Java - Getting Started

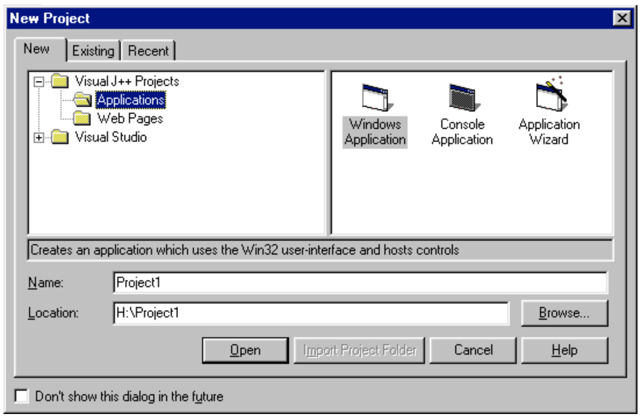
There are 3 sets of files needed to get the most out of Java

1. The JDK (Java Development Kit) this compiles and runs Java programs. This can be downloaded from www.java.sun.com. The JDK is all that is needed to write Java if you are happy with a Dos environment and coding in a text editor. Sun redo the JDK with dazzling frequency some of the changes are minor and others real showstoppers. Java 1.1 is still heavily used as this is the best that some browsers can cope with. From 1.1.6 Sun introduced Swing, a much improved visual interface. Java 1.2 is officially called Java 2 but the latest version is Java 1.3, which is not called Java 3. As Sun introduce new versions Java gets progressively more complex. As new versions support older JDKs there can be several ways to code the same effect. Older methods that are still supported but bound for the scrap heap are noted as legacy.
2. JavaDoc consists of all the help files for the JDK. This is also downloadable from Sun but is an even bigger file (30 Mb). The help files are not all that helpful. At best they will point out what has gone wrong if you already understand what is going on. Anyone used to the wonderful help and examples found with Visual Basic is in for a let down.
3. The IDE, a program that makes writing and compiling Java a lot easier. Calderdale College uses Microsoft Visual Java as it is fast and intuitive especially for those with a VB or VC+ background. There are more than a few problems. VJ only supports Java 1.1 so no swing. Microsoft also added various extensions to create controls by dragging and dropping and generating code. These only work in a Windows environment and are not pure Java. The Microsoft extensions will not be used on this course. To compare IDEs Forte for Java is also available in Calderdale College. This can be linked to any JDK, currently it is set to 1.3. Forte is written in Java so is not the fastest IDE around. It can create AWT controls by dragging and dropping but take care as it will create way too many user defined classes. Forte is available as a free download from http://www.sun.com/forte/ffj. There are other IDEs, JBuilder from Borland can sometimes be found on cover disks, it is very similar to Forte but somewhat faster.

Turning VJ on

This is a no-brainer but if you punch all the default buttons Microsoft adds all sorts of options and Windows that are not going to be used. To start Java we want a nice empty Class that will run in a Dos Window.

By default Microsoft want us to create a Windows thingy

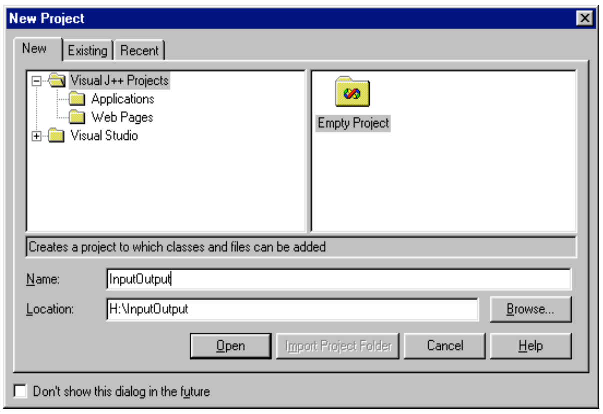


Creates an application which uses the Win32 user-interface and hosts controls

Name: Project1

Location: H:\Project1

Better by far is an empty project with a sensible name, note the file location

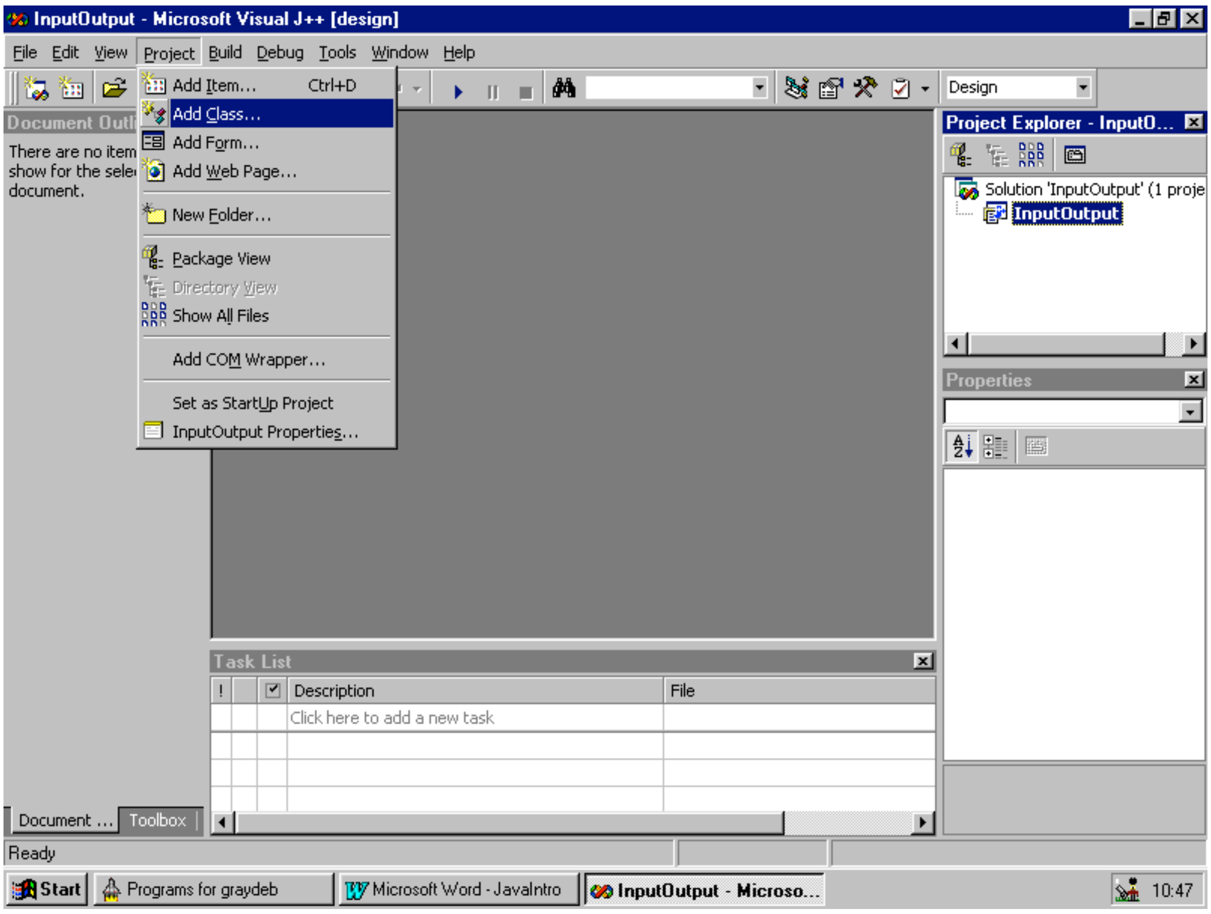


Creates a project to which classes and files can be added

Name: InputOutput

Location: H:\InputOutput

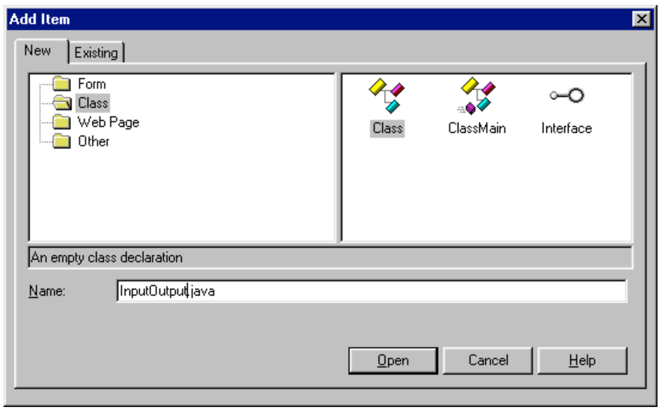
Java is made up of Classes, without at least 1 it is going nowhere, so add a class.



There are no items to show for the selected document.

Click here to add a new task

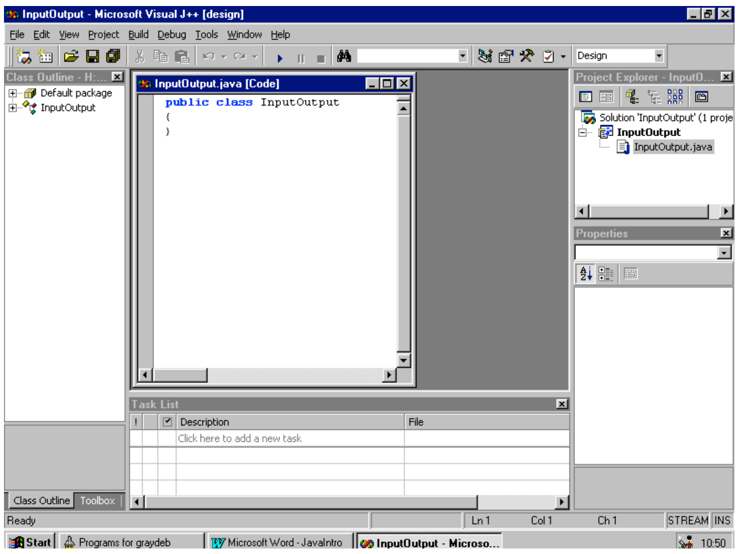
Ready



An empty class declaration

Name: InputOutput.java

Name the Class now. In Java a Class always starts with a capital letter. It will work with a little letter but you will be shunned by the Java great and good



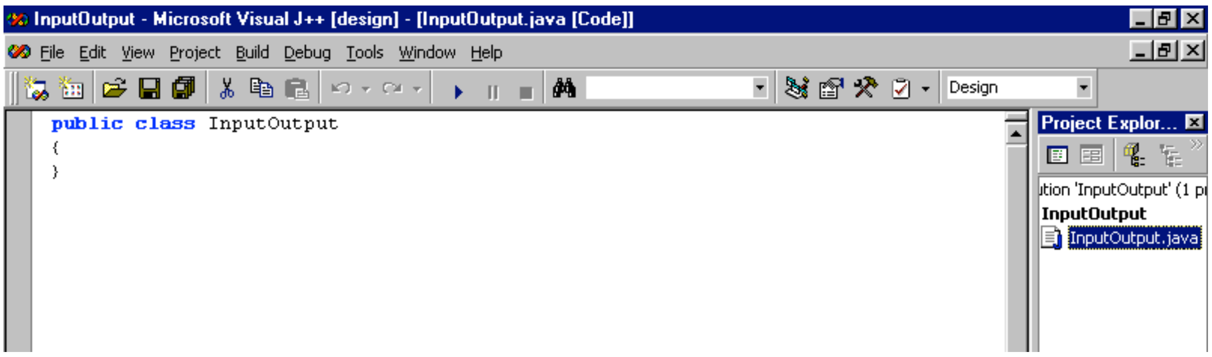
```
public class InputOutput
{
}
```

Click here to add a new task

Ready Ln 1 Col 1 Ch 1 STREAM INS

Voila.

Most of these Windows are of no earthly use so the IDE should be tidied up by closing a few.

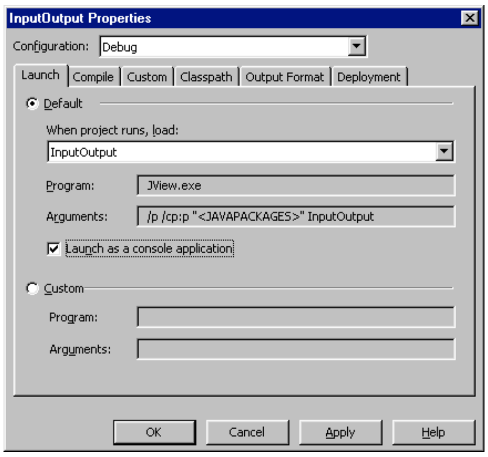

InputOutput - Microsoft Visual J++ [design] - [InputOutput.java [Code]]
File Edit View Project Build Debug Tools Window Help
Design
Project Explor...
ition 'InputOutput' (1 p
InputOutput
InputOutput.java

```
public class InputOutput
{
}
```

Writing some code

The IDE has already set up the Class outline. All the code will go within the braces of this Class. Java code is not dissimilar to C. Try something like this:

```
public static void main(String[] args)
{
        System.out.println("Compile boy compile");
}
```



Configuration: Debug

Default

When project runs, load: InputOutput

Program: JView.exe

Arguments: /p /cp:p "<JAVAPACKAGES>" InputOutput

☑ Launch as a console application

Custom

Program:

Arguments:

Then hit F5, anther Window appears. If there is more than 1 Class the IDE will not know which to run 1st so a drop down is provided. The important point here is to check the 'Launch as console application' box.

If you have typed the code right the output will appear with amazing speed and then go away again.

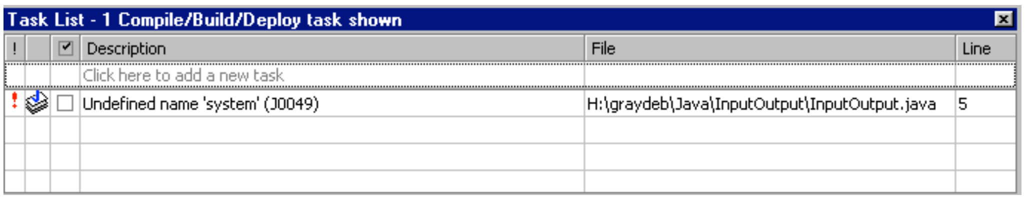
Problems

Naturally the above code worked 1st time but just for interest we will look at how errors are thrown and investigated. The Error Windows will become surprisingly familiar in time.

Type this code inside the Class:

```
public static void main(String[] args)
{
        system.out.println("Compile boy compile");
}
```

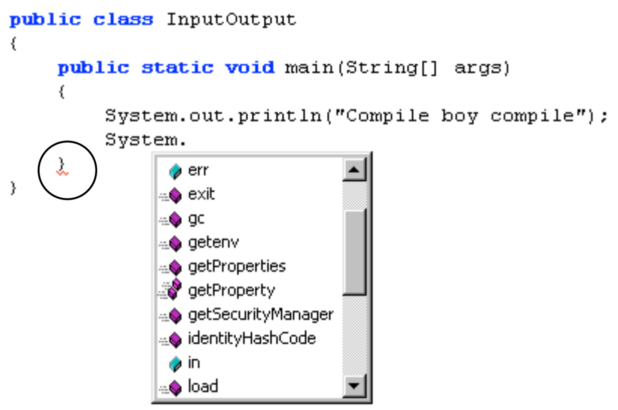
This may run but it also opens another Window


Task List - 1 Compile/Build/Deploy task shown
Click here to add a new task

!		✓	Description	File	Line
!		☐	Undefined name 'system' (J0049)	H:\graydeb\Java\InputOutput\InputOutput.java	5

The messages are typically opaque. Looking at the code the word 'system' in underlined in blue. This is more use as it pinpoints where the error is. The word system is actually a Class (and a static one too but not to worry about that) so it must be spelt System.

Type a new line of code but stop with the dot of System, a drop down appears:



```
public class InputOutput
{
    public static void main(String[] args)
    {
        System.out.println("Compile boy compile");
        System.
    }
}
```

These are all methods (functions) of the Class system. If it is not on the list System can't do it.

The dot is called the dot operator. If a drop down is expected and none appears something has gone wrong with the code, possibly on a previous line. It is best to stop coding and sort out the problem as Java is not going to be compiling. Note the squiggle under the last but 1 brace. This means an error has occurred, probably a syntax error. In this case 'System.' is not a valid Java line, no ; at the end and a dot without a method for a start.

Compete the output line to output any text.

Key points

1. Java classes need at least 1 method. For a console application this is:
    public static void main(String[] args) {}
It is similar to main(){} in C. The method may accept a String externally that is passed to the Class.

2. The Java Class file must have the same name as the Class it contains. If there is more than 1 Class in a file that name must be of the sole Public Class in that file.

3. Java code is very similar to C. The same rules of thumb apply.
- Most lines end in a ';'
- Braces divide up a program, if a brace starts it must finish.
- Variables must be declared; their scope is defined by the position of braces.

4. It pays to write a little code and then run it. It possible to clock up truly horrific numbers of errors in a short program.

5. Java is case sensitive; String refers to the class String but string is at best a variable (if you bothered to declare it. The following is legal but not recommended:

String string = "A Word";