

# Computer Science 1015F/1016S

2008

## Orientation Manual

### Introduction

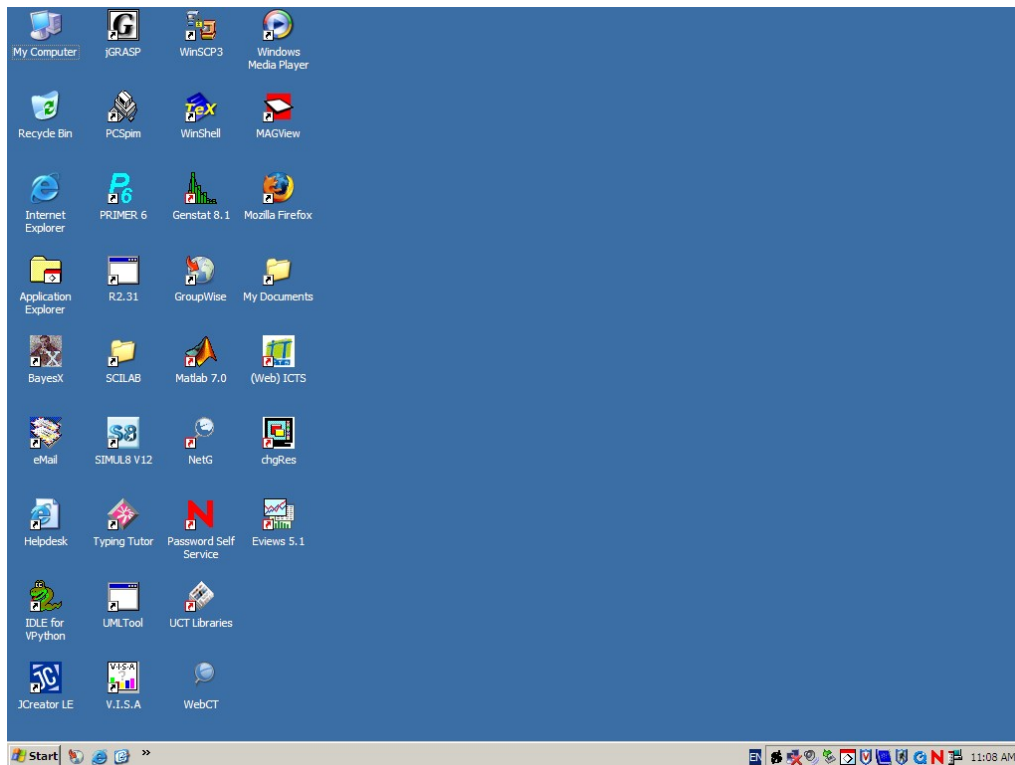
This is a self-paced tutorial on the procedure for using Scilab in order to complete CSC1015F practical assignments and submit them online using the *Vula* learning management system. As part of the process, you also will sign up for a weekly laboratory session if you have not already done so.

This procedure will be slightly different on computers in other labs or at home – if you need assistance with that, please do ask for help on the *Vula* forum or from your tutor.

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 or the person sitting next to you (assuming they are not lost as well ☺).

### Logging In

The very first step is to log into the Scilab computer with your student ID and password. After logging in you should be presented with a view of the *Windows XP* desktop that looks similar to the picture below.



*Windows XP* is the name of the operating system used in Scilab. 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.

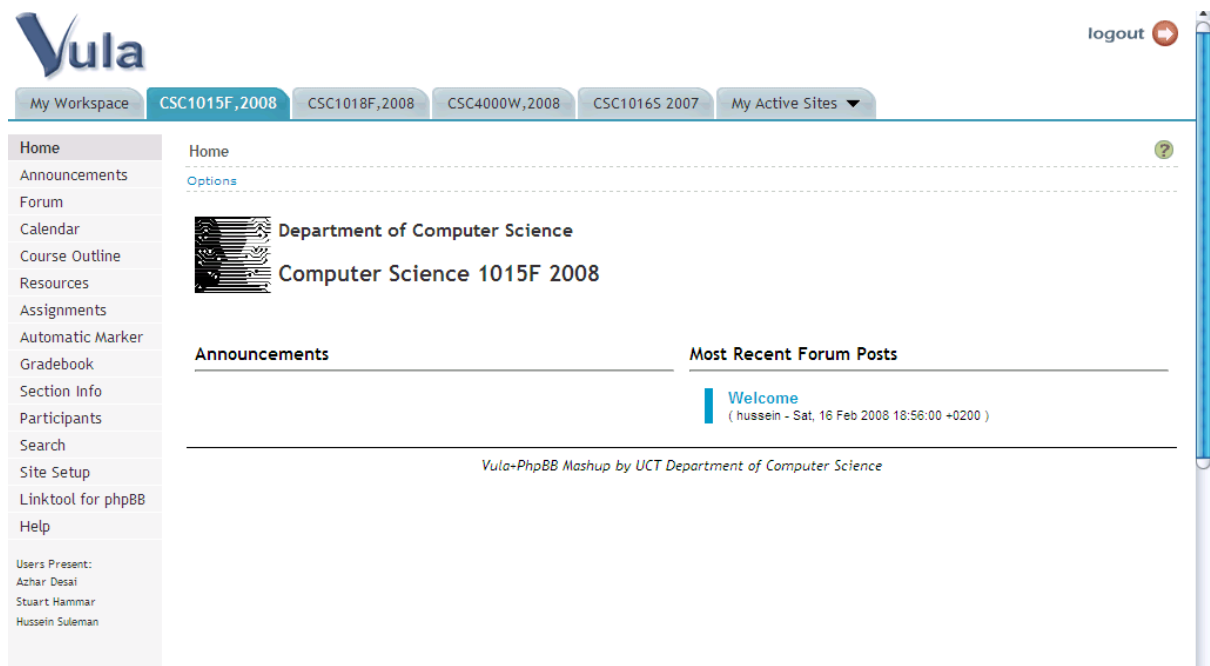
## Laboratory Group Signup

The first thing you will do is sign up for a laboratory group. If you are in the Science Faculty, you will be added to a laboratory group automatically. If you are in the Engineering faculty, a group may be recommended in your schedule of classes. In all cases, it is your responsibility to ensure that you are indeed in a group, and to join one if necessary.

Execute a Web browser application such as *Firefox* by double-clicking on the appropriate icon on the desktop. Enter a URL of <http://vula.uct.ac.za/> to go to the *Vula* website. If the Web browser asks you to log into **campusnet**, this is UCT's system to control access to the outside Internet – simply enter your userID and password as before and click Ok.



Enter your username and password and you will be able to log in. Then click on the *CSC1015F* tab at the top and you will be at the front page of the class website.



There are lots of tools available that you can explore later. For now we want to select a laboratory group. Click on *Section Info*. You will be presented with a list of all possible laboratory groups and the day of week and time at which they meet every week.

|  |  |
|--|--|
| Course Outline                                     | View <input type="button" value="All Sections"/> |
| Resources  |  |
| Assignments  |  |
| Automatic Marker                                   |  |
| Gradebook  |  |
| Section Info                                       |  |
| Participants                                       |  |
| Search   |  |
| Help   |  |
| Users Present:<br>Stuart Hammar<br>Tapuwa Mwashita |  |

| Section Name        | Tutor(s) | Day | Time            | Location          | Avail. |                      |
|---------------------|----------|-----|-----------------|-------------------|--------|----------------------|
| <b>Lab Sections</b> |          |     |                 |                   |        |                      |
| Laboratory Group M1 |          | M   | 2:00 pm-4:00 pm | Scilab A - Wilkes | 16     | <a href="#">Join</a> |
| Laboratory Group M2 |          | M   | 2:00 pm-4:00 pm | Scilab A - Wirth  | 16     | <a href="#">Join</a> |
| Laboratory Group M6 |          | M   | 4:00 pm-6:00 pm | Scilab A - Wirth  | 16     | <a href="#">Join</a> |
| Laboratory Group M7 |          | M   | 4:00 pm-6:00 pm | Scilab A - Wilkes | 16     | <a href="#">Join</a> |
| Laboratory Group T1 |          | T   | 2:00 pm-4:00 pm | Scilab A - Wilkes | 16     | <a href="#">Join</a> |
| Laboratory Group T2 |          | T   | 2:00 pm-4:00 pm | Scilab A - Wirth  | 16     | <a href="#">Join</a> |

Click on the word *Join* next to the laboratory group you want to join. If you do not see the word *Join* it means you are already allocated to a group.

Course Outline

Resources

Assignments

Automatic Marker

Gradebook

Section Info

Participants

Search

Help

Users Present:

Stuart Hammar

Tapuwa Mwashita

hstest.suleman

section for that section type.

View 

All Sections

| Section Name        | Tutor(s) | Day | Time            | Location          | Avail. |        |
|---------------------|----------|-----|-----------------|-------------------|--------|--------|
| Lab Sections        |          |     |                 |                   |        |        |
| Laboratory Group M1 |          | M   | 2:00 pm-4:00 pm | Scilab A - Wilkes | 15     | Member |
| Laboratory Group M2 |          | M   | 2:00 pm-4:00 pm | Scilab A - Wirth  | 16     | Switch |
| Laboratory Group M6 |          | M   | 4:00 pm-6:00 pm | Scilab A - Wirth  | 16     | Switch |
| Laboratory Group M7 |          | M   | 4:00 pm-6:00 pm | Scilab A - Wilkes | 16     | Switch |
| Laboratory Group T1 |          | T   | 2:00 pm-4:00 pm | Scilab A - Wilkes | 16     | Switch |
| Laboratory Group T2 |          | T   | 2:00 pm-4:00 pm | Scilab A - Wirth  | 16     | Switch |

After you have chosen a group, you may change your selection at a later date by using the Switch link, provided that there are still open seats. After everyone has been allocated to a session, changes will no longer be allowed.

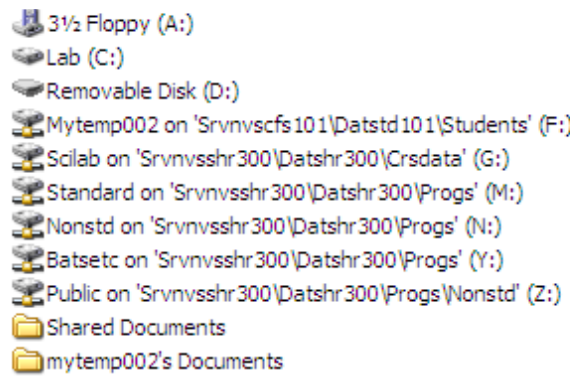
Now we do not need *Vula* for a while so you can click on *Log Out* and close the *Firefox* window by clicking on the X in the top right corner of the window.

Before we can try to write any **programs**, we need to understand some basic **file management** techniques.

## File and Folders

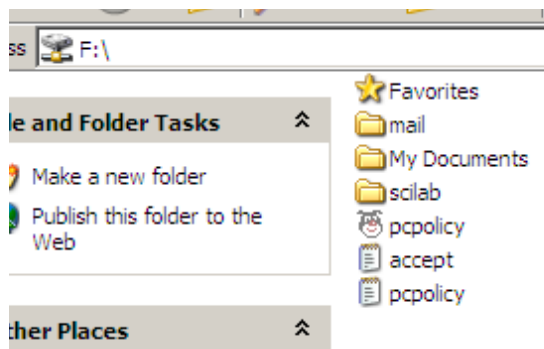
Any data on your computer is actually a collection of bits (0s and 1s) that are stored on some **storage device** (such as a **hard drive**) and given a name. In order to differentiate one piece of data from another, we refer to each piece of data as a **file** and give it a reasonably unique name, hereafter 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 *My Computer*, you will see a list of the storage devices and folders available to you in Scilab.



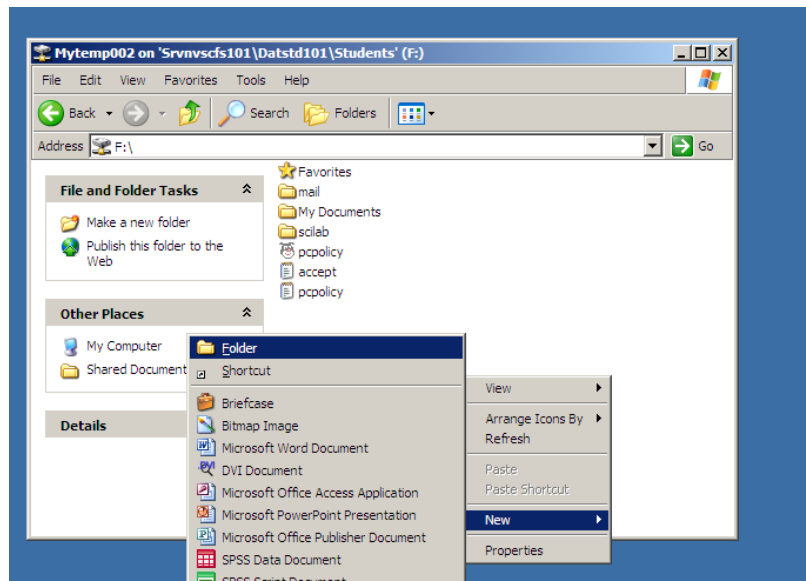
- A: is the floppy drive.
- C: is the hard drive that stores applications/programs on the computer.
- D: is a flash drive that was used for backups (it may not be present on your computer).
- F: is student storage space that will be available to you irrespective of which computer you log into (because it is actually on a remote computer and shared on the network). Each student can store up to 20 Megabytes of files on this device and this is where you should be storing your assignment in progress.
- G: has common files for all students in particular courses.

Double-click on F: and you will see the contents of this storage device. Currently it contains some links to websites (the star), some folders (with a yellow folder icon/picture) and some files (e.g., *accept*).

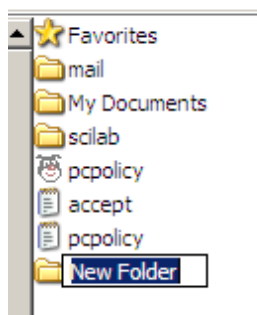


First we will create some new folders to organise the programs we write during the course and make sure that everything can be easily found in future.

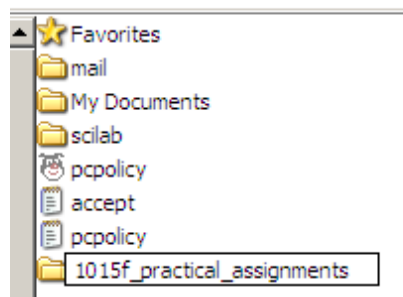
Right-click on the white part of the window just below the list of files and folders, select *New* and then *Folder*.



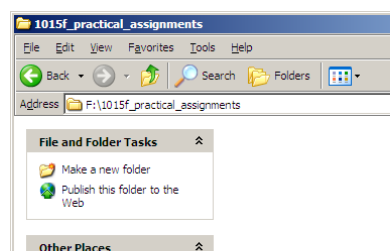
A new folder will be created and the text is selected so you can enter a new name (the default name is *New Folder*).



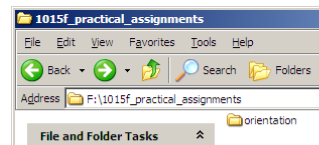
Type the name *csc1015f\_practical\_assignments* (or something else that you think is suitable) and press the *Enter* key. The folder will be renamed and the new name will be displayed.



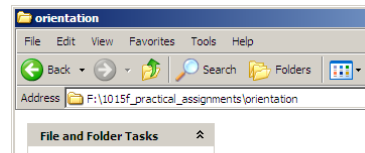
Now double-click on the new folder that you have created and you will be able to see the list of files it contains.



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.



## Java IDE

The ultimate 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**.

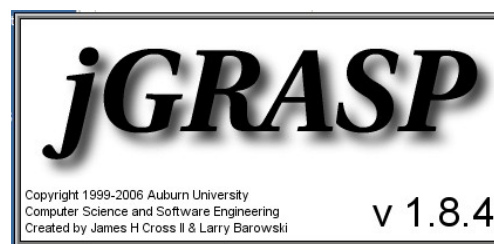
*Java* is the high-level programming language used in this course.

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**.

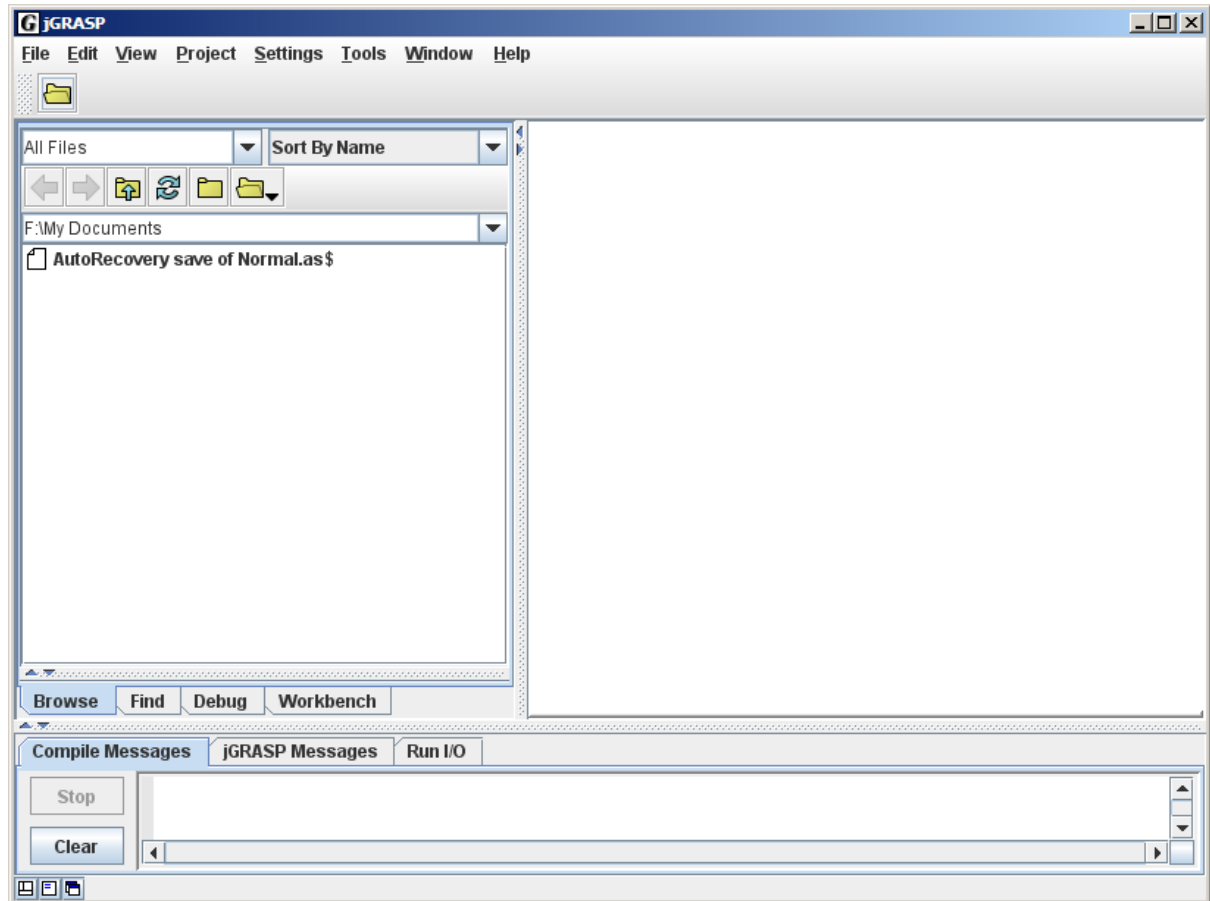
Since programming happens in stages, the programmer has to switch quite a lot between a text editor and a compiler – this can quickly get tiring. To make our lives much easier, the compiler and text editor are usually combined into a single application called an **Integrated Development Environment (IDE)**. 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 ☺.

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

Double-click on the *JGrasp* icon on the desktop to execute the application.



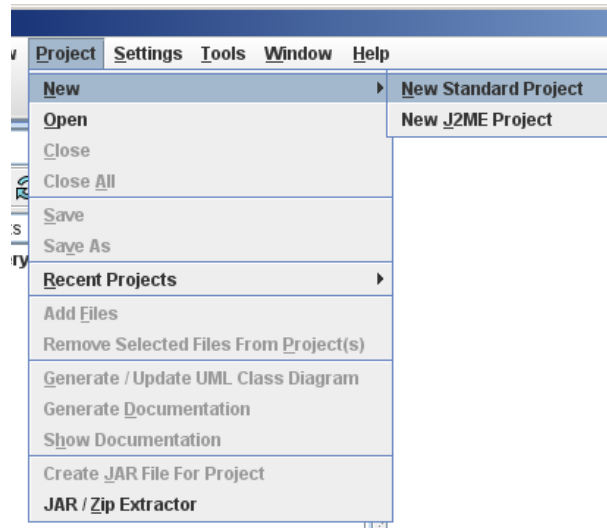
As *JGrasp* starts up, it temporarily displays a **splash-screen**, which provides some information about the **application** as it is loaded into the computer's memory. Eventually the splash-screen disappears and the main IDE window appears. A warning may be displayed about this version of Jgrasp not being tested on this platform – click Ok.



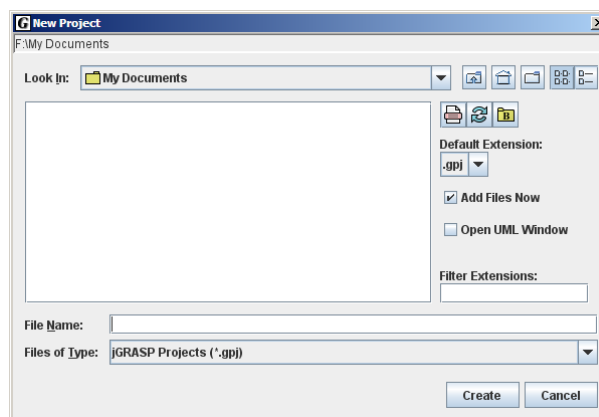
The *JGrasp* IDE has 4 **panes** (sections of a window) at startup. At the top are **buttons** and a **menu** to activate various functions of the IDE. The left pane displays the contents of a folder – we will not use this right now. The right pane is an empty **canvas** (blank window) where editing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

The first step is to specify that you are working on a new project. You will usually do this as the first step of the solution to each part of a programming assignment. Click on *Project* on the menu, then *New* and then *New Standard Project*.

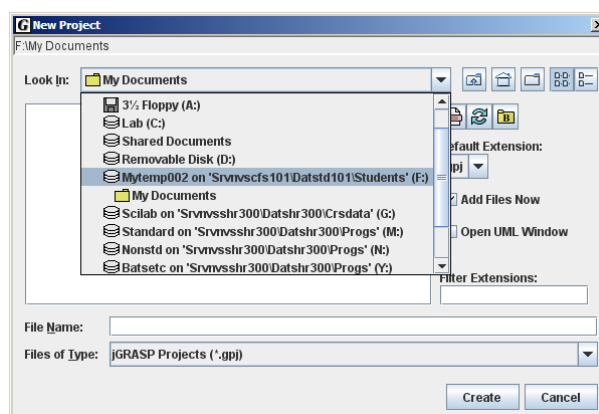




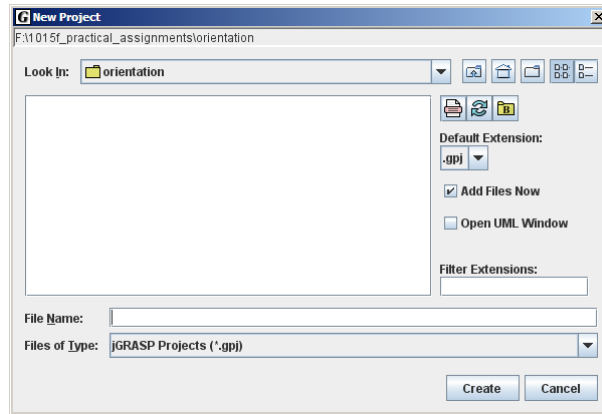
A window will be displayed where you can enter the name of a filename that will be used to store various bits of data about the project.



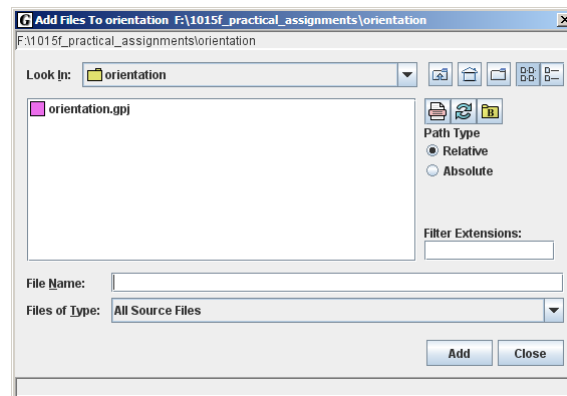
The first thing you want to do is make sure that you store the project in the folder you created earlier on. So, click on *My Documents*. A drop-down list will appear, which looks very similar to the folder you originally saw when you opened *My Computer*. This is not an accident – this is the same list!



Now, remember that you created a folder called *orientation* within a folder called *1015f\_practical\_assignments* on the F: drive. So, first select F:, then double-click on *1015f\_practical\_assignments* and finally click on *orientation*.



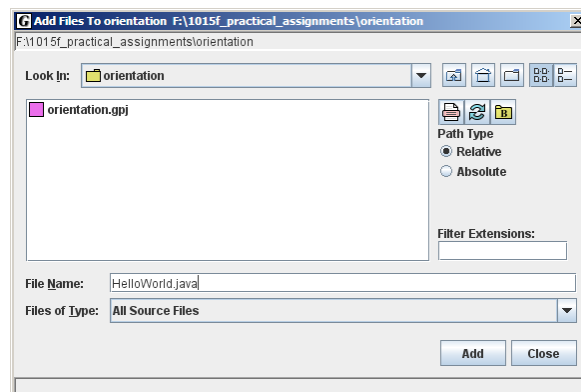
Now that you are in the right place on your storage device, enter *orientation* in the box next to *File Name*. To confirm the name, click on *Create* or press *Enter*. You will see a new window that looks very similar to the previous one.



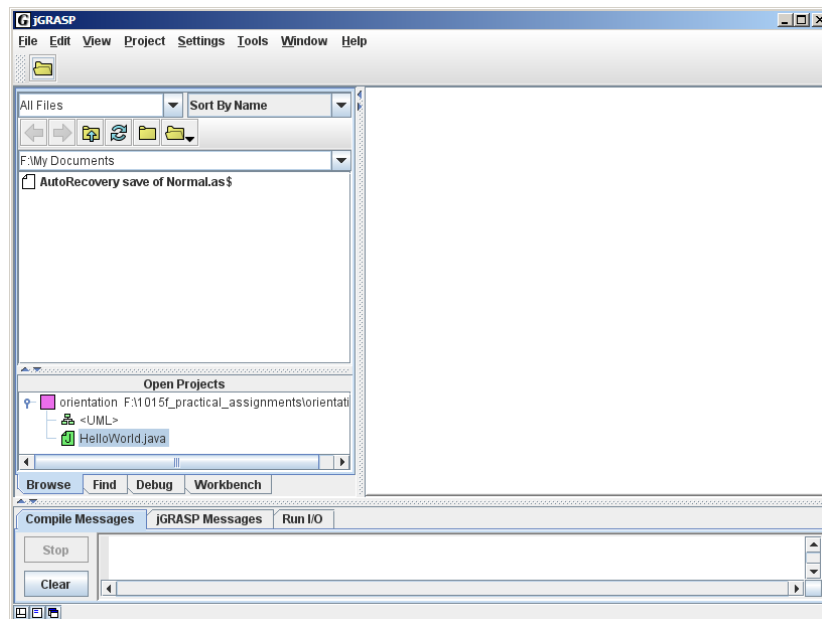
*JGrasp* is now asking you for a list of files that will store the Java program.

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. Now *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.

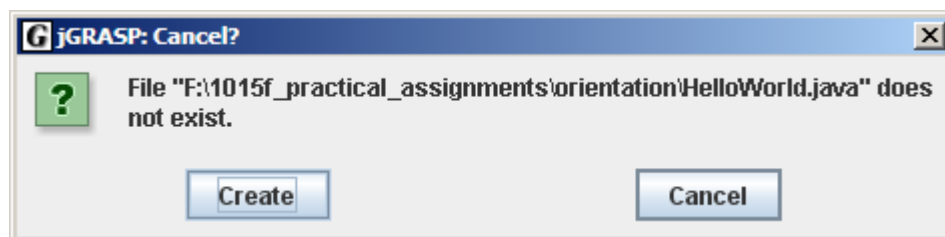
For this program there will be only one file. Enter `HelloWorld.java` as the filename.



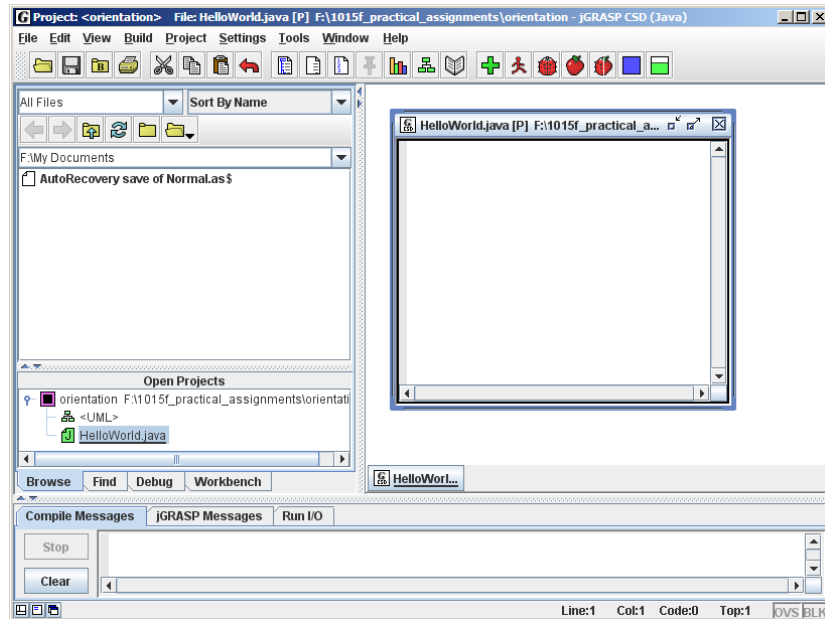
Press *Enter* if you have not already done so.



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 detail in second year computer science – now we are more interested in writing a simple program. So, double-click on *HelloWorld.java* in the Project view.

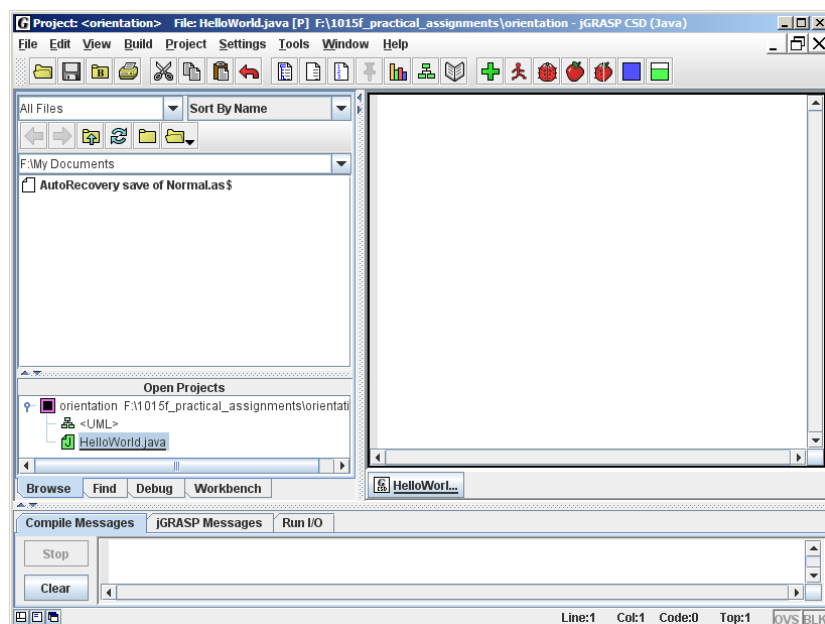


*JGrasp* lets you know that the file does not yet exist and asks if it should create the file. Click on *Create*.



A new window has appeared on the canvas where you can edit the contents of the *HelloWorld.java* file that will contain the Java program. Note that the **title bar** of the window indicates what the name of the file is.

Before continuing, we can expand this window by double-clicking in the middle of the title bar.



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:

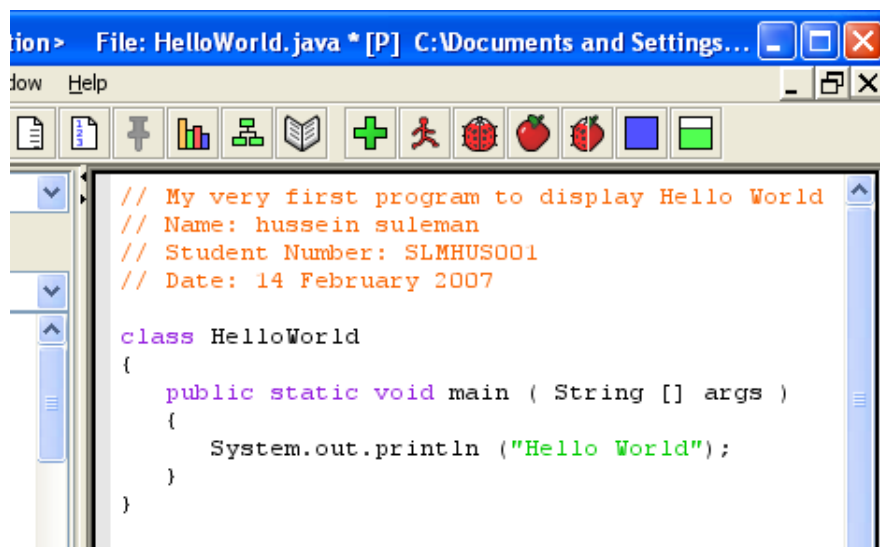
```
// My very first program to display Hello World on screen
```

```
// 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 make 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.

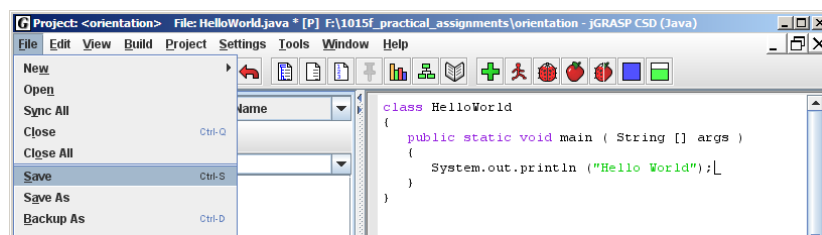


As you typed the program in, *JGrasp* highlighted the words in different colours. This **syntax highlighting** helps programmers to understand their code more easily. The red lines are comments that are not instructions to the computer but help any readers of the program understand what it does, who 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.

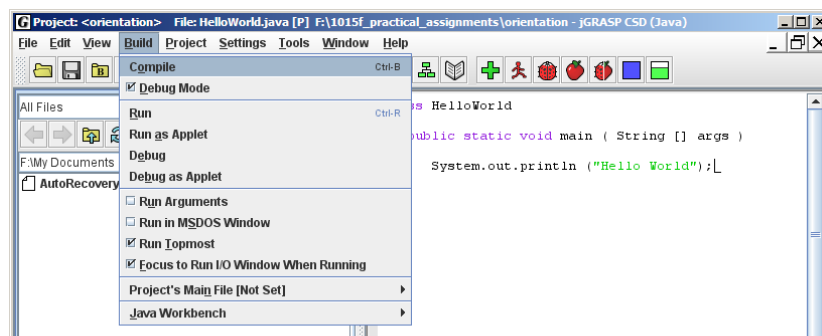
Now that we have entered the program, the **FIRST** thing we **ALWAYS** do is **save** the file. This transfers the program from the computer's memory to the computer's hard drive, where it is stored permanently. Without saving, if there is a power failure (for which Cape Town is infamous!), any changes 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*.

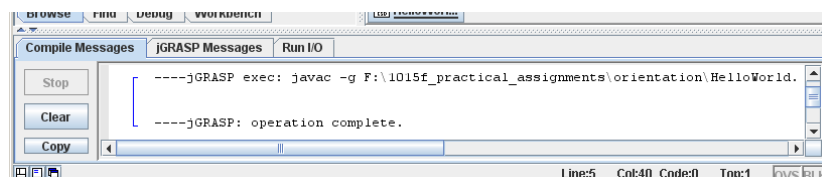


*JGrasp* saves the file – it does not ask you where to save the file because it already knows the filename (which is *HelloWorld.java* in this case).

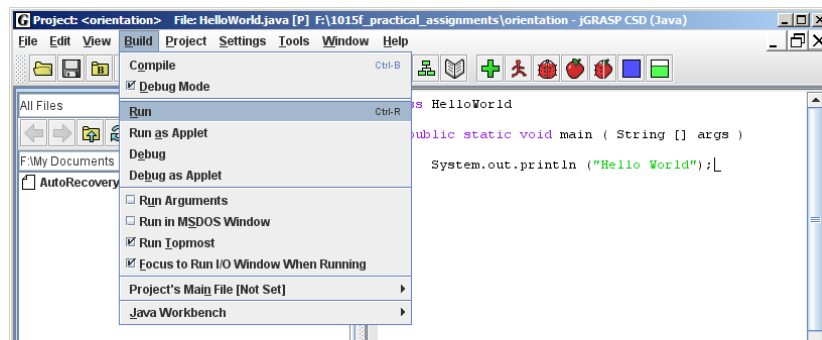
We can now compile the file (translate it from high-level source code to low-level machine language). Click on *Build* on the menu, then *Compile*.



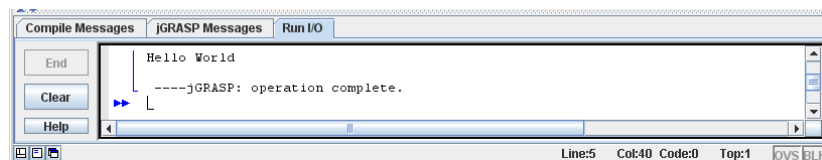
*JGrasp* displays a message in the message pane to indicate that it has executed the compiler on the source file and since there are no other messages between *exec* and *operation complete* it means that the compiler could understand the program perfectly well. If there are errors, it means that the 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



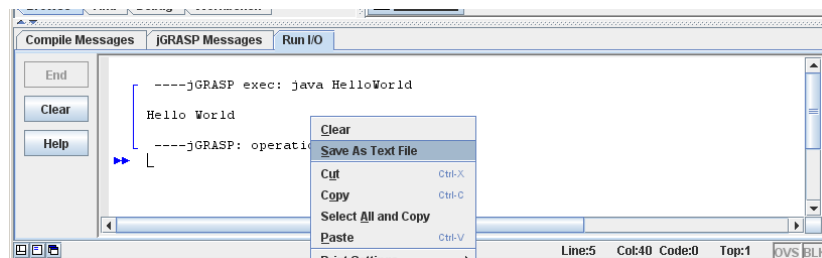
In the message pane, *JGrasp* has now switched to *Run I/O* mode and displays the results of executing the program. You should see the words *Hello World*.



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

Now, if you had 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 1015F assignments, but some courses may require that you submit output as well as the program you wrote.

Close *JGrasp*. We are done with it.

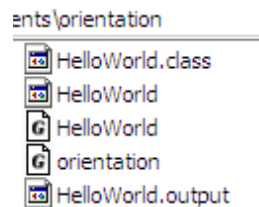
## Creating a Zip File

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 *Vula*.

If you have the *orientation* folder still open, you should see some other files in it.

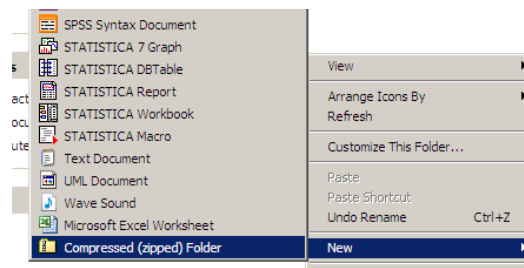
*HelloWorld.class* is the machine code version of the file after compilation. *orientation.gpj* contains information on the project.

(*HelloWorld.output* is the output file if you created this. If you open this file in *WordPad* or *NotePad* you will see the output from your program.)

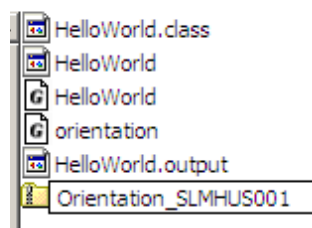


To submit an assignment, you can only send in one file. However you need to send in at least all the source code files. To make this possible we first create a **Zip file** that can contain other files. This is almost like a folder, and in fact is sometimes referred to as a compressed folder – compressed 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 empty space in the *orientation* folder, click *New* and then *Compressed (zipped) folder*.

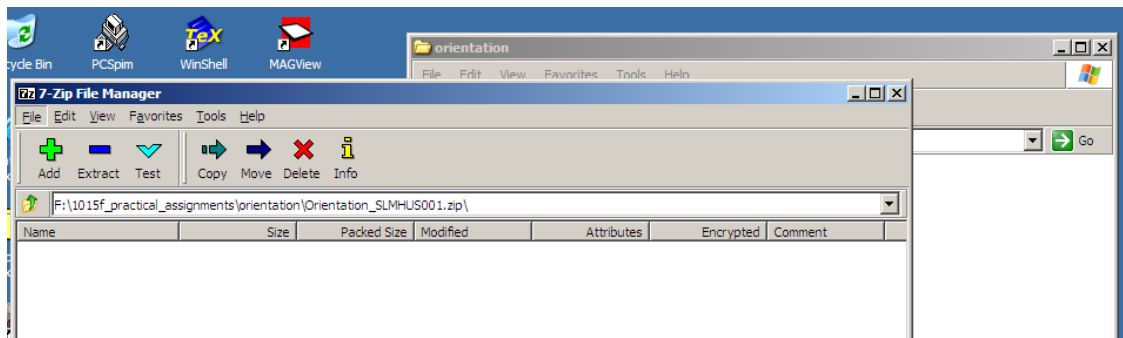


Name the file *Orientation\_ABCXYZ123*, where *ABCXYZ123* should in fact be your student number.



The Zip file has an obvious icon that includes a little zip down the left hand side. Double-click on the Zip file and you will open the Zip **archive** to display what is inside.



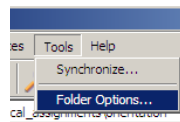


There should be nothing – just like an empty folder.

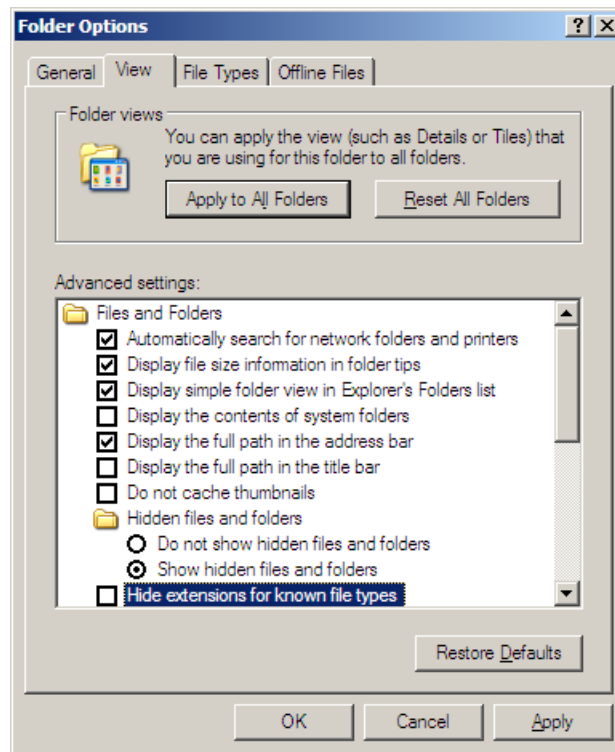
Click on the partially-hidden *orientation* folder as we need to put the files we want to submit into the Zip archive.

But there may be a small problem. In *JGrasp* we had created a file called *HelloWorld.java* but the folder may display something called *HelloWorld*, or *orientation* instead of *orientation.gpj*. 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 XP 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.

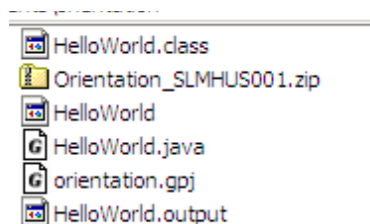
Luckily we can tell Windows XP not to hide the extensions. Click on the *orientation* window. Then click on the *Tools* menu and choose *Folder Options*.



A window pops up with various options related to the folder. Click on *View*.



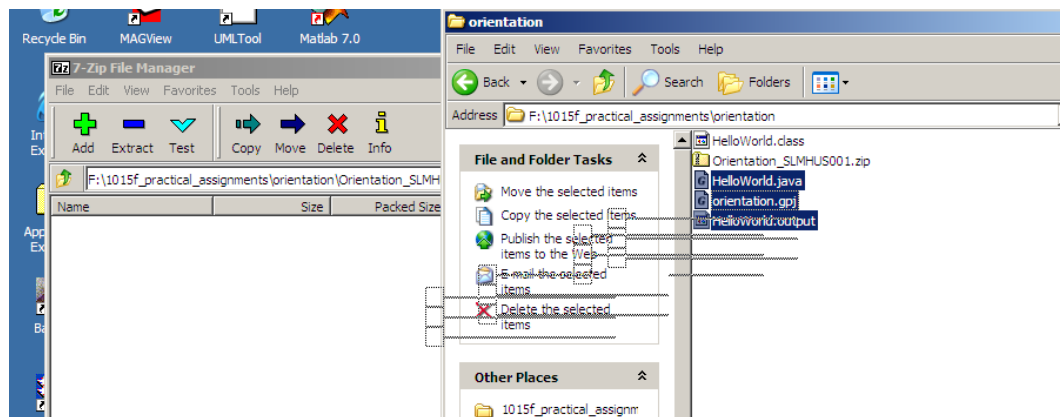
The option right at the bottom is the one we want to make sure is not **checked**, i.e., there is no tick next to it – if there is still a tick, click on the little box to remove the tick. Then click on *Ok*.



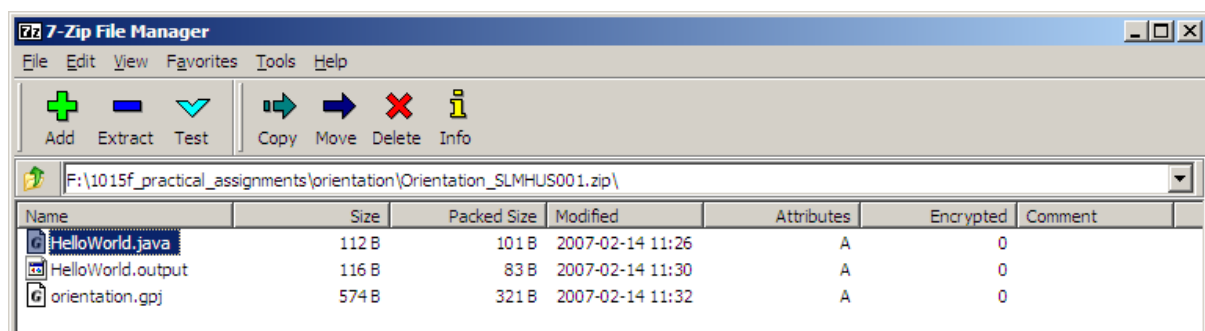
The folder now displays the complete filenames for all files. You want to copy *HelloWorld.java* (maybe *HelloWorld.output*) and *orientation.gpj* to the Zip archive. We do not submit *.class* files with assignments as these can easily be recreated by opening the project in *JGrasp* and **building** the project.

Select each 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.



Before your files can be added to the Zip archive, you will be asked to confirm that this is in fact what you want to do. Do say *Yes*.



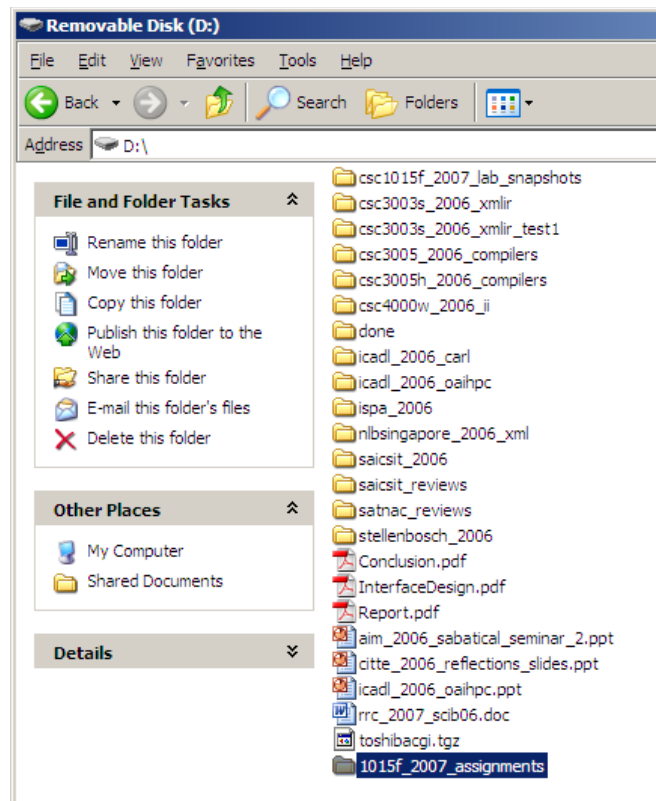
The Zip archive will finally display two or three files, the ones that we dragged over. We can test that these files can be **extracted** or **viewed** if they are text files using the various options available in the Zip file manager – this is left as an exercise.

This Zip file is the file that will be submitted. 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, even if it is just a floppy disk for now!

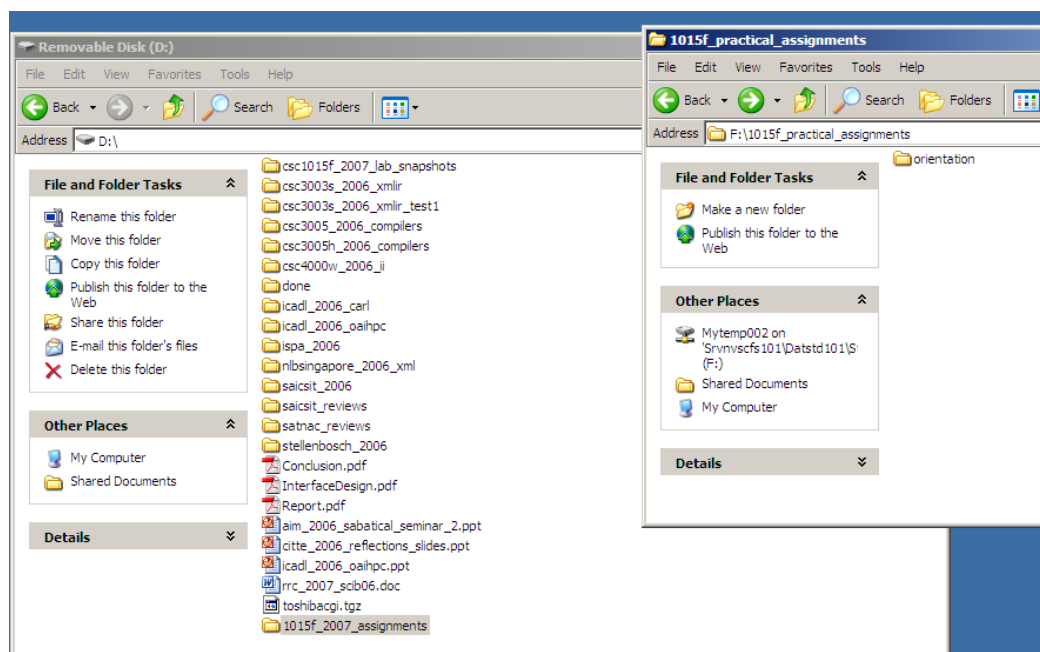
## Backups

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.

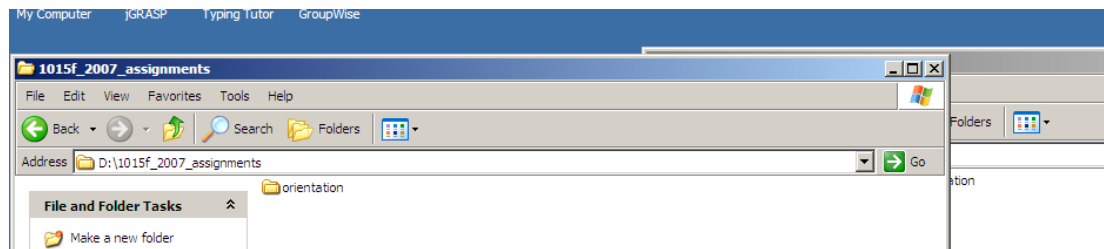
Backups are usually stored on removable storage devices such as floppy disks, USB flash drives and removable USB hard drives. When you connect a removable drive to the computer, you can usually immediately see it in the list of devices listed on *My Computer*. Double-clicking on such a device will bring 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 F: drive 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. Windows XP 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!

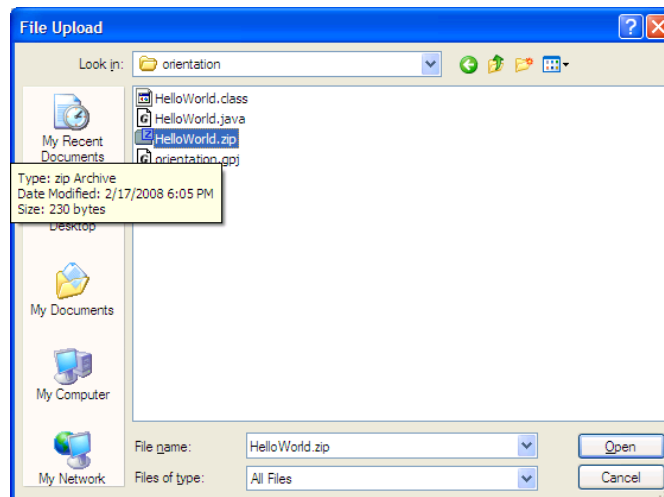
Run *Firefox* like before and go to the *Vula* website. Log in and switch to the class website.

|                  |               |                     |         |   |
|------------------|---------------|---------------------|---------|---|
| Assignments      |               |                     |         |   |
| Automatic Marker | Assignment 1  | 2008-03-06T17:00:00 | 0 / 10  |   |
| Gradebook        | Orientation   | 2008-03-10T17:00:00 | 0 / 100 | <input type="text"/> <input type="button" value="Browse..."/> <input type="button" value="Submit"/> |
| Section Info     |               |                     |         |   |
| Participants     |               |                     |         |   |
| Search           |               |                     |         |   |
| Help             |               |                     |         |   |
| Users Present:   | Assignment 2  | 2008-03-13T17:00:00 | 0 / 10  |   |
| Shaun Crossman   | Assignment 3  | 2008-03-20T17:00:00 | 0 / 10  |   |
| Flora Kundaali   | Assignment 4  | 2008-04-03T17:00:00 | 0 / 10  |   |
| Sarah Palser     | Assignment 5  | 2008-04-10T17:00:00 | 0 / 10  |   |
| Schalk Potgieter | Assignment 6  | 2008-04-17T17:00:00 | 0 / 10  |   |
| htest suleman    | Assignment 7  | 2008-04-24T17:00:00 | 0 / 10  |   |
|                  | Assignment 8  | 2008-05-02T17:00:00 | 0 / 10  |   |
|                  | Assignment 9  | 2008-05-08T17:00:00 | 0 / 10  |   |
|                  | Assignment 10 | 2008-05-15T17:00:00 | 0 / 10  |   |

Click on *Automatic Marker*.

A list of all assignments is displayed and you can click on the one that you wish to submit. 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 are allowed to 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.



Then, click on *Submit* and your assignment will be submitted and marked automatically. The system unzips your file, **compiles** it and then **executes** a battery of tests against it. In this case, there is only one question and only one trial that is done.

logout

My WorkspaceCSC1015F, 2008

Home

Announcements

Forum

Calendar

Course Outline

Resources

Assignments

Automatic Marker

Gradebook

Section Info

Participants

Search

Help

Users Present:  
Shaun Crossman  
Flora Kundaeli  
Sarah Palser  
Schalk Potgieter  
htest suleman

 Automatic Marker

**Marking assignment ...**

Automarking ...


Unzipping file  
Archive: /home/automark/submissions/slumou@gmail.com/Orientation/1/HelloWorld.zip  
  inflating: HelloWorld.java  
  Compiling Question 1

Trial 1:  
Comparing output  
Output correct  
Score for question: 100  
Total marks for assignment: 100

**List of Submissions**

| Assignment      | Due Date            | Attempts | Submissions  |
|-----------------|---------------------|----------|--|
| Assignment Test | 2008-02-05T17:00:00 | 3 / 100  | assignment1i.zip[68] assignment1.zip[100] assignment1i.zip[48] |

At the bottom of the screen is the table of assignments and now it indicates your submission and the mark you got. The submit button is still there because you have not exhausted your allowed submissions and the assignment deadline has not passed. If you got less than 100, and you have time to try again and resubmit, you should do so!


logout

My Workspace

CSC1015F, 2008

[Home](#)  
[Announcements](#)  
[Forum](#)  
[Calendar](#)  
[Course Outline](#)  
[Resources](#)  
[Assignments](#)  
[Automatic Marker](#)  
[Gradebook](#)  
[Section Info](#)  
[Participants](#)  
[Search](#)  
[Help](#)  
  
Users Present:  
Shaun Crossman  
Flora Kundaali  
Sarah Palser  
Schalk Potgieter

Automatic Marker

---

### List of Submissions

| Assignment      | Due Date            | Attempts | Submissions  |
|-----------------|---------------------|----------|--|
| Assignment Test | 2008-02-05T17:00:00 | 3 / 100  | assignment1.i.zip[68] assignment1.zip[100] assignment1.i.zip[48]   |
| Assignment 0    | 2008-02-28T17:00:00 | 0 / 10   | <input style="width: 100px;" type="text"/> <input type="button" value="Browse..."/> <input type="button" value="Submit"/>                            |
| Assignment 1    | 2008-03-06T17:00:00 | 0 / 10   |  |
| Orientation     | 2008-03-10T17:00:00 | 1 / 100  | HelloWorld.zip[100]<br><br><input style="width: 100px;" type="text"/> <input type="button" value="Browse..."/> <input type="button" value="Submit"/> |

You can also check that Vula 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.

My Workspace

CSC1015F, 2008

[Home](#)  
[Announcements](#)  
[Forum](#)  
[Calendar](#)  
[Course Outline](#)  
[Resources](#)  
[Assignments](#)  
[Automatic Marker](#)  
[Gradebook](#)  
[Section Info](#)  
[Participants](#)  
[Search](#)  
[Help](#)  
  
Users Present:  
Shaun Crossman  
Flora Kundaali  
Sarah Palser  
Schalk Potgieter  
htest suleman

Assignments

---

### Assignment

Title      Orientation

Student    htest suleman

Grade:     100.0 (max 100.0)

Instructions

## Orientation Assignment

Write a program to print out the words "Hello World" to the screen.

Name your file "HelloWorld.java" and submit only a Zip file containing the source code.

---

Additional instructor's comments about your submission

Marked automatically by AutoMark

Vula's Assignment section can also be used to submit assignments that are not automatically marked – this is used in other courses and may be used later in CSC1015F or CSC1016S.

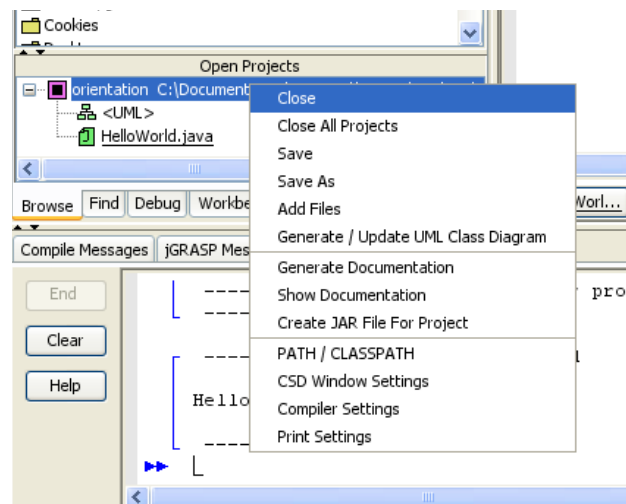
That's it folks! This is the end of the orientation exercise.

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.

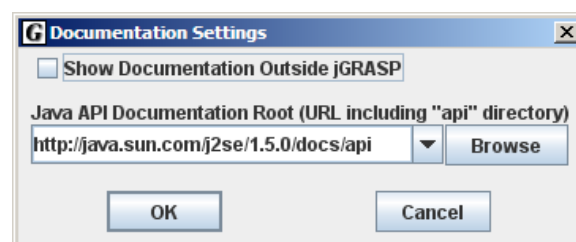
You can also add additional files to the project if your project has more than one class (most projects have more than one class). Right-click on the name of the project in the *Open Projects* pane and select *Add Files*.



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

You can link to the Java documentation so that you can access it from within *JGrasp*. To do this, select the *Settings / Documentation* menu item and choose the location of the documentation to be the local copy on your G: drive. This should make it much easier to access the Java API documentation while you are programming.





## 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 applications 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

C:\Program Files\java\jdk1.5.0\_05\bin;

(This is the location of Java on some computers – it could be different on yours!)

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).

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.

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*

```

C:\Documents and Settings\hussein>cd Desktop
C:\Documents and Settings\hussein\Desktop>cd Orientation_SLMHUS001
C:\Documents and Settings\hussein\Desktop\Orientation_SLMHUS001>javac HelloWorld.java
C:\Documents and Settings\hussein\Desktop\Orientation_SLMHUS001>java HelloWorld
Hello World
C:\Documents and Settings\hussein\Desktop\Orientation_SLMHUS001>

```

To edit your program, you can use any text editor – *edit HelloWorld.java* brings up the old-fashioned pre-Windows editor that many programmers use for quick changes.

```

File Edit Search View Options Help
C:\...\hussein\Desktop\Orientation_SLMHUS001\HelloWorld.java
// My very first program to display Hello World
// Name: hussein suleman
// Student Number: SLMHUS001
// Date: 14 February 2007

class HelloWorld
{
    public static void main ( String [] args )
    {
        System.out.println ("Hello World");
    }
}

```

F1=Help | Line:1 Col:1

Press ALT to get to the menu here. Finally, at the command-prompt, type *exit* to close the window.

the end!

hussein suleman 2008-02-18