# Our First Java Program

```
/* Comment 1:
   Written by George Wells  --  6 February 2001 */

public class HelloWorld
  {
    public static void main (String[] args)
      {
        System.out.println("Hello, world\n");
                                          // Comment 2
      } // main

  } // class HelloWorld
```

# Looking at the program: Strings and Escape Sequences

```
/* Comment 1:
   Written by George Wells  --  6 February 2001 */

public class HelloWorld
  {
    public static void main (String[] args)
      {
        System.out.println("Hello, world\n");
                                          // Comment 2

      } // main

  } // class HelloWorld
```

Escape Sequence

# Input and Output

- Output:
  - **`System.out`** is an *output stream* (connected to the screen by default)
  - **`println`** is a *method* to display information
- Input:
  - **`System.in`** is an *input stream* (connected to the keyboard by default)
  - More complicated! (wait until Chapter 10)

# Compiling and Running Java Programs

- A little different!
- DOS command-line tools
- Compiler: **`javac`**

**Case is significant!**

# Compiling (cont.)

- The compiler produces a **`.class`** file
  - *Bytecode* (not machine code)
  - Allows portability (Java programs can run on almost any hardware and operating system)
- Needs an *interpreter*
  - The Java Virtual Machine (JVM)
  - A program that “executes” bytecode

# Compiling

HelloWorld.java
public class HelloWorld
{ ...
javac HelloWorld.java
HelloWorld.class
File of bytecodes

# Running

`HelloWorld.class`

`java HelloWorld`

Run the interpreter

Command Prompt

```
D:\JavaProgs>java HelloWorld
Hello, world
```

# A Second Example Program

```
public class Factorial
  { public static int fact (int n)
    //============================
      { int k, result=1;
        for (k=1; k<=n; k=k+1)
          { result = result * k;
          }
        return(result);
      } //fact

    public static void main (String a[])
    //===================================
      { System.out.print("Factorial of 7 = ");
        int f = fact(7);
        System.out.println(f);
      } // main
  } // class Factorial
```

# Declarations

```
public class Factorial
  { public static int fact (int n)
    //============================
      { int k, result=1;
        for (k=1; k<=n; k=k+1)
          { result = result * k;
          }
        return(result);
      } //fact

    public static void main (String a[])
    //=================================
      { System.out.print("Factorial of 7 = ");
        int f = fact(7);
        System.out.println(f);
      } // main
  } // class Factorial
```

Type comes before identifier

Can have *initialiser*

# Parameters are Passed by Value

```
public class Factorial
  { public static int fact (int n)
    //============================
      { int k, result=1;
        for (k=1; k<=n; k=k+1)
          { result = result * k;
          }
        return(result);
      } //fact

    public static void main (String a[])
    //===================================
      { System.out.print("Factorial of 7 = ");
        int f = fact(7);
        System.out.println(f);
      } // main
  } // class Factorial
```

Changing **n** has *no* effect outside the method

# The **for** Loop

Initialisation

Termination test

Increment

Body

```
public static int fact (int n)
 //==============================
   { int k, result=1;
     for (k=1; k<=n; k=k+1)
       { result = result * k;
       }
     return(result);
   } //fact
```

# Returning a Value from a Method

```
public static int fact (int n)
 //=============================
    { int k, result=1;
      for (k=1; k<=n; k=k+1)
        { result = result * k;
        }
      return(result);
    } //fact
```

Leaves the method, returning the result

# Another Example: An Object!

```
class Factorial
  { public int fact (int n)
    //=====================
      { int k, result=1;
        for (k=1; k<=n; k=k+1)
          { result = result * k;
          }
        return(result);
      } //fact
  } // class Factorial

public class Program3
  { public static void main (String args[])
    //=================================
      { Factorial f = new Factorial();
        System.out.println("Factorial of 7 = ”
                                + f.fact(7));
      } // main
  } // class Program3
```

# Two Classes

```
class Factorial
  { ...
  } // class Factorial

public class Program3
  { ...
  } // class Program3
```

- Only one **`public`** class (filename: **`Program3.java`**)
- Good rule of thumb:
  - One class per file