

Programming Practice with Java

One of the first programs in any language is usually to output "Hello World".

In Java this First program appears quite complex, as shown below.

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

This Java program involves 3 kinds of parentheses (curly, round, square), and many advanced concepts such as class, method string, array, and some words (public, static, void, main, args, System, println, etc) which cannot be understood easily at this early point. The JJ environment introduces the main ideas of Java in gentle stages.

In JJ this entire program can be done as a single line:

```
JJS.outputString ("Hello World");
```

Here outputString is imported from **JJS** (JJ System) and it prints out the string "Hello World" which is between the brackets.

A Simple Second Program: with some computation (multiplication) follows; code is at the left; an execution or run is at the right in a dotted box.

```
// Name An Onymous
// Does simple computation with integers

JJS.outputString ("10*9*8*7*6*5*4*3*2*1 = ");
JJS.outputInt    ( 10*9*8*7*6*5*4*3*2*1 );
```

```
10*9*8*7*6*5*4*3*2*1 = 3628800
```

Comments follow two slashes (//) to the end of the line. They can provide a description (documentation) but are ignored by the computer.

Strings are the contents between quotes ("). The **outputString** which is imported from the class JJS (JJ System) prints the string within the quotes exactly.

Multiplication is a mathematical operation which is indicated by an asterisk (*).

outputInt, (spoken as "output line integer") is a command or action which evaluates the arithmetic integer expression, prints its value and then outputs a new line.

outputInt is a similar action; it outputs the expression but does not follow with a new line (**ln**), so that any later output occurs on the one same line.

Semicolons are required after each command; they are not needed after comments.

Case sensitivity refers to upper and lower case letters being different, so **outputString** must begin with lower case "o" and the inner word "String" must begin with a capital "S". Similarly **JJS** must be all caps; it is not **jjs**, **Jjs**, **jjs** or even **J J S**.

Type refers to the kind of data and the operations possible on that data.
The most common types are integer and real; others will be considered later.
Integers come from counting; reals come from measuring.

Integer numbers (of type **int**) are whole numbers, which arise from counting such things as populations, product, occurrences, etc.
Integers may be positive, or negative. Examples are:

7, 11, 360, 0, -7, 12345678

The upper (and lower) limits are around 2 billion (plus or minus).

The actual range of ints is

-2,146,473,648 to +2,147,483,647

Real numbers (of type **double**) arise from measuring such things as lengths, areas, volumes, speeds, sizes, etc.
Reals involve a radix point (or decimal point in the base 10 system).
Examples are:

1.23, -56.789, 0.5, 7.0, 12345.6789

The actual values of doubles involve about 14 significant digits.

Operations on both ints and doubles include the four arithmetic functions: addition, subtraction, multiplication, division, and others (square root, etc).
Division of ints results in a integer; the decimal part is chopped off.

Logical values (of type boolean) will be considered later.

Program Structure

Programs consist of 3 parts:

Description (or documentation)

Declaration (of data and their types)

Directives (or commands, instructions)

An example of a Java program written in the JJ environment follows in the box below where the three parts are separated by dashed lines (not part of the code).

Runs (or executions) of this code are always shown in a dotted box at the right.

```
// Name OmynousAnn
// Does compute the circumference given the diameter
// Shows a typical simple computation with double type
-----
double PI;                // variables
double radius, diameter; // ( boxes )
-----
JJS.outputLnString ("Enter the diameter:"); // prompt
diameter = JJS.inputDouble ();             // input
JJS.outputLnDouble (diameter);            // echo

radius = diameter / 2.0;
PI = 3.14159;

JJS.outputLnString ("The circumference is ");
JJS.outputLnDouble ( 2.0 * PI * radius );
```

```
Enter the diameter:
2.0
The circumference is
6.28318

Enter the diameter:
0.0
The circumference is
0.0

Enter the diameter:
-20.0
The circumference is
-62.8318
```

A description of this code follows.

You should modify this code to compute the area also.

Why was the radius not input, instead of the diameter?

Description or documentation is not required, but is nevertheless useful. It consists of a number of comments, beginning with two slashes (//) and continuing to the end of the line. Comments should include **who** wrote the code, **what** the code does, and possibly describing **why, when, where** but not necessarily **how**, for the code shows how.

Declaration of data associates (or binds, connects) names with boxes of a specified type. Variable declarations consist of lines beginning with a type (int, double, etc) and followed by the names of boxes of that type. Names begin with a letter and can be followed by any number of letters or digits. Sometimes comments may be used to clarify the names and allow shorter names to be used. Examples of Declarations are:

```
int count;  
double radius, length, width, height; // size in meters  
int age, populationOver21, populationOfLA;
```

Directives (or actions, instructions) are commands, such as input, output, and set or assignment (and others later, such as ifs and whiles). All commands end with semi-colons.

Assignment, or set, puts a value into a box (variable) which has been declared to be of a given type; the set command has the form:

```
<name> = <expression> ;
```

such as

```
count = 0;  
price = 7.85;  
rate = maxRate;  
area = length * width;
```

Convert is a program which converts a real temperature (of type double) from degrees Celsius into degrees Fahrenheit; Some runs are shown at the right.

```
// File Convert
// Name A Nonymous
// Does Convert temperatures c to f

double c; // temperature Celsius
double f; // temperature Fahrenheit

JJS.outputInString ("Enter degrees C : ");
c = JJS.inputDouble ();
JJS.outputInDouble (c);

f = (9.0 / 5.0) * c + 32.0;

JJS.outputInString ("Degrees F is: ");
JJS.outputInDouble ( f );
```

```
Enter degrees C:
100.0
Degrees F is:
212.0

Enter degrees C:
0.0
Degrees F is:
32.0

Enter degrees C:
-40.0
Degrees F is:
-40.0
```

Testing, at the right shows some interesting values: that -40 is the same value in Celsius and Fahrenheit.

Modify this code to convert from Fahrenheit to Celsius.
Test other values (such as 0 degrees fahrenheit).
Does the midpoint of 50 celsius map to mid 122 fahrenheit?