Your First Program

Introduction to Programming

Getting started

1. Open the JGrasp IDE
2. Create a new Java file
3. Save the file as **`HelloWorld.java`**
4. Type the following program into the file:

The Program

```
public class HelloWorld
{
    public static void main(String [] args)
    {
       // Print the text to screen
       System.out.print("Hello World");
    }
}
```

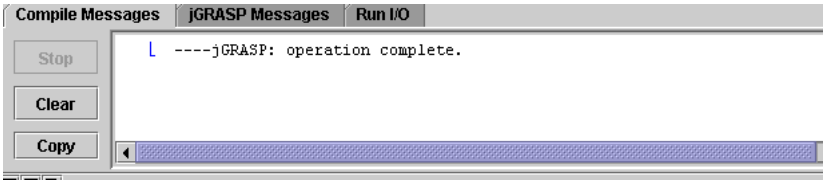
Saving and Compiling

5. Save the program.

6. Compile the program by clicking on the *compile button* on the toolbar. The compile button is represented by a cross:

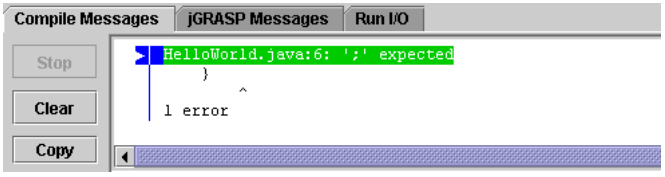
Compile Messages

7. The *Compile Messages* window (at the bottom) should indicate whether the program has been successfully compiled.



Fixing Errors/Bugs

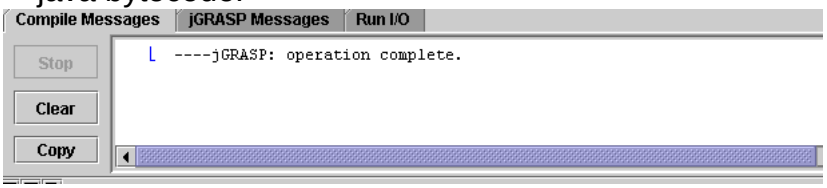
8. At this point you will need to fix any errors which the complier found in your program. These will be indicated in the Compile Messages window. These may seem a little cryptic at first, but they are very helpful in debugging programs



Saving and Compiling

9. Remember to save the file prior to recompiling each time you make a change.

10. Your program is free from errors/bugs when the compiler gives you the *operation complete* message. This means that your program has been translated into java bytecode.



Running the Program

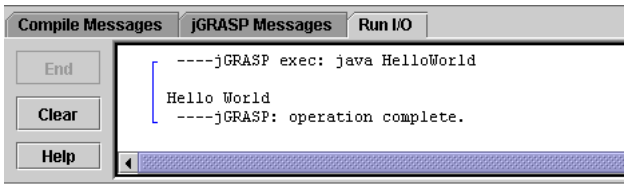
11. Have a look in the directory where you saved the program. You will see that the compiler has created a file called `HelloWorld.class`. This file contains the java bytecode.

12. To run the program, click on the *Run button* on the toolbar. The run button is represented by a little man running:

Output

13. When you run a program, the compile messages window is replaced by a *Run I/O* window.

14. The Run I/O window will contain any output from your program.

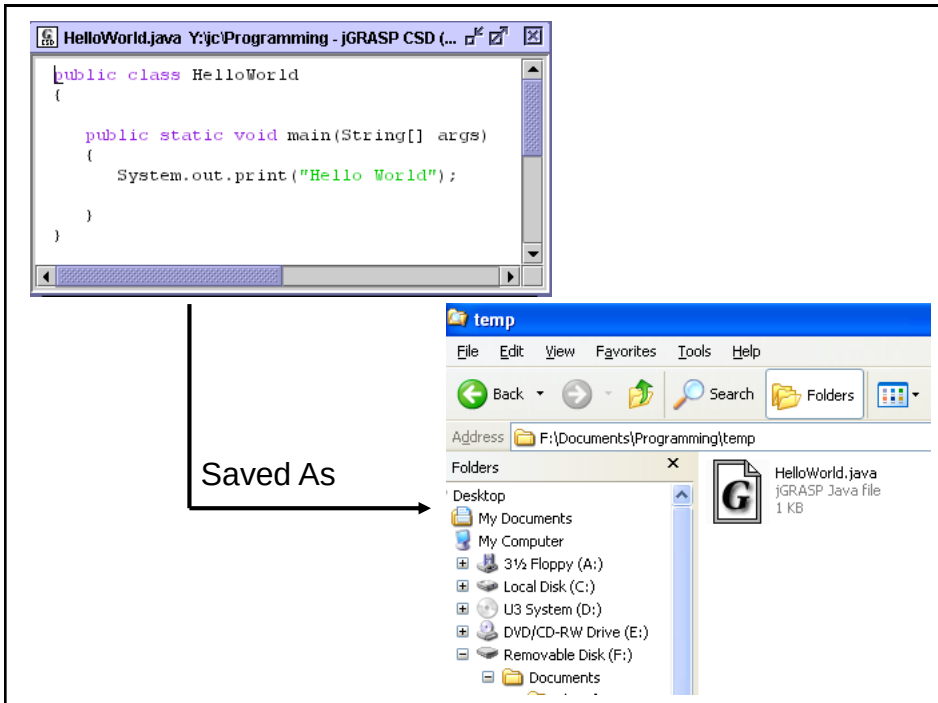


Analysis of the Program

```
public class HelloWorld
{

}
```

- A Java program is known as a **`class`**.
- Every class has a name which should start with a capital letter
- In this example we have called our class `HelloWorld`
- **You <u>must</u> save the program with the same name as the class name**



Analysis of the Program

```
public class HelloWorld
{

}
```

- Everything in the class must be contained between two curly brackets { }
- These tell the compiler where the class begins and ends.
- Every opening curly bracket must have an associated closing bracket.

Analysis of the Program

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

- This line of code starts the method called **`main`**, which is where execution of the program begins.
- A program starts with the first instruction in **`main`**, then executes each instruction in sequence. The program terminates when it has finished executing the final instruction of **`main`**

Analysis of the Program

```
System.out.print("Hello World");
```

- Your `main` method will contain a number of **statements**.
- A statement is an instruction to the computer.
- Every statement must be ended by a semicolon **;**
- Our `main` method contains a single statement.

Analysis of the Program

```
System.out.print("Hello World");
```

- This statement tells the computer to display the string Hello World on screen
- A string is a sequence of characters enclosed by inverted commas.

Analysis of the Program

```
System.out.print("Hello World");
```

- `System.out.print( )` is used to display information to screen.
- We put the information we want to display between the brackets of `System.out.print( )`
- Text to be displayed is enclosed in inverted commas.

The Program

```
public class HelloWorld
{
    public static void main(String [] args)
    {
       // Print the text to screen
       System.out.print("Hello World");
    }
}
```

Comments

```
// Print the text to screen
```

- Comments are explanations and notes which are included in a program.
- Comments are ignored by the compiler – they are written for programmers.
- It is good programming practice to comment your code.

Comments

There are two ways to write comments in Java:

- Single line comment:

```
// This is my single line comment
```

- Multiple line comment:

```
/* This is my multiple line comment.  It
  is designed for longer comments */
```

What you should Comment

1. You should use comments to include the following information in every program:

```
// Title of the program
// Author
// Date
// Purpose of the program
```

What you should Comment

2. You should use comments to explain what the program is doing. These comments can be included anywhere in your program.

```
// Print the text to screen
System.out.print("Hello World");
```

The Program

```
// Practical 1 Question 1
// Joe Bloggs
// Program to print text on screen

public class HelloWorld
{
    public static void main(String [] args)
    {
       // Print the text to screen
       System.out.print("Hello World");
    }
}
```