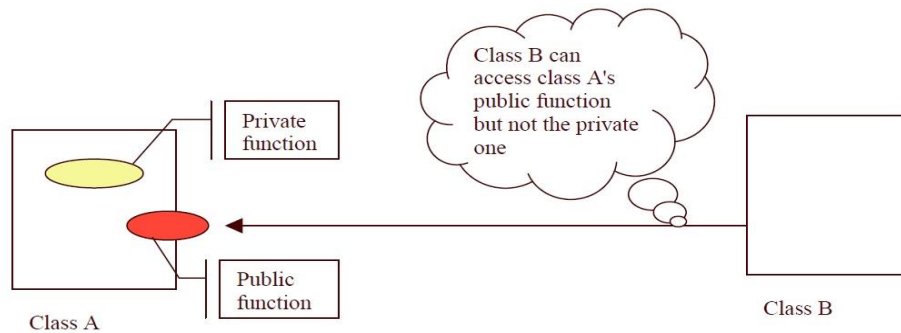


Java is a pure object-oriented language. Which means that everything in this language is an object belonging to a class. Complete programs (and applications) are simply executable classes. There are two types of Java programs

1. Programs that are executed by the Java run-time system from the windows console. I will call these **Applications**.
2. Programs which run with an Internet browser. This second type of program is called an **Applet**. (beyond the scope of this module)

Like C++, Java classes have data values and member functions. Also like C++ properties (i.e., data values) and functions may be described as either public or private or protected. A public item is one that is visible to the outside world and thus useable by objects belonging to other classes. A private item is for internal use and therefore not accessible to other objects outside the class. A protected item is one that may be accessed only by classes derived from the class containing the item.

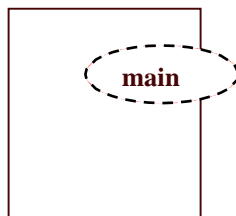


A Java Program is an executable Class

An executable Java class is simply a class that contains certain standard functions. The functions required depend upon whether or not the class is an **Application** or an **Applet**. It is possible to arrange a class so that it runs both as an application and an applet.

A Java **Application** is a class containing a function called **main**. When the program is started, this function is executed.

The simplest example program is usually the "hello world" program you will see an example of this program built written in Java as an Application.



HelloWorld
application

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

Class definition for HelloWorld object

To create the HelloWorld object, you must first type the definition of the HelloWorld class into a text file. Normally in Java each class defined will be in its own text file. In every case the name of the file must be identical to the name of the class being defined but the file must have a .java extension. In this case our text file must be called HelloWorld.java.

The class definition file is then semi-compiled to create a class file which will have the same name as the class but with a .class extension. The tool used to make this class file is the Java compiler called **javac**. **Note that the **javac** and **java** are not recognized by the lab machines so write the program and execute only from NetBeans and do not try the manual way.** If you are using **NetBeans IDE** then you should be able to automatically run the javac compiler with the **Run** (have a look on the Laboratory Resources - **NetBeans Tutorials - Tutorial 1**). To run the javac tool manually then you must go to the windows console, and type

javac HelloWorld.java

If there are no errors, then javac will have created a file called HelloWorld.class in your directory.

In order to execute the HelloWorld program, we must execute the code contained in the HelloWorld.class file. We use a tool called **java** to do this. Again, if you are using **NetBeans IDE** you should be able to run this command from the execute menu. To execute manually simply type at the console

java HelloWorld

The figure below shows the results of executing the compile and then run command manually in a windows console. The output "Hello World" from the program is displayed and the program terminates.



```
C:\WINNT\system32\cmd.exe
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

G:\CCM289\src>javac HelloWorld.java

G:\CCM289\src>java HelloWorld
Hello World

G:\CCM289\src>
```

Some points about the code

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

Java does not use separate definitions from implementations so there is only one file. Java does not have header files. Notice also that this class is defined as public so that it can be used by other classes. It has a function called main which is the function entered when the class is executed.

Java function definitions are similar to C++. However, each function must be individually labelled as *public/private/protected*. There is no public: section as in C++. But like C++ the type of the value returned by the function (here void) is placed before the function's name (main). Any parameters to be passed to the function are enclosed between open and closing round brackets. In this particular case the main function is always expecting to be passed an array of strings called an argument vector (typically called args). This array contains all the other items typed on the command line after the Java class. Therefore, if we type:

java HelloWorld and the rest

The argument vector (an array really) would contain the three strings "and", "the", and "rest". Argument vectors are a way of passing data to the running program. If no data is passed, then the array will contain no data.

Java has a String class. Strings are different from arrays of char (they are the same in C++). The String class contains many operations which are documented in Java's on-line html based documentation. You should consult the documentation constantly and treat it as a valuable programming resource (<http://docs.oracle.com/javase/8/docs/api/>). Strings may be represented by items in double quotes, i.e. "this is a string".

Like C++ there are standard I/O stream objects defined in the System package called

- System.in - for input from the keyboard
- System.out - for output to the screen
- System.err - for error messages.

Output is performed using the functions **print** and **println** defined in the stream class. Both these functions take a string as input and display it on the screen when applied to the System.out stream. Note that because streams are objects we use the standard dot (.) notation to call the functions; hence we get:

```
System.out . println("this will be displayed on the screen");
```



Example using Argument Vector and Converting Strings to int's

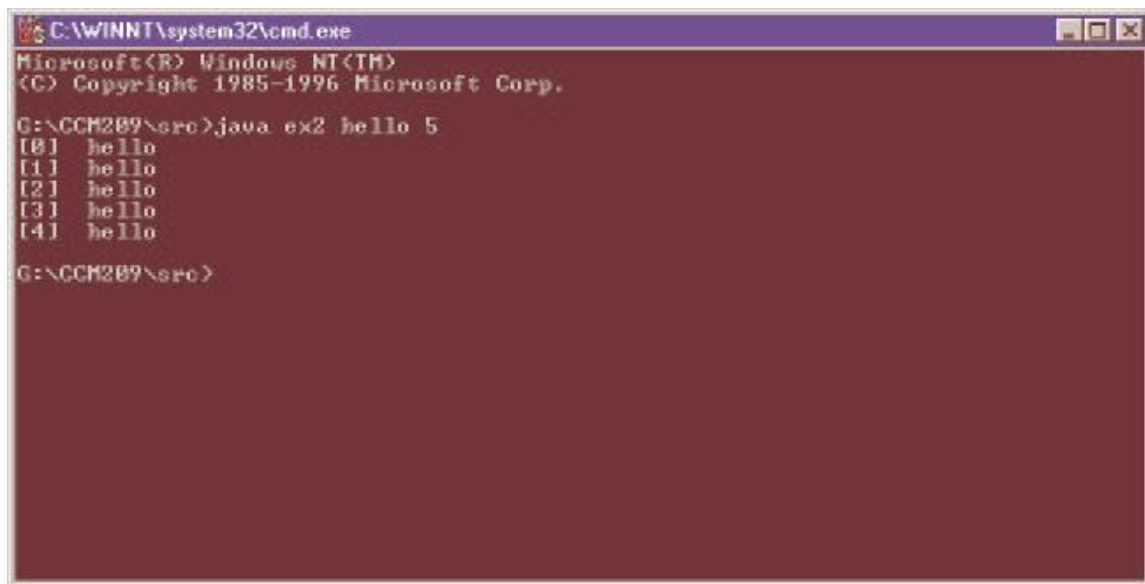
If you wish to input an integer (int) into a program it must be passed in as a string. The program must then convert this String into an int using the Integer.parseInt() function. Implement and run ex2.java

```
/* program to output a string a given number of times */

public class ex2 {
    public static void main(String[] args) {
        /* check if no argument is passed */
        if (args.length != 2) {
            System.err.println("Usage: java ex2 <String> <int>");
            System.err.println("you must supply two argument values");
            System.exit(0);
        }
        else {
            String name = args[0];           // the first argument
            int num = Integer.parseInt(args[1]); // the second argument

            for (int i = 0; i < num; i++) {
                System.out.println "[" + i + " ] " + name);
            }
        }
    }
}
```

The output of this example is shown below

A screenshot of a Windows command prompt window titled "C:\WINNT\system32\cmd.exe". The window shows the output of the command "java ex2 hello 5". The output consists of five lines, each starting with a bracketed index followed by the word "hello":
[0] hello
[1] hello
[2] hello
[3] hello
[4] hello
The prompt "G:\CCM2B9\src>" is visible at the bottom of the window.

You can pass arguments using **NetBeans IDE** as follows:

1. Click Run >> Set Project Configuration >> Customize
2. Then type the arguments in the Arguments field
3. Make sure that the Main Class of the project is the correct one
4. Click OK and Run your Java program (or press F6)

Using the same ex2 above (do not modify the code) try to output (by passing arguments) the following String:

Hello World

which consists of two words; ten times.

Lessons to be learnt from ex2

1. Comments in Java are identical to comments in C++
2. If .. then .. else statements (and switch statements) are identical in Java and C++
3. For loops (and while loops and do-while loops) are identical in Java and C++
4. Boolean operators (!= and ==) plus all the others are identical in Java and C++
5. int declarations (and float, double , char) are identical in Java and C++.
6. Arrays in Java are indexed from 0 identical to C++. Notice that the first command line argument is in the 0th position in the array args.

In fact if we exclude classes and libraries such as I/O most of the core syntax of Java is identical to the syntax of C++. But here are some differences

1. Arrays have a **length** property so unlike C++ you can always find the length of an array. For example in the program above **args.length** returns the size of the array, which would be 2 in the example.
2. You can terminate a program by calling System.exit(0)
3. Strings can be concatenated together using the + operator. i.e. "jig" + "saw" returns "jigsaw"
4. A String may be converted into an int using the function Integer.parseInt(<String>). If the String cannot be converted into a valid int (an error is generated in the form of a NumberFormatException). Have a look at the [Java Documentation](#) for the String and Integer classes.