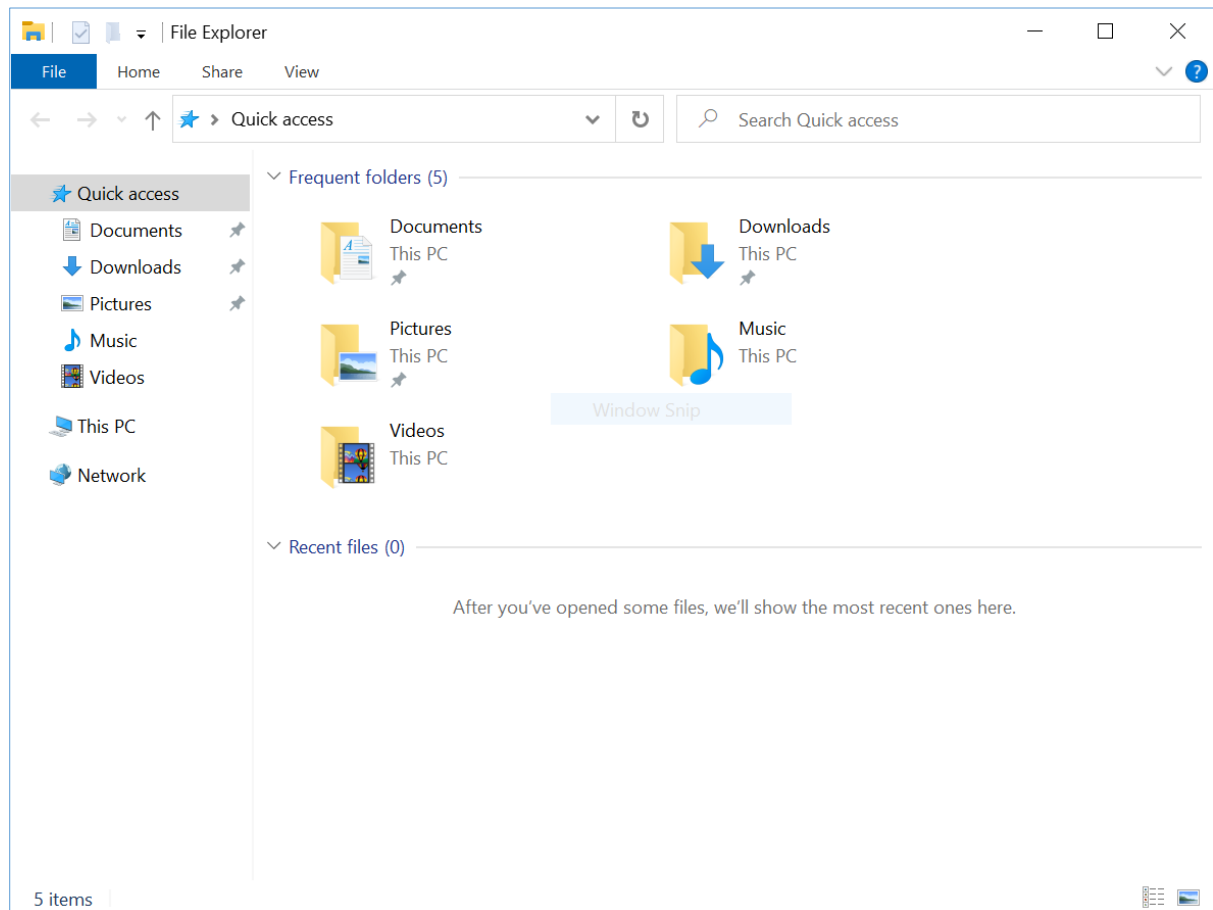
12/07/2022

When you're done, click on the *Submit* button to indicate that you accept the conditions.

### 3. File and Folders

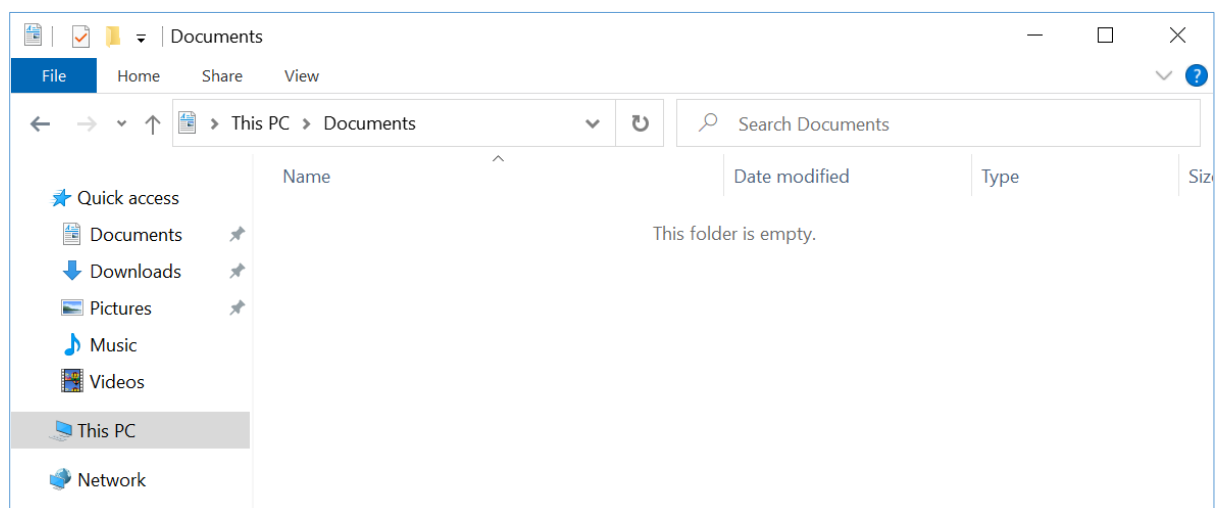
Any data on your computer is actually a collection of bits (0s and 1s) that are stored on some **storage device** (such as a **hard drive**) and given a name. In order to differentiate one piece of data from another, we refer to each piece of data as a **file** and give it a reasonably unique name, hereafter called the **filename**. Obviously if we have a lot of files it could be difficult to come up with unique names or even to find anything useful. To solve both these problems, files are arranged by the computer into **folders** or **directories**. Just like folders in filing cabinets, these computer folders may themselves contain files or even other folders. This leads to a hierarchical storage system that many believe is a useful method of organising lots of information.

If you double-click on the *Folder* icon (on the task bar at the bottom of the screen), you will see something like the following:



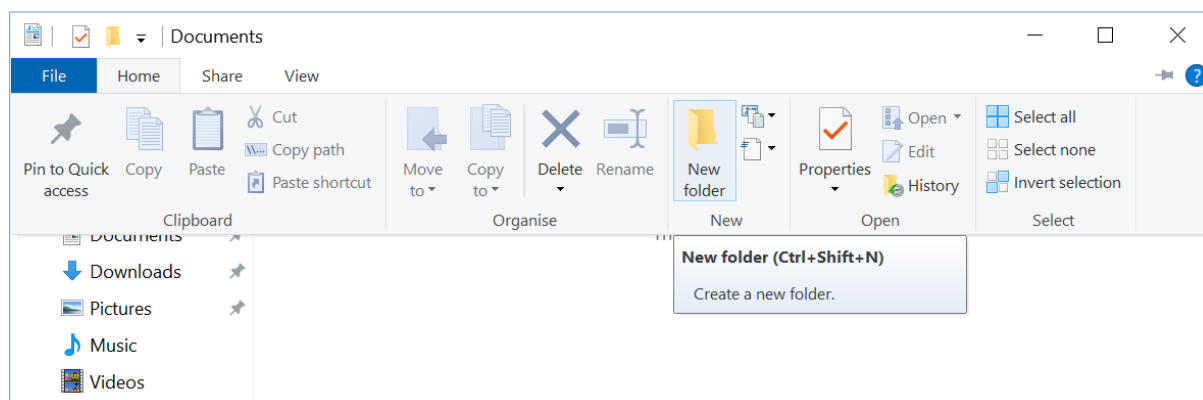
We will be using the 'Downloads' and 'Documents' folders in this tutorial.

Double-click on the 'Documents' folder and you will see its contents.

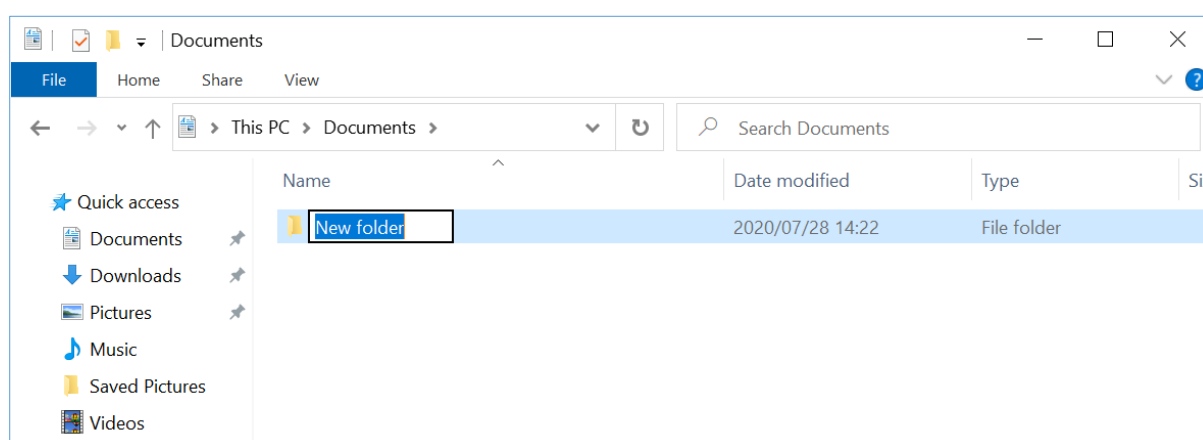


First, we will create some new folders to organise the programs we write during the course and make sure that everything can be easily found in future.

Click on the 'Home' tab and then the 'New folder' icon (in the centre near the top of the window).

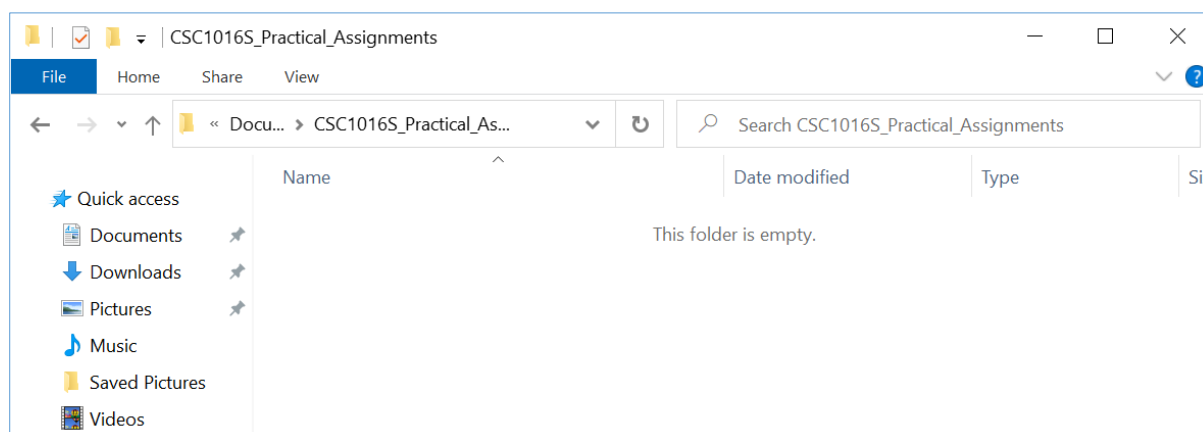


A new folder will be created, and the text is selected so you can enter a new name (the default name is *New Folder*).

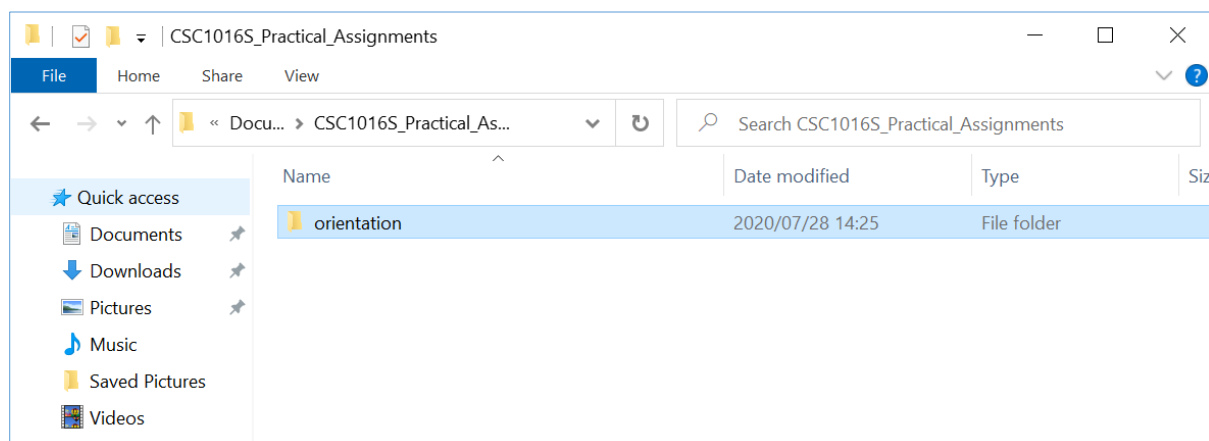


Type the name *csc1016s\_Practical\_Assignments* (or something else that you think is suitable) and press the *Enter* key. The folder will be renamed, and the new name will be displayed.

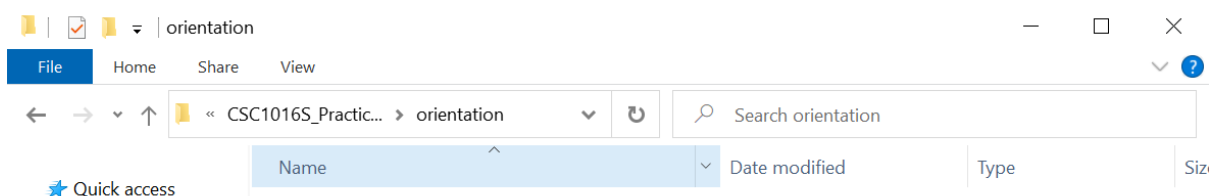
Now double-click on the new folder that you have created, and you will be able to see the list of files it contains.



As you can expect, the folder is empty as we have just created it! Now using the same procedure, create a folder here called *orientation*.



Double-click to change to this folder. Note that the **address bar** always lists the current location in the hierarchical file system.



## 4. Installing Software

Now we need to install the software, the computer programs that we will be using on the course. There are two things: The Java compiler, and the *jGrasp* integrated development environment.

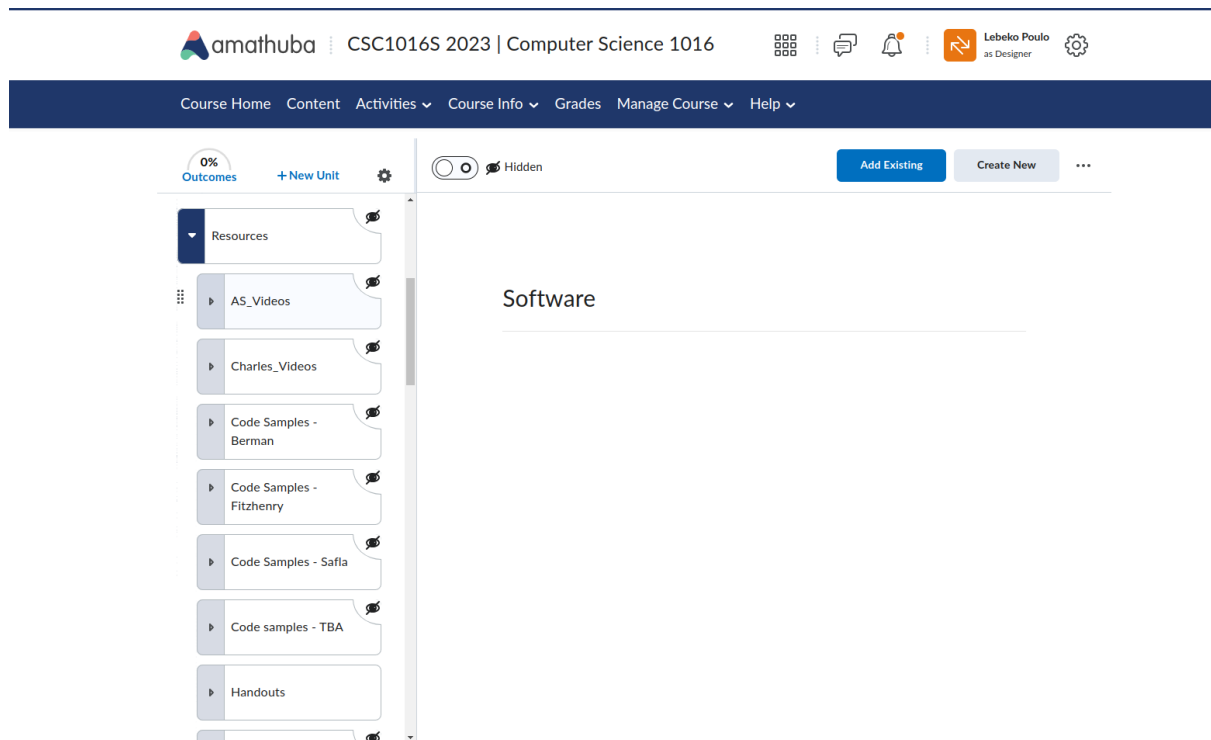
The aim of programming is to give the computer a series of instructions to solve a problem. Alas, the language understood by computers – **machine language** – is very difficult for humans to comprehend. So, we use a simpler programming language and then somehow translate the program from what we can understand into something that the computer can understand. These human-understandable languages are called **high-level languages**, while machine language is a **low-level language**.

**Java** is the high-level programming language used in this course.

Programs written in Java need to be **compiled** (translated) into some form of machine language before we can ask the computer to execute our instructions and for this, we use a **Java compiler**. However, before a program can be compiled, it has to be stored in a file – to do this we can use a **text editor**.

Since programming happens in stages, the programmer has to switch quite a lot between a text editor and a compiler – this can quickly get tiring. To make our lives much easier, the compiler and text editor are usually combined into a single application called an **Integrated Development Environment (IDE)**. The IDE recommended for use in this course is *JGrasp*, though you are welcome to use any other tools – even a separate text editor and compiler if you wish to simply do things the hard way ☺.

The software that we need for the course may be found under the *Content* tab in the Resources section of the CSC1016S Amathuba site. Assuming the site is still open in your browser, click 'Resources' then 'Software'.



**You will see a folder for *Java SDK* (the compiler) and a folder for *jGrasp*. Please note that these might not be the latest versions of these software packages. Please download the latest versions as follows:**

Java SDK: <https://www.oracle.com/za/java/technologies/downloads/#jdk20-linux>

jGrasp: [https://spider.eng.auburn.edu/user-cgi/grasp/grasp.pl?dl=download\\_jgrasp.html](https://spider.eng.auburn.edu/user-cgi/grasp/grasp.pl?dl=download_jgrasp.html)

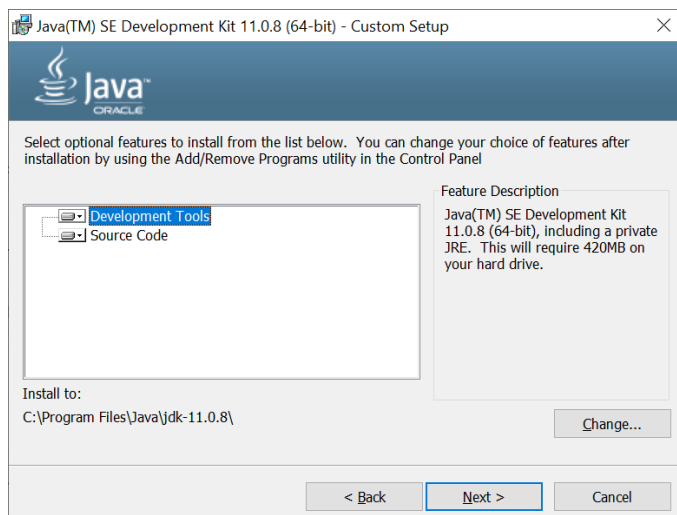
Or any other website where you can download these software packages. Install the correct one for your operating system and attempt the “hello world” example that follows later in this manual.

## 4.1 Installing the Java compiler

After downloading the Java SDK, you are now ready for the installation. When complete, click on it to launch (start) the installation.



The installation process is straightforward. Click 'next' to start. You will then be asked to select optional features to install.



Just click 'next' again. Once installation is complete you will be invited to access further resources. Just click 'close'.

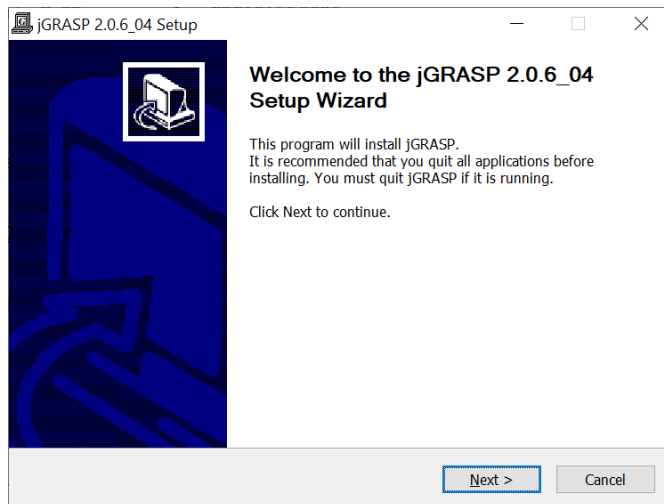




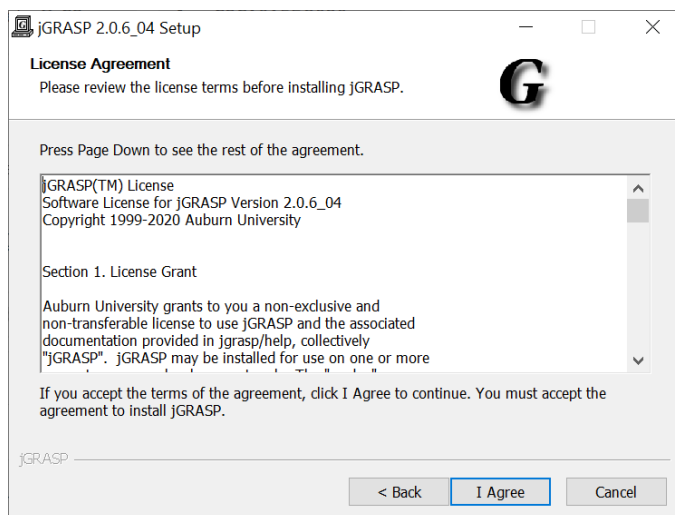
## 4.2 Installing the *jGrasp* IDE

To install the *jGrasp* IDE we follow a similar process. Go back to the location where you have saved the downloaded jGrasp installer. Double click on 'jGrasp'. And the installation will start. Note that the installation process is the same for both Windows and Mac OS users.

As before, there are versions for Windows, Linux and Mac OS. Make sure you have downloaded the correct one for your laptop's operating system.



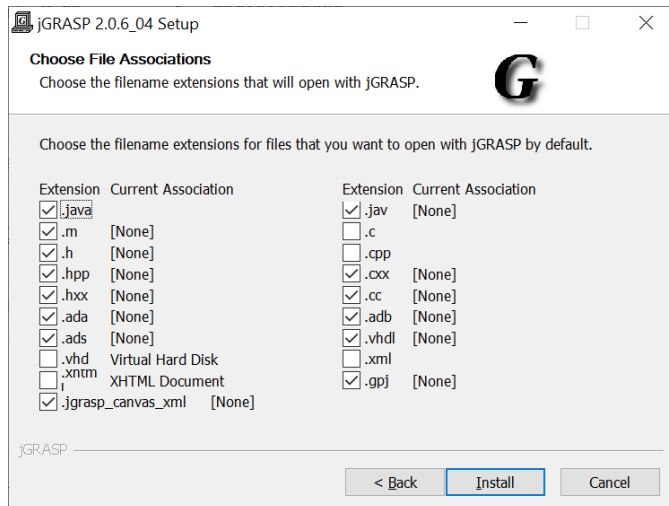
When you click 'Next' you are presented with the licence agreement



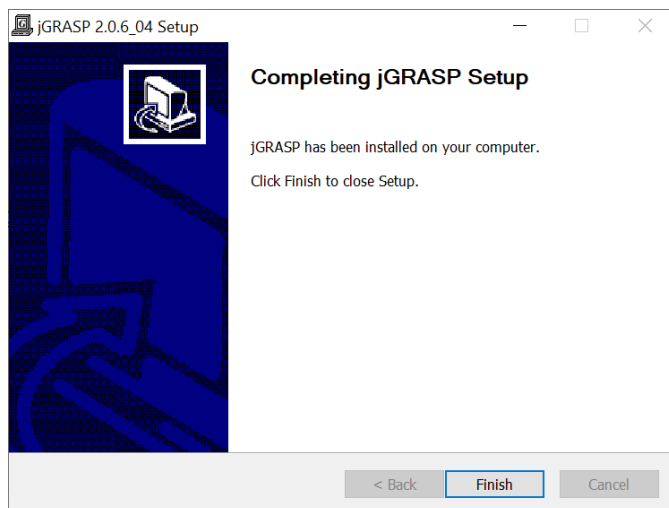
Clicking on 'I agree' leads you through a series of panels:

1. 'Choose Components',
2. 'Choose Install Location',
3. 'Choose Start Menu Folder'.

Click 'next' for each of these to arrive at 'Choose File Associations'.



Click 'Install' and then once installation is complete, click 'finish'.



## 5. Using the Java IDE

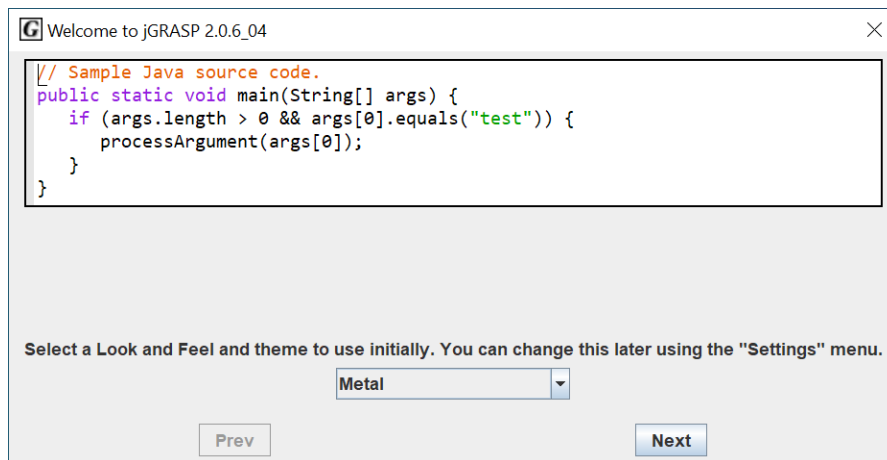
We will now write a simple program in *JGrasp* and compile and execute it.

### 5.1 Starting jGrasp for the first time.

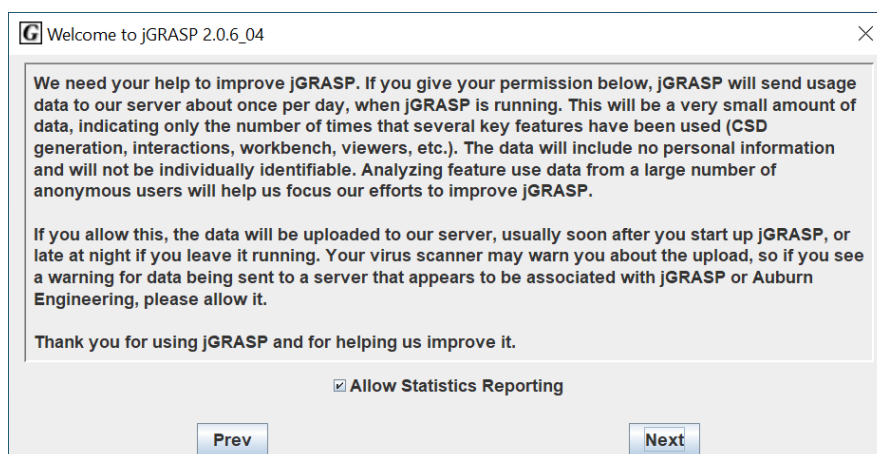
Run *JGrasp* from the Start Menu or double-click on the item ('icon') placed on the desktop by the installer.



Be patient. When *JGrasp* eventually starts up, it temporarily displays a **splash-screen**, which provides some information about the **application** as it is loaded into the computer's memory.

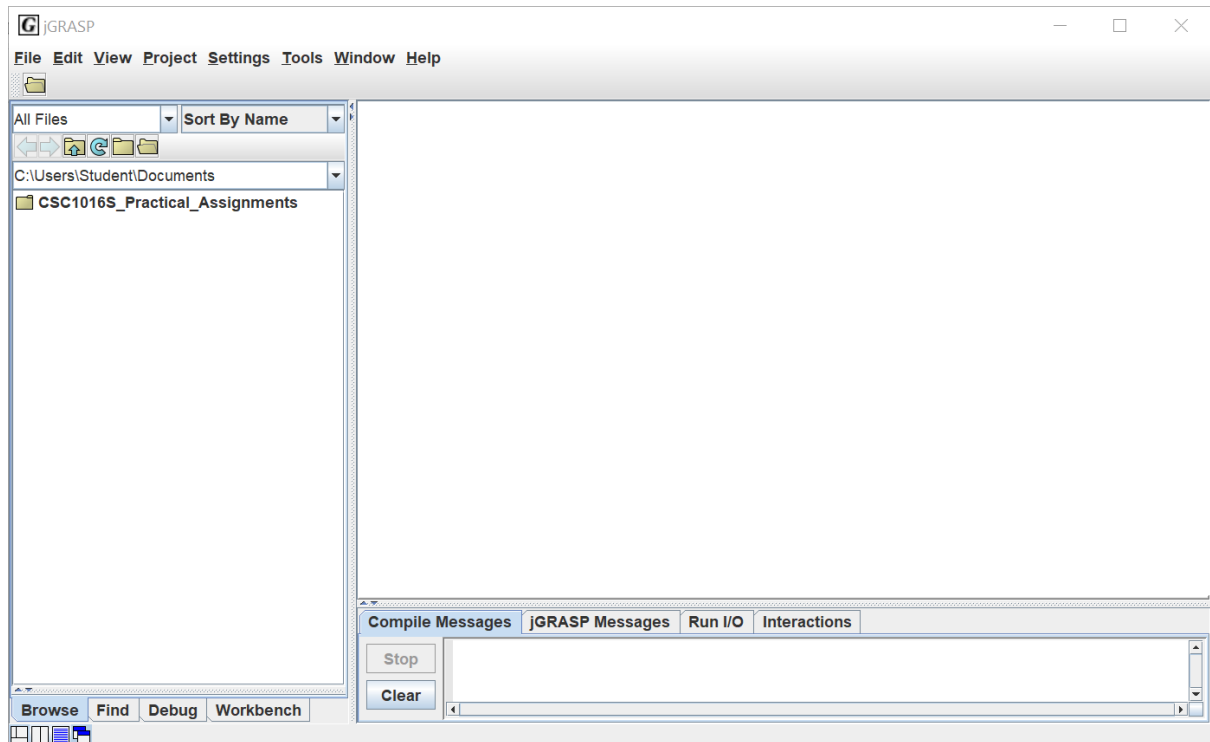


Once the splash-screen disappears, a window appears inviting you to choose the 'look and feel' of the program. Just click 'next'.



Now a window invites you to participate in the *jGrasp* improvement program. **Untick** the 'Allow Statistics Reporting' and then click 'Next'.

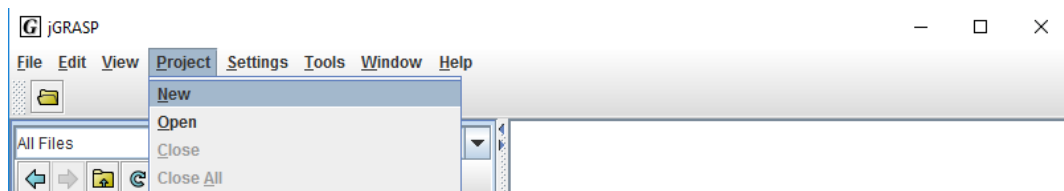
Finally, we reach the main IDE window/workspace



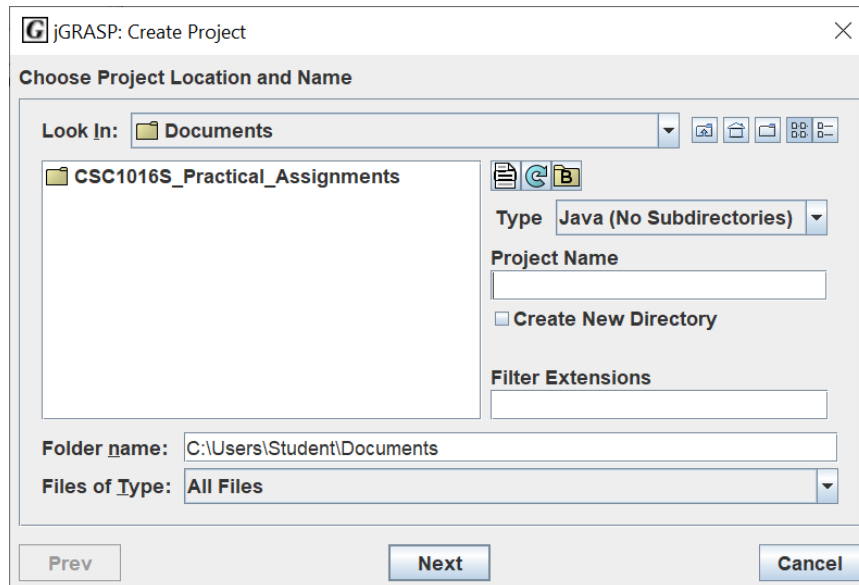
The *JGrasp* IDE has 4 **panes** (sections of a window) at start-up. At the top are **buttons** and a **menu** to activate various functions of the IDE. The left pane displays the contents of a folder – we will not use this right now. The right pane is an empty **canvas** (blank window) where editing of files will take place. At the bottom is a message pane where *JGrasp* displays messages for the programmer.

## 5.2 Writing a program

The first step in writing a program to specify that you are working on a new project. You will usually do this as the first step in a programming assignment. Click on *Project* on the menu, then *New*.

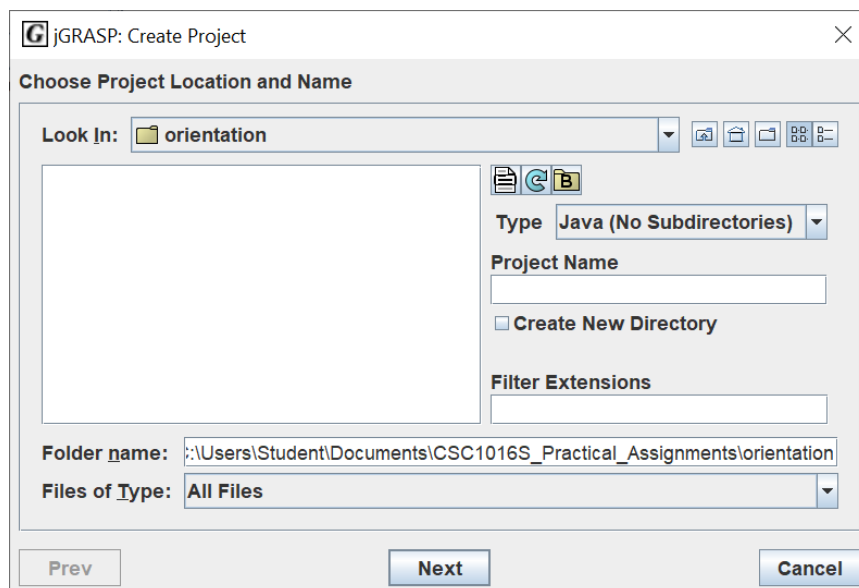


A window will be displayed where you can enter the name of the project.

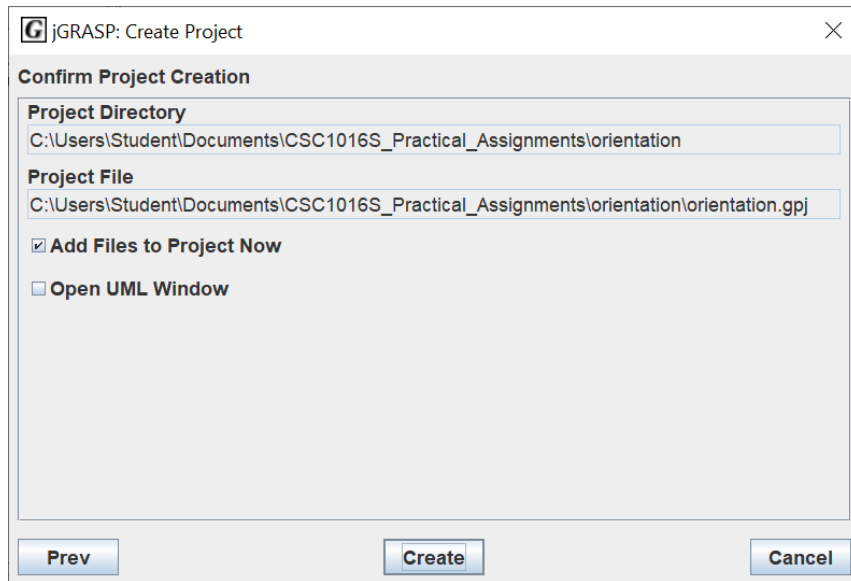


You want to make sure that you store the project in the folder you created earlier on.

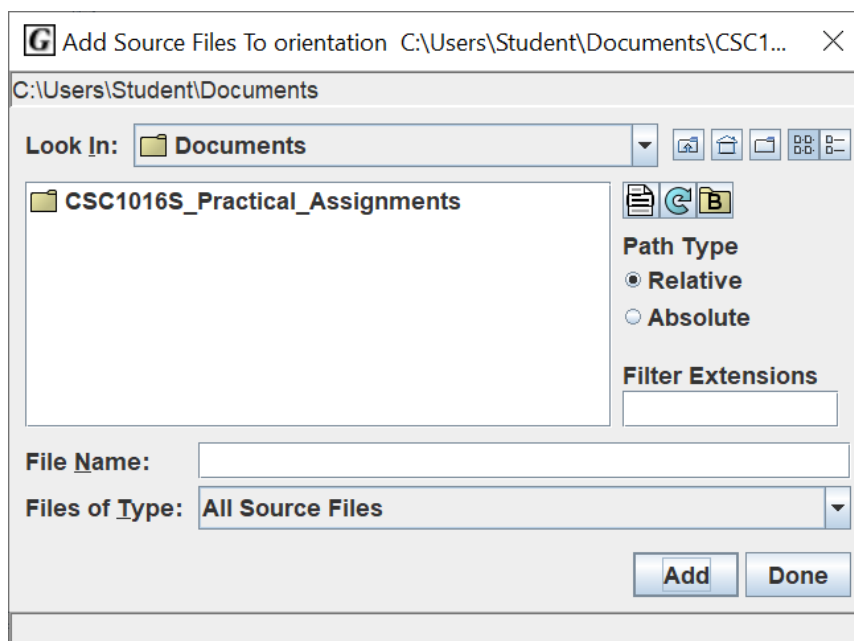
Remember that you created a folder called *orientation* within a folder called *1016s\_practical\_assignments*. So, double-click on *1016s\_practical\_assignments* and finally click on *orientation*.



Now that you are in the right place on your storage device, enter *orientation* in the box next to *Project Name*. To confirm the name, click on *Next*. You will be asked to confirm the creation of the project. Select *Add Files to Project Now* (the box may already be ticked) before clicking on *Create*.



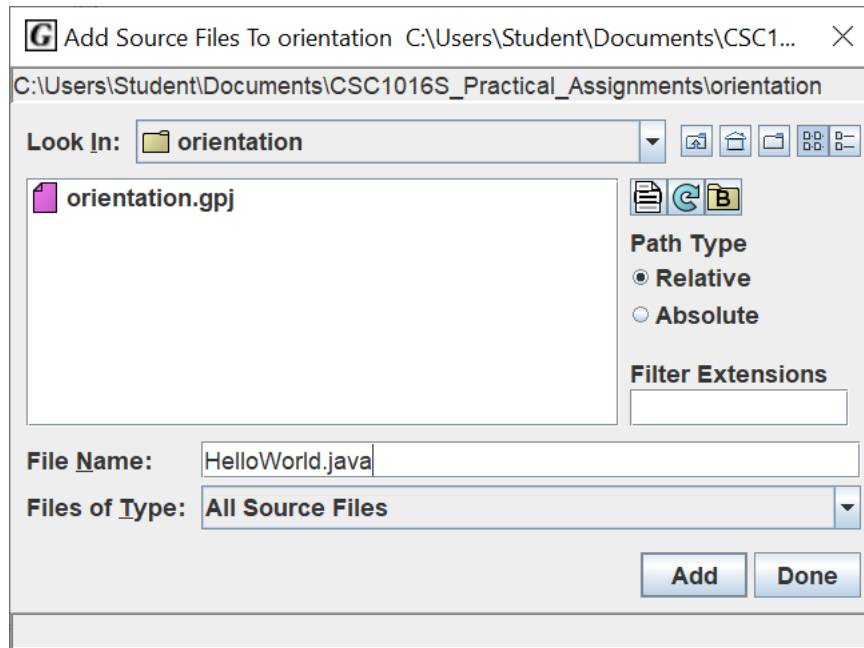
*JGrasp* now asks you for a list of the files that will store the Java program (notice that it says *Add Source Files* in the title bar).



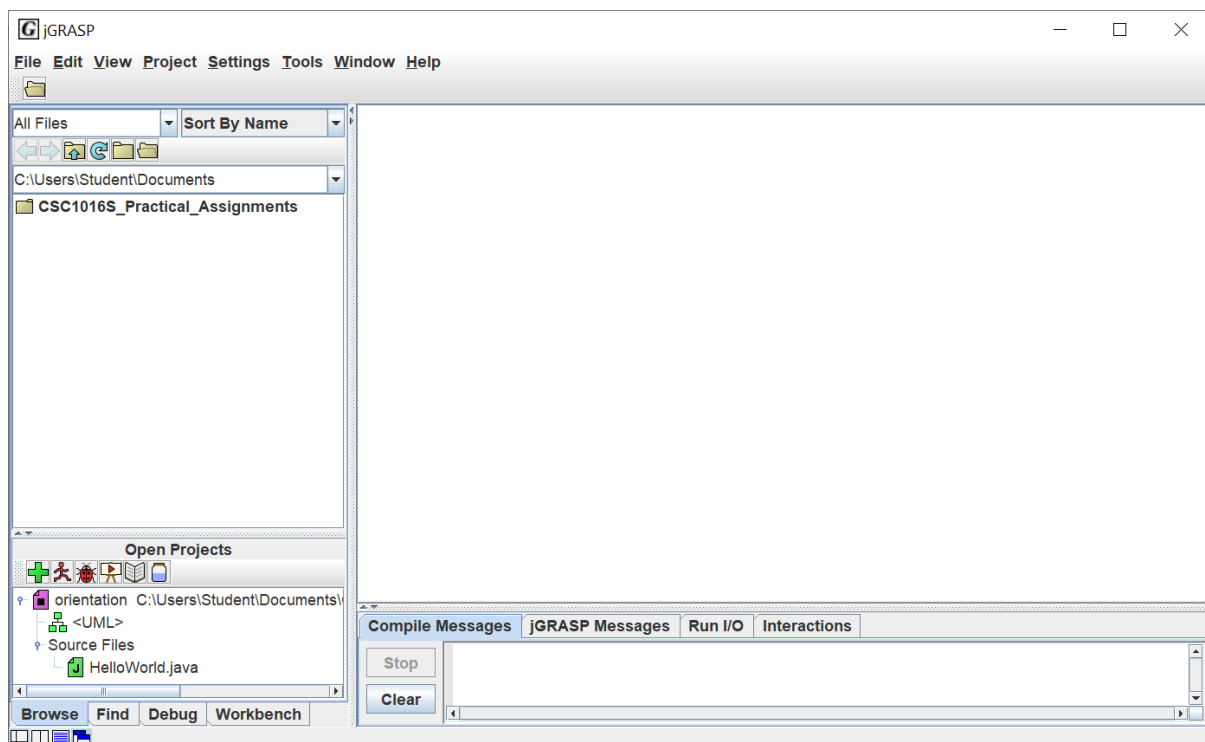
We usually write programs and store them in multiple files. There are many reasons for this, including that large files are difficult to manage. The project we have just created is therefore a mechanism for *JGrasp* to recombine the multiple files we create into a single software application.

*JGrasp* is asking for the names of the files that store the program **source code**, i.e., the program as written in the high-level language.

For this program there will be only one file. Double-click on *1016s\_practical\_assignments* then *orientation* and enter *HelloWorld.java* as the filename.

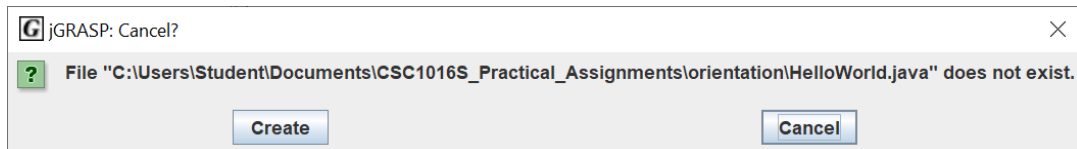


Click on *Add* and then *Done* (because you do not wish to add any more files).

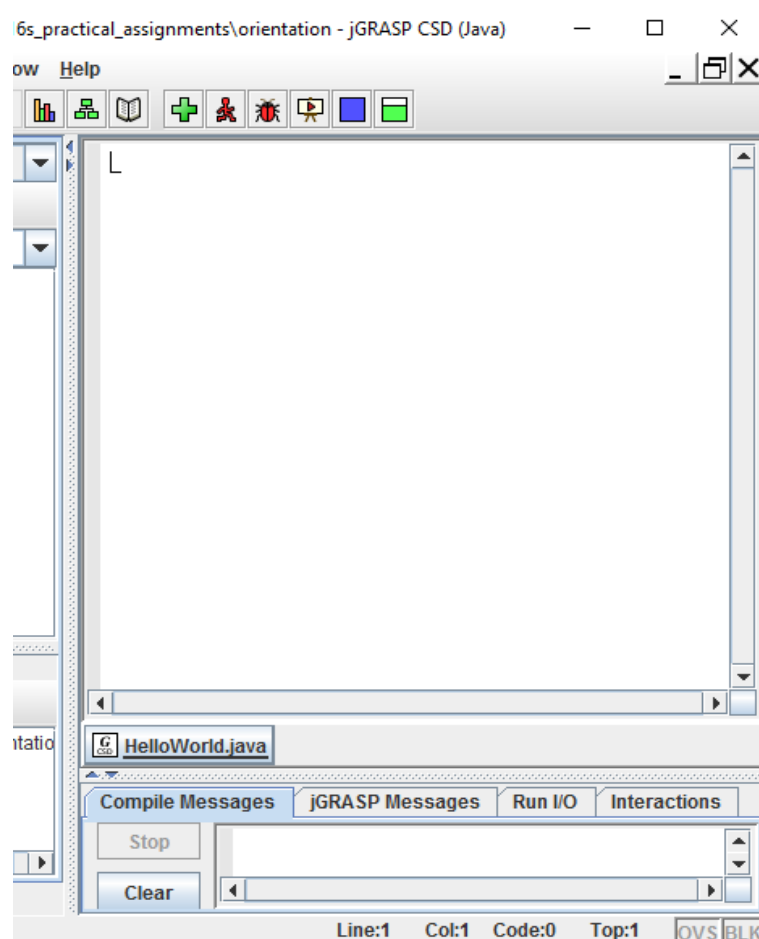


You are returned to the main IDE **workspace**. Now, just below the **folder view** there is a **Project view**, where the files contained in the project are all listed. (Besides *HelloWorld.java* there is an entry for *UML*, which can generate a formal description of the software automatically. This topic is discussed in class throughout the semester – now we are more interested in writing a simple program.)

Double-click on *HelloWorld.java* in the Project view.



*JGrasp* lets you know that the file does not yet exist and asks if it should create the file. Click on *Create*.



The edit panel becomes active. This is where you can edit the contents of the *HelloWorld.java* file that will contain the Java program. Note that below the pane on the left is a tab that indicates the name of the file. Now we are ready for some real programming!

Click anywhere on the white portion of the right pane to make sure your cursor is in the right place and then type in the following program exactly as it appears, but inserting your name, student number and date where specified:

```
// My very first Java program to display Hello World on screen
// Name:
// Student Number:
// Date:

class HelloWorld
{
    public static void main (String [] args )
    {
```



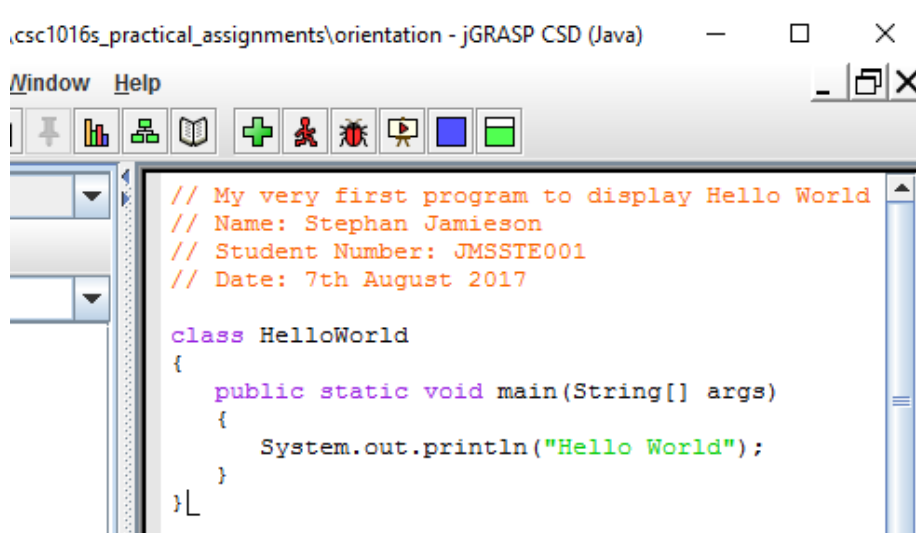
```

    System.out.println ("Hello World");
}
}

```

This small program simply prints out the words *Hello World* to the screen. It is classically the first program most people learn to write when learning a new programming language.

The **indentation** of some lines makes it easier for programmers to read and understand code – we say that it enhances **readability** and you ought to always do this when writing your own programs.

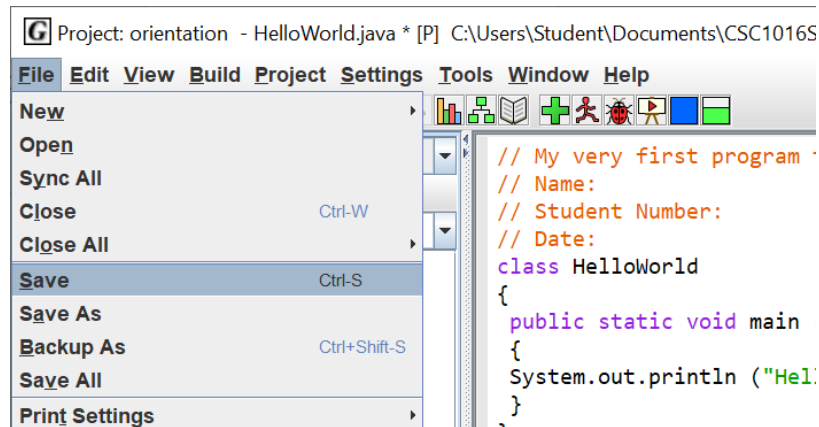


As you type the program in, *JGrasp* will highlight the words in different colours. This **syntax highlighting** helps programmers to understand their code more easily. The red lines are comments that are not instructions to the computer but help any readers of the program understand what it does, who wrote it, when it was written, etc. Comments are absolutely critical, and all your programming assignments **MUST** contain appropriate comments – this is elaborated upon further in the first chapter of the textbook.

The purple words are **reserved words** that have special meaning in Java so cannot be used by the programmer in any other way. Anything within quotation marks is green and the rest is black.

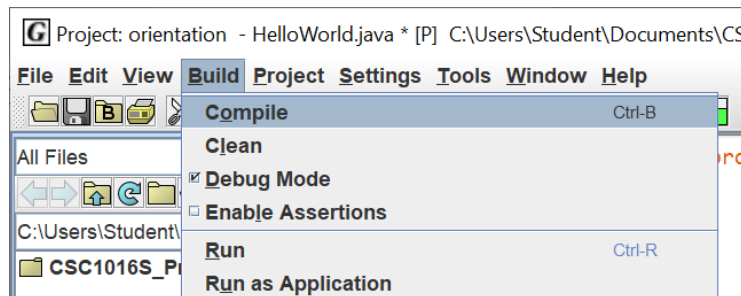
Having entered the program, the **FIRST** thing we **ALWAYS** do is **save** the file. This transfers the program from the computer's memory to the computer's hard drive, where it is stored permanently. Without saving, if there is a power failure (for which Eskom is infamous!), any changes made in the IDE will be lost! For longer programs, we save the file much more often – not just after writing the whole program. You also **MUST** save the file every time you modify it – your IDE may try to help by reminding you if you forget but don't rely on that.

Click on *File* and then *Save*.

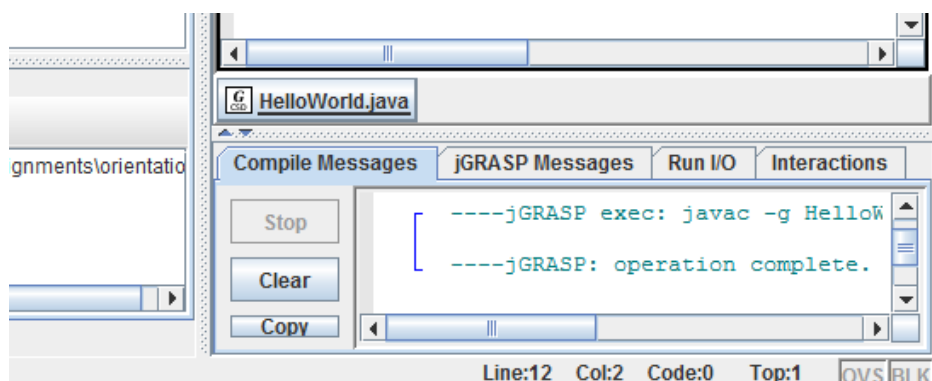


*JGrasp* saves the file – it does not ask you where it must put it because it already knows the filename (which is *HelloWorld.java* in this case).

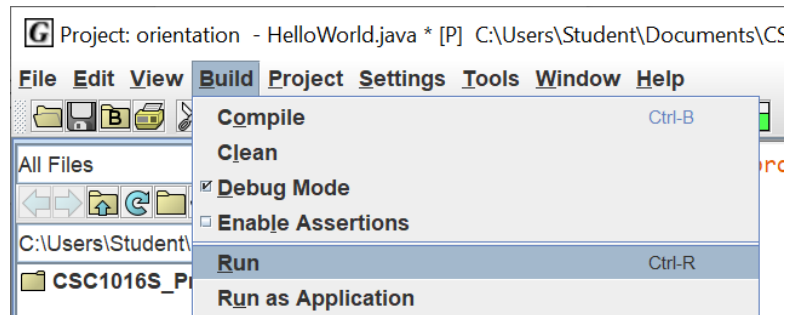
We can now compile the file (translate it from high-level source code to low-level machine language). Click on *Build* on the menu, then *Compile* (or the green cross in the toolbar).



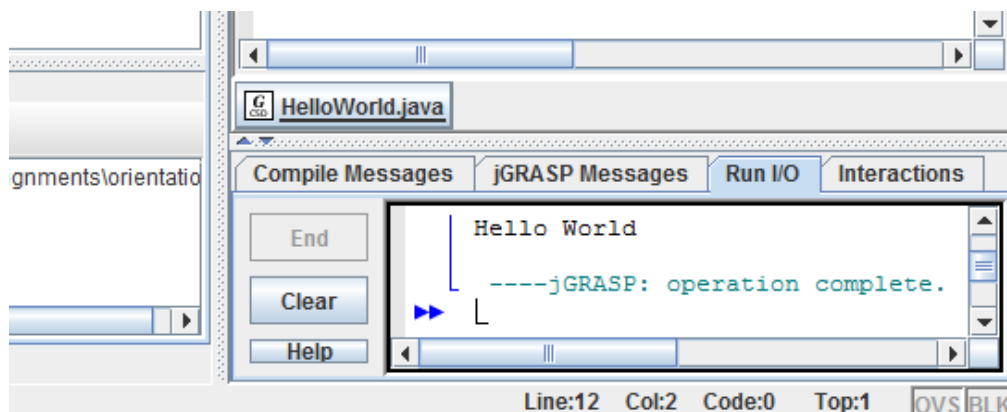
This process sometimes takes a few seconds the first time so be patient. *JGrasp* displays a message in the message pane to indicate that it has executed the compiler on the source file and since there are no other messages between *exec* and *operation complete* it means that the compiler could understand the program perfectly well. If there are errors, it means that the compiler could not understand the language used by the programmer – the programmer probably made an error. If you get error messages you need to correct the error in your program before proceeding.



Now that we have successfully compiled the program, we can execute it on the machine. Click *Build* on the menu then *Run* to **run** or **execute** the program (or use the red runner icon in the toolbar).



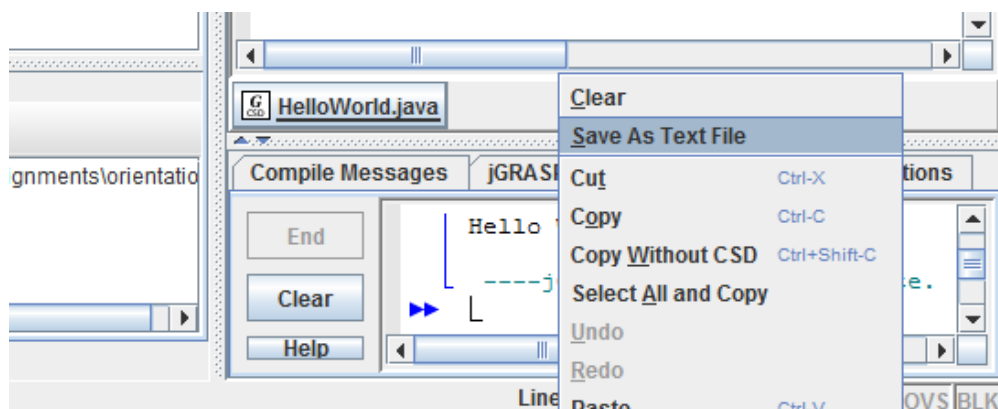
In the message pane, *JGrasp* has now switched to *Run I/O* mode and displays the results of executing the program. You should see the words *Hello World*.



Congratulations! You have written your first program in this course!

If you wanted to save the results of this program for some reason, say to impress your better half, you can easily save the **output** to a file. In this program, the only output that we generated was the words *Hello World* printed on the screen. We therefore save the contents of the *Run I/O* window to a file.

Right-click on the *Run I/O* pane and select *Save as Text File*.



When a window pops up to ask for a filename, remember to first change the folder to *orientation* like you did before. Then enter *HelloWorld.output* as a filename. *JGrasp* will then create a file and save the contents of the *Run I/O* pane to it. This is usually not necessary for CS1 assignments, but some courses may require that you submit output as well as the program you wrote.

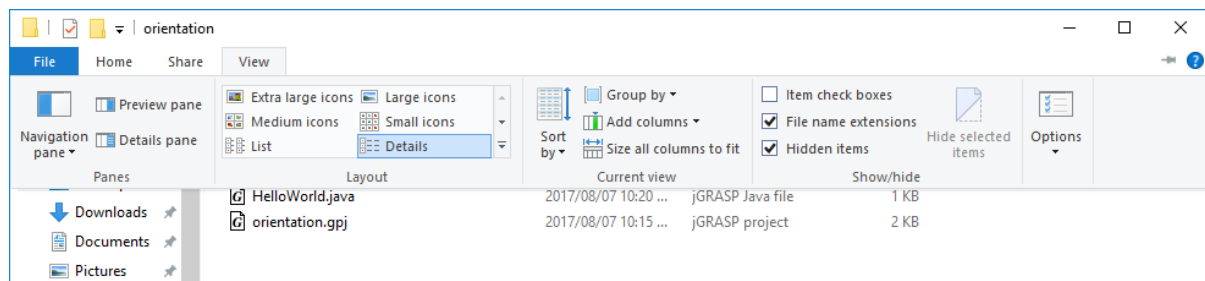
Close *JGrasp*.

## Creating a Zip File

Now that the programming is done, it is almost time to submit the files that constitute this program to the online assignment submission system on *Amathuba*.

If you have the *orientation* folder still open, you should see some other files in it. But there may be a small problem. In *JGrasp* we had created a file called *HelloWorld.java* but the folder may display several items called *HelloWorld*. Which do we want? The filename *HelloWorld.java* has an **extension** of '*java*'. These are the letters after the period and usually indicate what the purpose of the file is. Windows takes some liberties and tries to make folders more graphical by removing the old-style extensions and replacing them with little icons e.g., the *G* to the left of *HelloWorld*. For normal computer users, this is fine. Programmers, however, need a bit more control so we need to see the extensions.

Luckily, we can tell Windows 10 not to hide the extensions. Click on the *view* tab (at the top of the window, just below the title), and then select the tick box labelled '*File name extensions*'. The folder now displays the complete filenames for all files.



The folder now displays the complete filenames for all files. *HelloWorld.class* is the machine code version of the Java file after compilation. *orientation.gpj* contains information on the project.

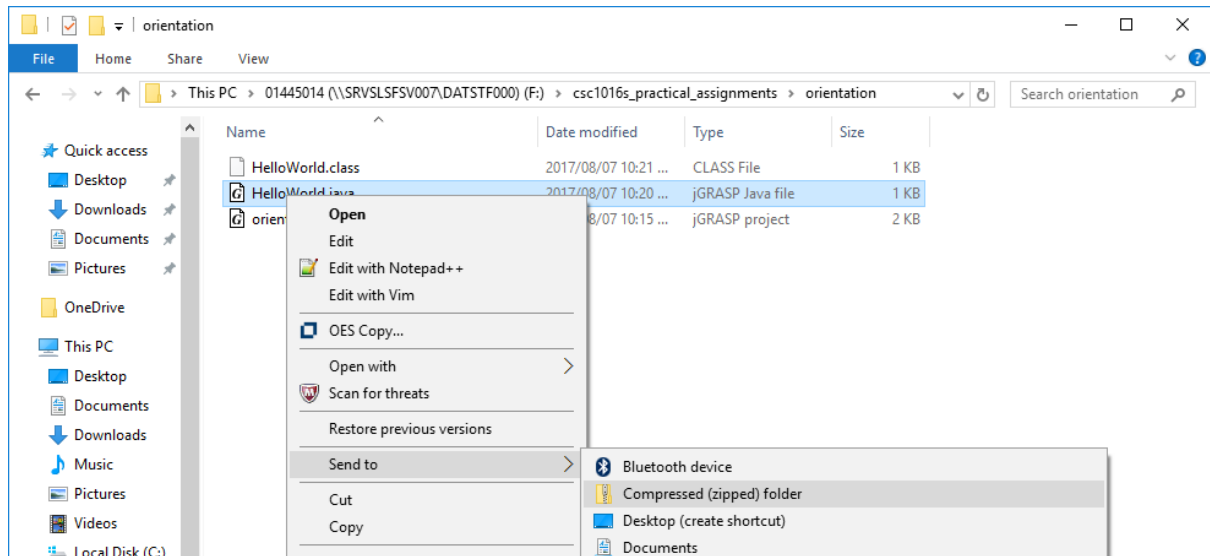
(If you chose to create the *HelloWorld.output* output file then you will see it listed as well. If you open this file in *WordPad* or *NotePad* you will see the output from your program.)

To submit an assignment, you can only send in one file. However, you need to send in at least all the source code files. To make this possible we first create a **Zip file** that can contain other files. This is almost like a folder, and in fact is sometimes referred to as a compressed folder – compressed since special algorithms are used to make it use less storage space than one would expect.

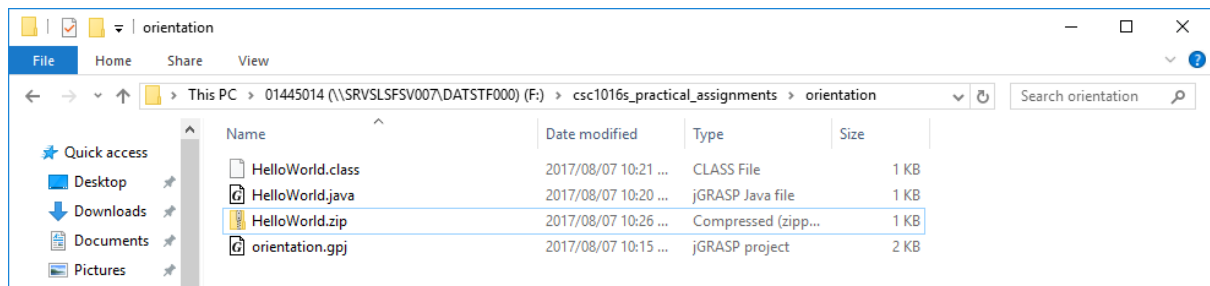
To create a Zip file, right-click on the *HelloWorld.java* file, select '*Send to*' and then click on '*Compressed (zipped) folder*'.<sup>1</sup>

---

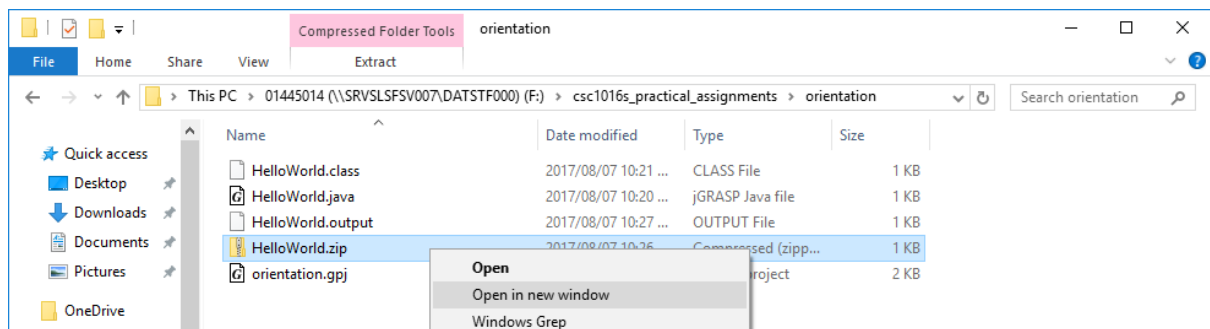
<sup>1</sup> Note that the screenshots on the following pages were captured on an ICTS computer and the address bar contains details that reflect that ('> This PC > 01445014 (\SRVSLSFV007.....'). So don't be concerned that it is different on your computer.



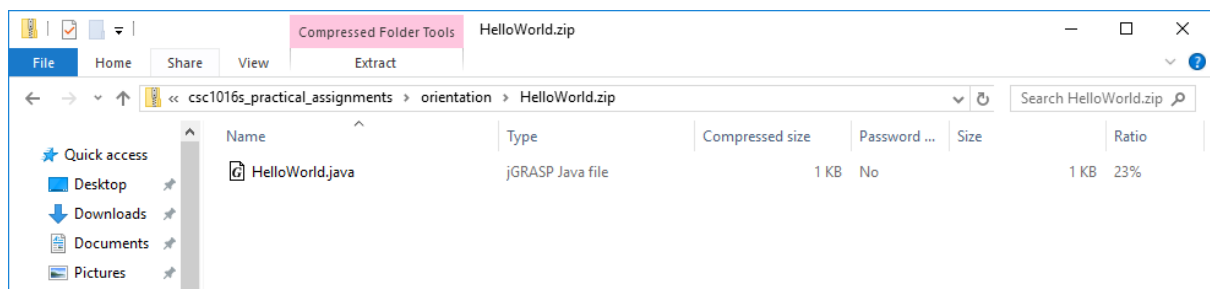
If you do not see the new file, press F5 to refresh the window:



The Zip file has a different icon that usually includes a little zip. Right-click on the Zip file and then click *Open in new window*.

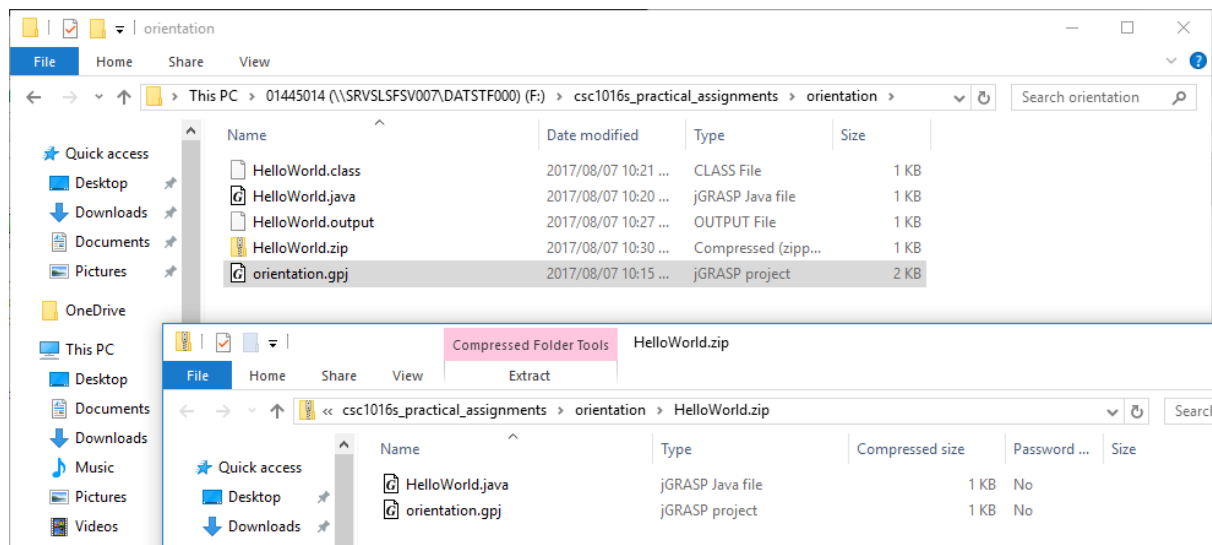


This opens the Zip **archive** to display what is inside.



There should be only the one file you initially selected.

Click on the partially hidden *orientation* folder as we need to put all the other files we want to submit into the Zip archive.



You want to add *orientation.gpj* (and maybe *HelloWorld.output*) to the Zip archive. We do not submit *.class* files with assignments as these can easily be recreated by opening the project in *JGrasp* and **building** the project.

Select the file and **drag-and-drop** it onto the Zip folder (that should still be open). Drag-and-drop means that you should click on the file and not let go of the button, then slowly move or **drag** the mouse over to the other window and let go of the button to **drop** the file at its destination.

(You can also select multiple files at once by holding down Ctrl on your keyboard while clicking on each of the required files. Then drag-and-drop the whole set at once.)

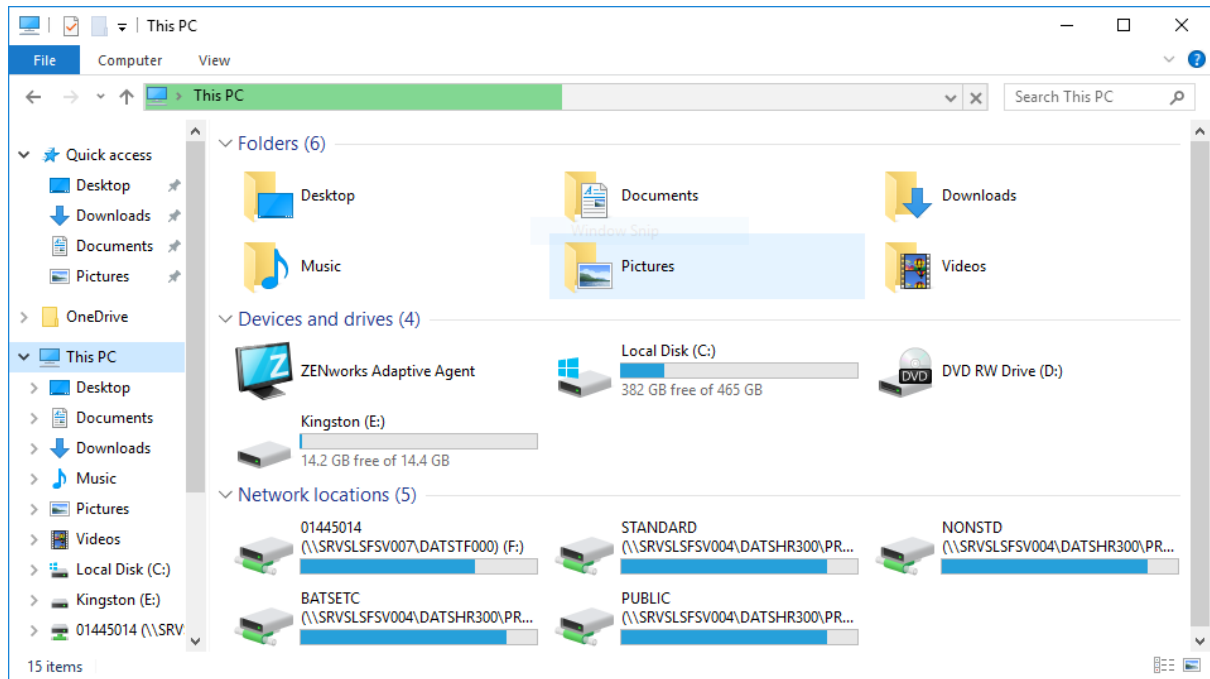
You can close the *HelloWord.zip* window.

The Zip file is complete and ready for submission. But it is good practice to first make sure the data cannot be lost accidentally by making a redundant copy or **backup**. If you do not have a flash drive or other backup device or disk, you should skip this step now but make sure you get a backup device very soon!

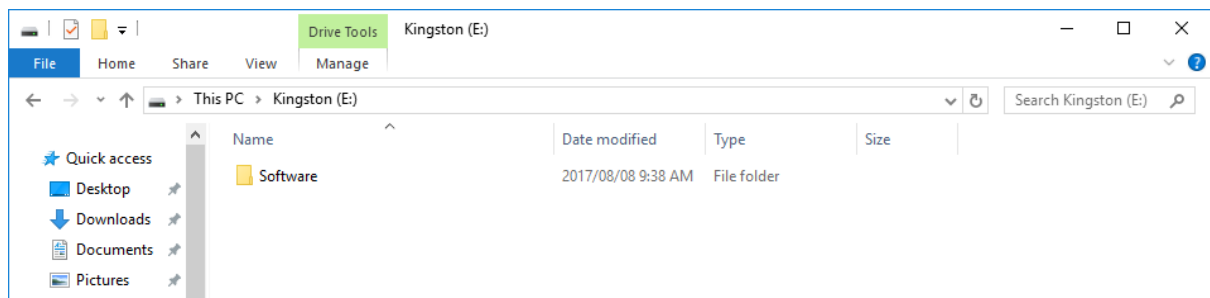
## Backups

A backup is a secondary copy of any data that you keep in case something goes wrong with the primary copy. You can keep multiple backups – some of your lecturers keep 4 copies of work in progress and up to 8 copies for really important data! Departmental policy also dictates that making redundant copies is completely your responsibility – so losing your data because a storage device was damaged is not an acceptable excuse for not handing in a practical assignment.

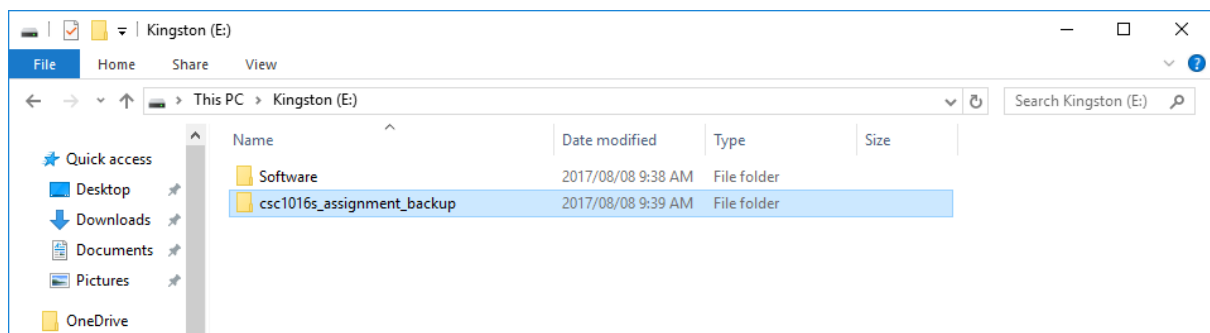
Backups are usually stored on removable storage devices such as floppy disks, USB flash drives and removable USB hard drives. When you connect a removable drive to the computer, you can usually immediately see it in the list of devices listed on *This PC*.



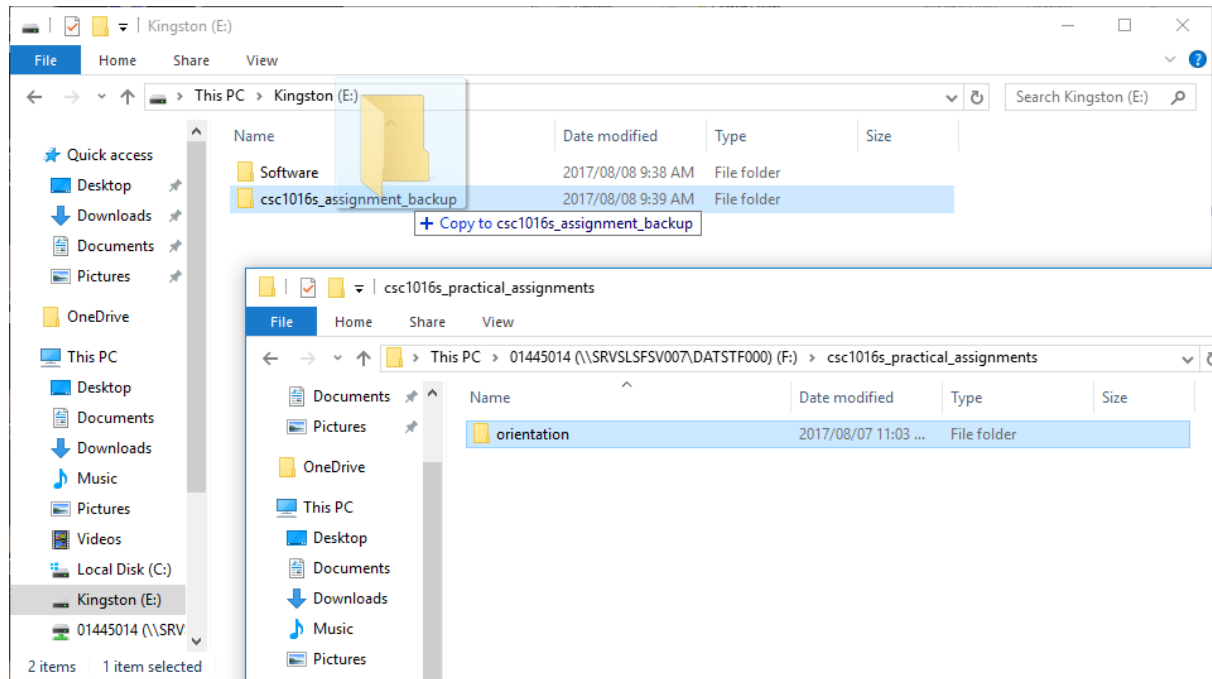
Double-clicking on such a device ('Kingston (E:)' in the screenshot) will bring up a view of the files and folders contained on that device:



Create a new folder where you can make backups of your assignments for this course. You could simply use the entire device or disk but it is likely that you want to store other data here as well.



With both your *csc1016s\_practical\_assignments* folder and backup device/folder open, drag-and-drop the *orientation* folder (Click *Back* on the F: drive window if necessary) onto the folder you just created within your backup device/folder. Microsoft Windows will copy the *orientation* folder and everything contained within it.



You can navigate into the backup folder and you should see an identical copy of the files on your F: drive. You should then remove the backup device/disk (If it is a flash drive, ask a tutor or friend for help if you have not done this before).

Now we are really ready to submit the assignment.

## Submission of Assignment

Assignments must be submitted on time. Any assignment that is handed in late is usually assessed a 10% penalty per day late, up to the 5<sup>th</sup> day – after that a mark of zero will be assigned!

Run a web browser like before and go to the *Amathuba* website. Log in and switch to the class website. Click on *Automatic Marker*.<sup>2</sup>

---

<sup>2</sup>Screenshots are from CSC1016S 2023, however, the process and details are unchanged.



[Course Home](#)
[Content](#)
[Activities](#)
[Course Info](#)
[Grades](#)
[Manage Course](#)
[Help](#)

# CSC1016S 2023 | Computer Science 1016

Course Content

Automatic Marker

Quick Eval

Welcome to the course!
  
[Launch the Quick Guide](#)

Announcements
  

## SIPP Pop Quiz

Maria Keet posted on 23 July 2023 3:40 PM

Dear all,

Welcome to CSC1016S! While for the most part of the course, you'll learn about object-oriented programming, we also want you to become acquainted with a few aspects of the broader context of software development with its Social

After clicking on the Automatic Marker Widget, you will be redirected to a new tab that will display the automarker (note that this is the same process that we followed in CSC1015F when submitting assignments).

[Submit](#) | [Edit](#) | [Manage Users](#) | [Marking](#) | [Review Marking](#) | [Batch Download](#) | [Log](#)

lebeko.poulo logged in as lebeko.poulo

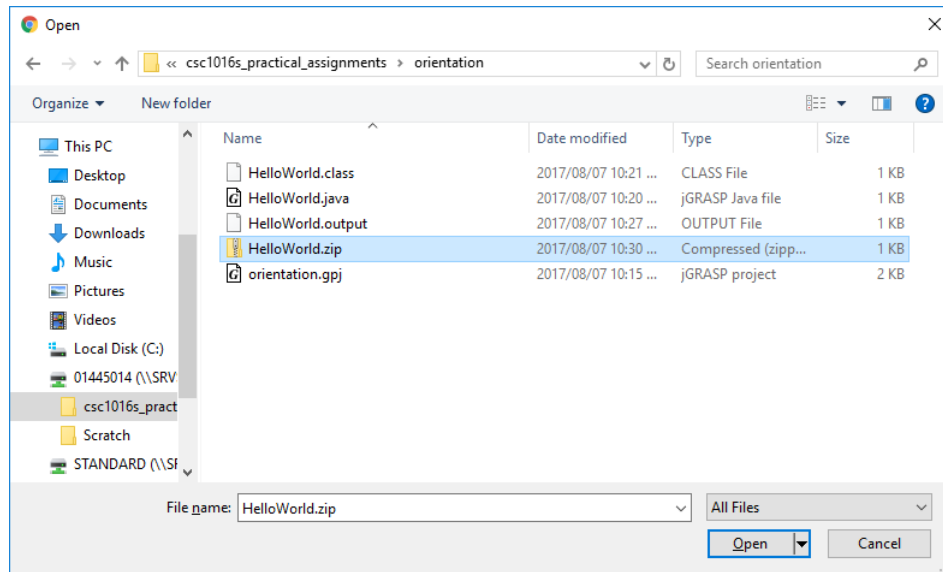
Open Assignments()

Assignment	Due Date	Attempts	Submissions
Closed Assignments			
Assignment	Due Date	Attempts	Submissions
Assignment 1: Orientation and Academic Dishonesty for Computer Program Submissions Policy Acceptance	2023-08-04T19:00:00	0 / 50	<a href="#">Choose file</a> 01421403.zip <a href="#">Submit</a>

A list of all open assignments is displayed, and you can submit to any assignment where a submit button is active.

For this orientation task, you want to use *Orientation*. The date indicates when the assignment is due and if your assignment is late this will be indicated by a change in colour, as well as the penalty you will be assessed for late submission. The number of attempts indicates how many times you have submitted this assignment and how many times you can submit it. In general, you can resubmit your assignment until you are satisfied with the result you obtain – the last mark is the one that we will use.

Click on *Browse* and choose your Zip file.



The output should be as shown above, but it is possible that something went wrong and instead you got errors and less than 100 marks. In this case, study the error report closely as it will suggest what went wrong. The automatic marker will indicate what the expected output was; what output was produced when your program was run; and the differences between the 2. The most common errors are incorrectly naming the files – that is, the automatic marker expects a particular name, but you used a different one – and zipping a directory rather than a set of files. Make sure that the files listed in the unzipping step are in fact what should be in the Zip file and that you have submitted a Zip file and not something else. If you are not able to fix the problem, ask for help.

You can also check that Amathuba knows about your mark – if you click on *Assignments* and then *Orientation* you will see the question that corresponds to this solution, and the mark you just obtained. In general, this section is where the questions for assignments can be obtained.

Amathuba's Assignment section also can be used to submit assignments that are not automatically marked – this is used in other courses and will be used for the final assignment in CSC1016S.

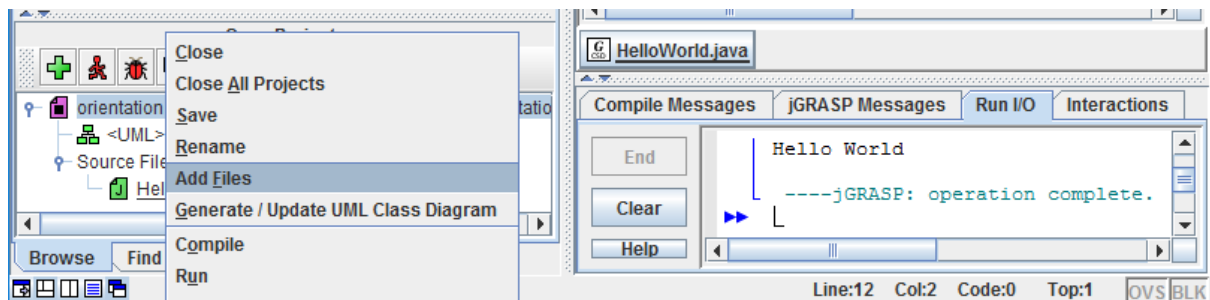
That's all folks! This is the end of the orientation exercise.

The following sections contain information that you can work through in your own time.

## Editing an Existing Project and Multiple Classes

If you want to modify an existing project, open *JGrasp* and select *Project / Open* from the menu. Then choose the project file and you should be presented with the project as before. If you are opening a project that you have not compiled, remember to **build** it before you **run** the program or *JGrasp* will display an error.

You can also add additional files to the project if your project has more than one class (most projects have more than one class). Right-click on the name of the project in the *Open Projects* pane and select *Add Files*.



Enter a name for the new class just like you did before. You should then see the second filename in the list of filenames, and you can create and/or edit the file as before.

## Java Documentation

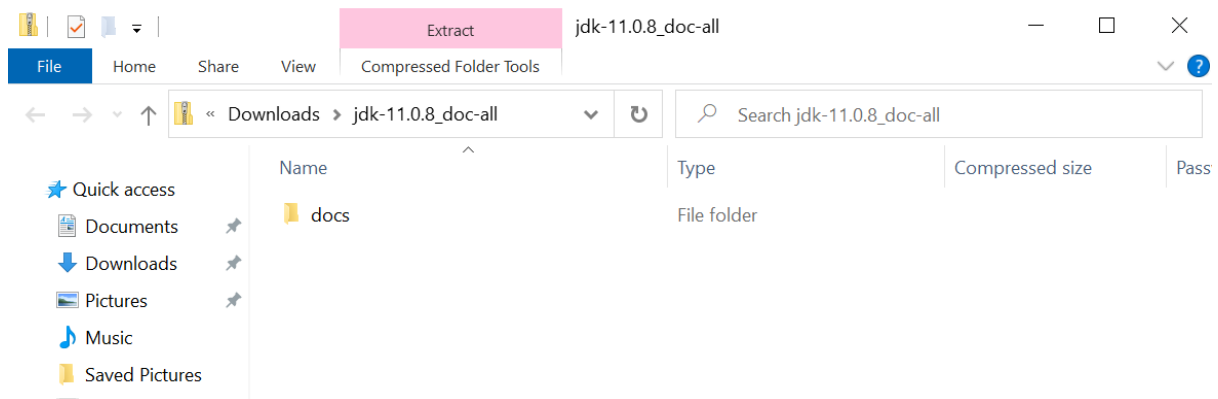
Much of the power of the Java language comes from the *extensive* library of pre-written components, the 'Java API'. As you become proficient in the language you will increasingly need to refer to the API documentation. To that end, it is available on the CSC1016S Amathuba site for download.

You must navigate to 'Content > Resources > Software > Java SDK > Java Documents All Platforms'.

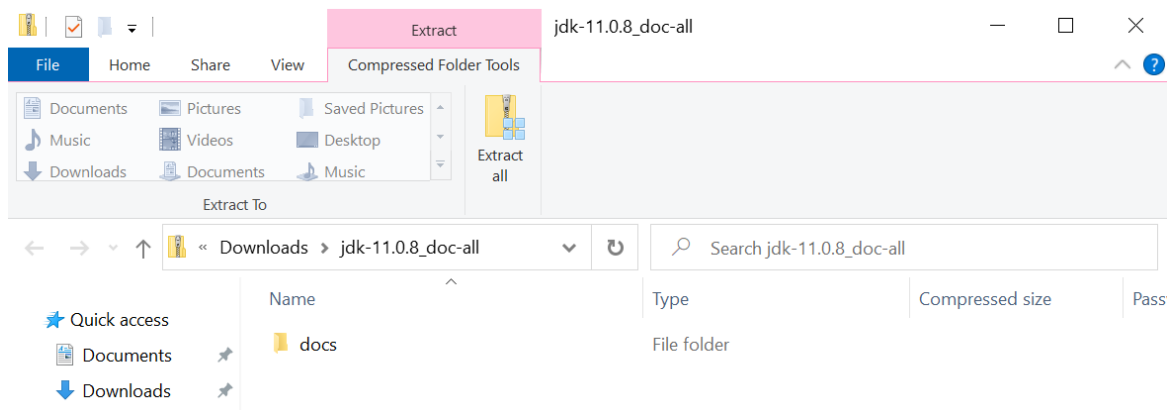
Click on the ZIP file 'jdk-11.0.8\_doc-all.zip' to download. Or you can search online for the most recent version of the ZIP archive.

s

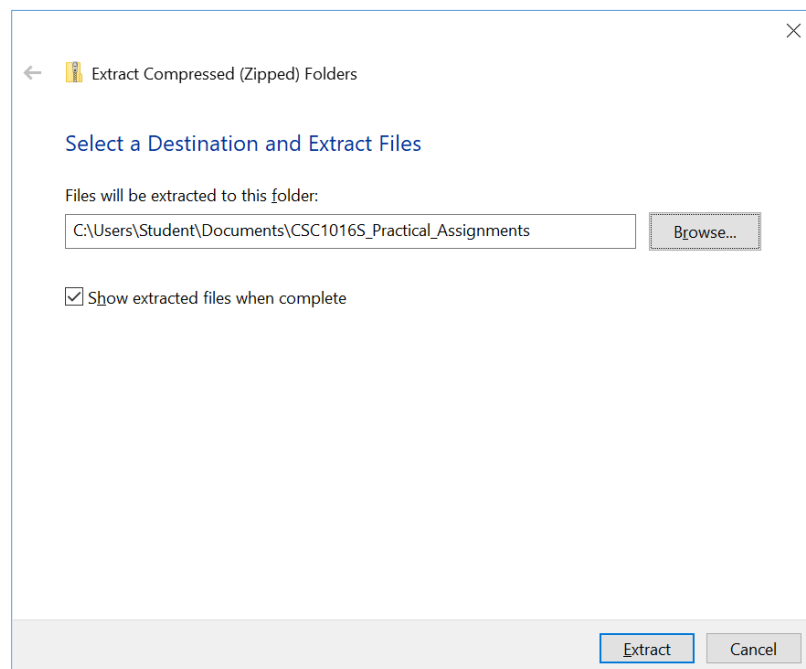
If the downloaded file is not automatically opened then click on it.



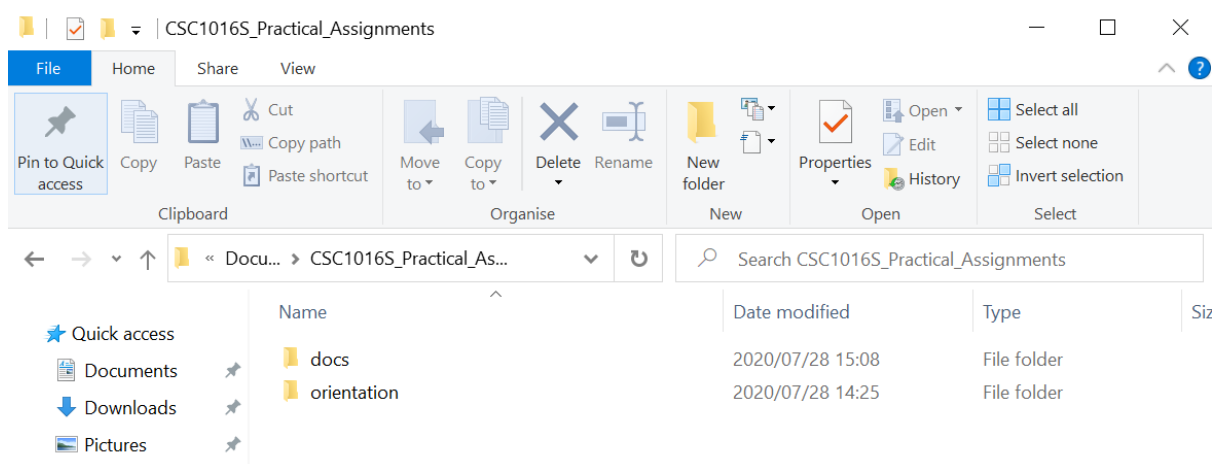
You will be presented with a window showing the contents of the ZIP. The next step is to extract the contents to your *csc1016s\_practical\_assignments* folder.



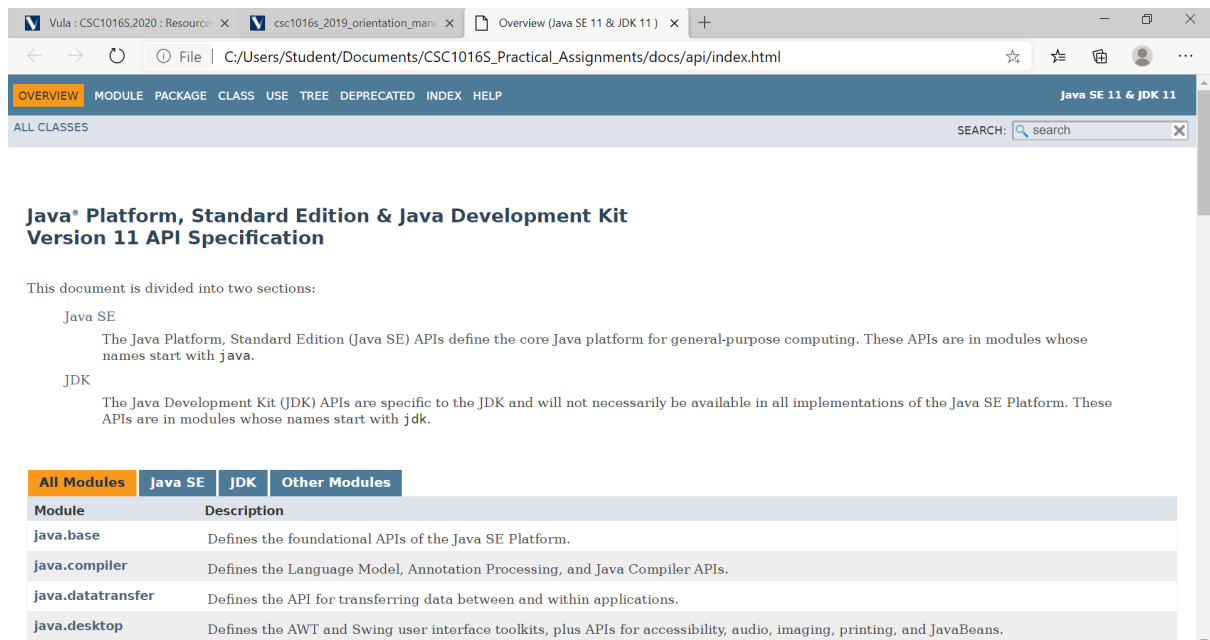
Click on the 'Extract' tab and then click on 'Extract all'. We don't want to extract the documents to the Downloads folder, so click 'Browse' and navigate to *csc1016s\_practical\_assignments*.



A window will open showing the extracted 'docs'.



If you open the *docs* folder, you will see four sub folders and a file called ‘index.html’. Click on *index.html* to view the documentation in your browser.



**Java Platform, Standard Edition & Java Development Kit Version 11 API Specification**

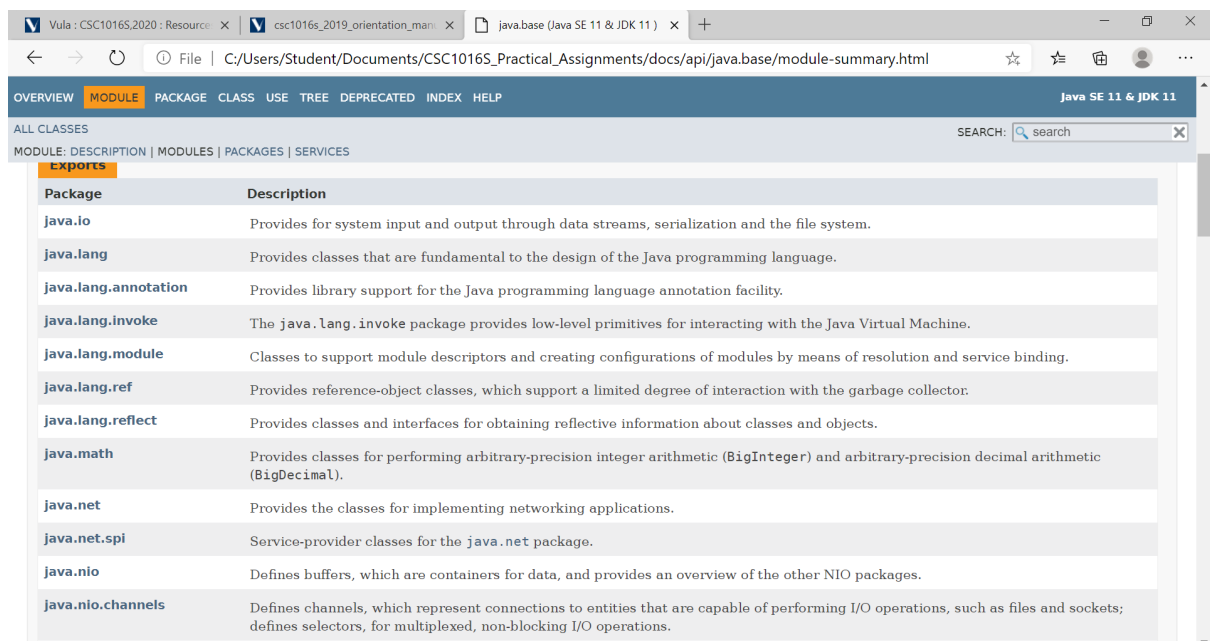
This document is divided into two sections:

**Java SE**  
The Java Platform, Standard Edition (Java SE) APIs define the core Java platform for general-purpose computing. These APIs are in modules whose names start with `java`.

**JDK**  
The Java Development Kit (JDK) APIs are specific to the JDK and will not necessarily be available in all implementations of the Java SE Platform. These APIs are in modules whose names start with `jdk`.

Module	Description
<code>java.base</code>	Defines the foundational APIs of the Java SE Platform.
<code>java.compiler</code>	Defines the Language Model, Annotation Processing, and Java Compiler APIs.
<code>java.datatransfer</code>	Defines the API for transferring data between and within applications.
<code>java.desktop</code>	Defines the AWT and Swing user interface toolkits, plus APIs for accessibility, audio, imaging, printing, and JavaBeans.

As mentioned, the Java API (component library) is extensive, however, you will only be using components in the Java ‘base’ package on this course. Clicking ‘**java.base**’ takes you there.

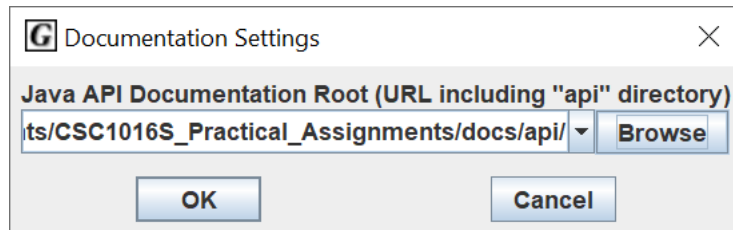


**java.base (Java SE 11 & JDK 11)**

Exports

Package	Description
<code>java.io</code>	Provides for system input and output through data streams, serialization and the file system.
<code>java.lang</code>	Provides classes that are fundamental to the design of the Java programming language.
<code>java.lang.annotation</code>	Provides library support for the Java programming language annotation facility.
<code>java.lang.invoke</code>	The <code>java.lang.invoke</code> package provides low-level primitives for interacting with the Java Virtual Machine.
<code>java.lang.module</code>	Classes to support module descriptors and creating configurations of modules by means of resolution and service binding.
<code>java.lang.ref</code>	Provides reference-object classes, which support a limited degree of interaction with the garbage collector.
<code>java.lang.reflect</code>	Provides classes and interfaces for obtaining reflective information about classes and objects.
<code>java.math</code>	Provides classes for performing arbitrary-precision integer arithmetic ( <code>BigInteger</code> ) and arbitrary-precision decimal arithmetic ( <code>BigDecimal</code> ).
<code>java.net</code>	Provides the classes for implementing networking applications.
<code>java.net.spi</code>	Service-provider classes for the <code>java.net</code> package.
<code>java.nio</code>	Defines buffers, which are containers for data, and provides an overview of the other NIO packages.
<code>java.nio.channels</code>	Defines channels, which represent connections to entities that are capable of performing I/O operations, such as files and sockets; defines selectors, for multiplexed, non-blocking I/O operations.

We won’t say any more about the documentation other than to note that you can link to it so that you can access it from within *JGrasp*. To do this, select the *JGrasp* ‘Settings > Documentation’ menu item and choose the location of the documentation to be the local copy you’ve just created.



Note that you actually need to set it to the 'csc1016s\_practical\_assignments/docs/api' directory i.e. to the 'api' directory inside the 'docs' directory.

Alternatively, the Java Documentation can be accessed online from the following website:

<http://docs.cs.uct.ac.za>

This will be helpful in later courses e.g. CSC2004Z in your second year. Navigate through the site and follow the links under "Java SE Development Kit API" to access the full documentation.

## Programming Documentation

### C++

- [CPP Reference](#)
- [C++ Tutorial \(PDF\)](#)

### Python

- Version 3: [3.4.1](#)
- Version 2: [2.7.8](#)

### Java

#### Java SE Development Kit API

- Version 8: [8u11 \[API\]](#)
- Version 7: [7u67 \[API\]](#)
- Version 6: [6u25 \[API\]](#)

Last updated: 1 Oct 2014

## Command-line Compiling

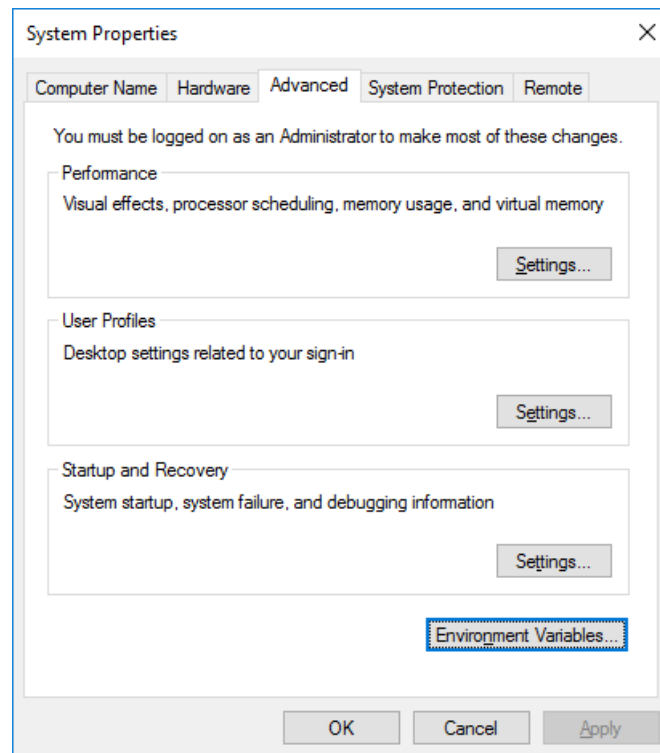
Instead of using JGrasp, you can use a separate editor and compiler. The Java Development Kit has a compiler – *javac* – that you can invoke from a command prompt (the command prompt is a text-

based application to navigate through the folders and tell Windows which application to execute without using the Start menu). You can execute Java programs using the *java* command.

Before you can do this, however, you need to check two settings on your computer.

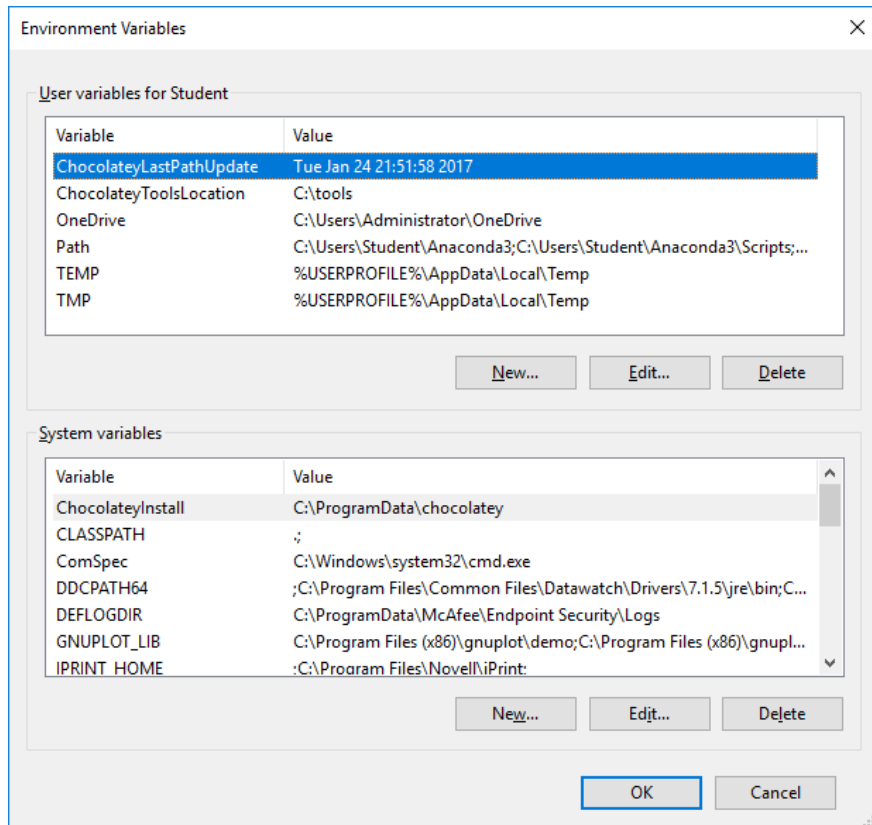
Firstly, we need to make sure that Windows can find the *java* and *javac* applications. Windows maintains a list of folders where it searches for applications in the **PATH** environment variable – we have to make sure that this list contains the folder where the Java applications are stored.

On your desktop, right-click *My Computer* and select *Properties*.



Click on *Advanced* then *Environment Variables*. Under *System variables*, click on *Path* then *Edit*.





Make sure that the *Variable value* starts with the text corresponding to the JDK location:

C:\Program Files\java\jdk-11.0.8\bin;

(If you followed the Java installation instructions at the start of the guide then this is its location.)

Secondly, we need to make sure that Java can in fact find the programs we want to execute. Java maintains a list of all places where it looks for compiled files in the system variable **CLASSPATH**. We need to tell Java that it should first look in the current folder.

Under *System variables*, click on *Classpath* then *Edit*.

Make sure that the *Variable value* starts with the text

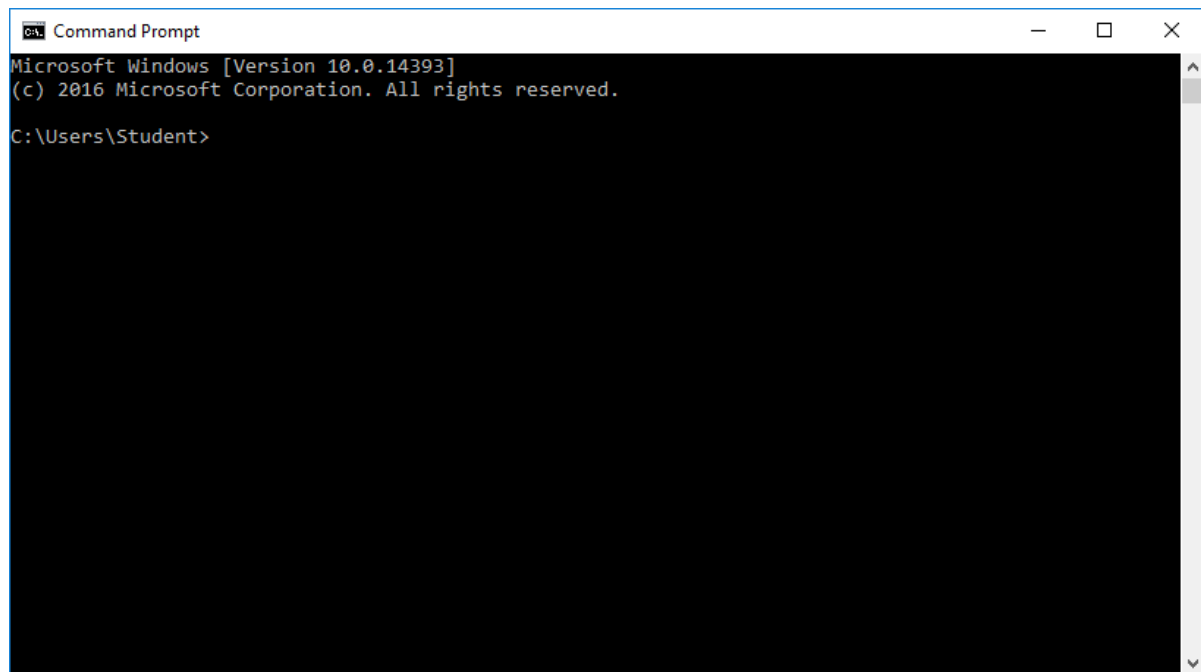
.;

(a single decimal point and a semicolon)

The single decimal point refers to the current folder. The semicolon separates multiple entries in the list. If there is no *Classpath* listed, click on *New* under *System variables* and enter the *Variable name* as *CLASSPATH* and the *Variable value* as indicated above.

Now you are ready to compile a program the hard way 😊

Run *Start / All Programs / Accessories / Command Prompt*. The version of Windows is displayed, followed by the current directory (folder).



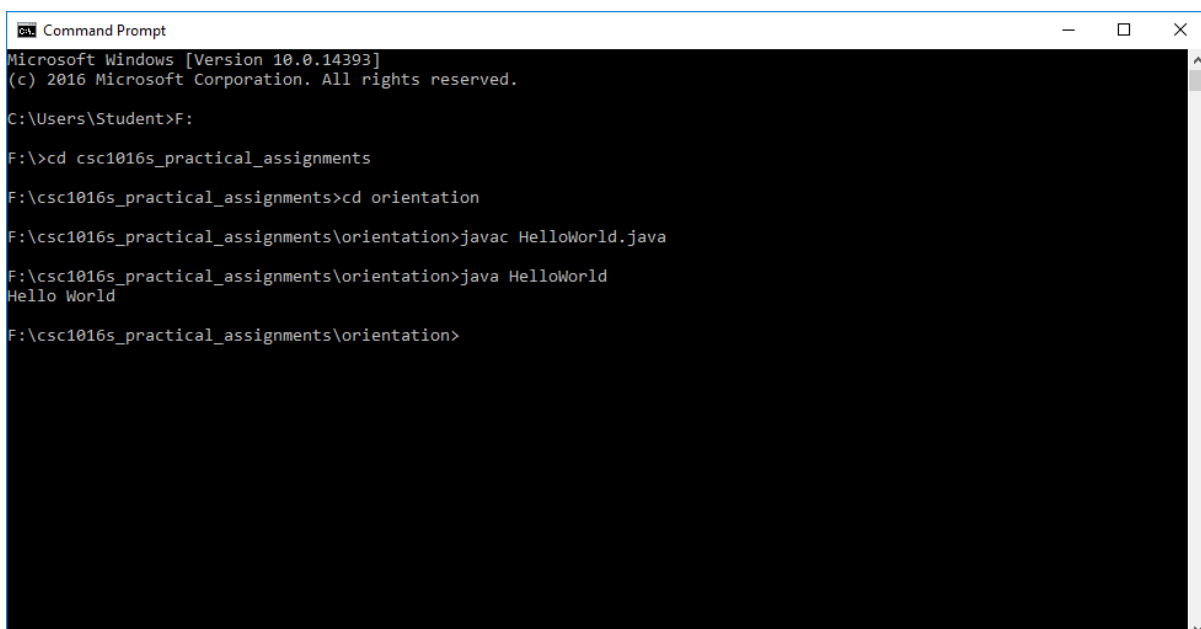
```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Student>
```

Using the following commands, navigate through the directory structure to where your program is stored:

cd Desktop	changes to the Desktop directory (analogous to opening the Desktop folder).
cd ..	changes to the directory containing the current one.
F:	switches to the F: drive.

Then compile your program using *javac* followed by the name of the class/file to compile.



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Student>F:
F:\>cd csc1016s_practical_assignments
F:\csc1016s_practical_assignments>cd orientation
F:\csc1016s_practical_assignments\orientation>javac HelloWorld.java
F:\csc1016s_practical_assignments\orientation>java HelloWorld
Hello World
F:\csc1016s_practical_assignments\orientation>
```

Finally, execute your Java application with the program *java*, followed by the name of the class which contains the *main method*, but do not specify the filename extension, i.e., only type *java HelloWorld*

To edit your program, you can use any text editor, such as notepad.

END

hussein suleman 2014-07-2

Stephan Jamieson 2020-07-31

Lebeko Poulo 2023-07-24