

```
Hello, wikibooks!
```

or, if you do not pass a command line parameter, it will simply print

```
Hello, World!
```

Wikipedia has a hello world sample for each type of java code, for more information see [w:Java \(programming language\)#Hello World](#)

Common Problems

If the program does not work as you expect, check the following common errors.

- Are you sure all words are spelled correctly and with the exact case as shown?
- Are there semicolons and brackets in the appropriate spot?
- Are you missing a quote? Usually, modern IDEs would try coloring the entire source as a quote in this case.
- Are you launching the javac or java correctly? Javac requires the full filename with the .java extension, while java requires the class name itself.

The Next Step

Now that you have seen the classic *Hello, World* program in Java, let's move on to a more realistic example which highlights the object oriented nature of Java. Visit [Understanding a Java Program](#) which presents key Java language features along with a more complete explanation of the syntax and structure of a basic Java program.

Beginner topics	User Interface topics	Advanced topics
• Overview • Getting started • Language fundamentals • Classes and objects • Basic arithmetic • Collections • Exceptions • Concurrent Programming • Javadoc & Annotations • Byte Code	• Basic I/O • Event handling • Intro to Swing • Graphics • Applets	• Networking • Database programming • Reflection • Design patterns • Libraries, extensions and frameworks • 3D programming • Threads
Extra Reading		
Appendices • Links • Glossary • Index		

Understanding a Java Program

Navigate **Getting Started** topic:

- Installing Java on Your Computer
- Compiling programs
- Running Java programs
- Developing the first Java program
- Understanding a Java program

This article presents a small Java program which can be run from the console. It computes the distance between two points on a plane. You need not understand the structure and meaning of the program just yet; we will get to that

soon. Also, because the program is intended as a simple introduction, it has some room for improvement, and later in the module we will show some of these improvements. But let's not get too far ahead of ourselves!

The Distance Class: Intent, Source, and Use

This class is named *Distance*, so using your favorite editor or Java IDE, first create a file named `Distance.java`, then copy the source below and paste it into the file and save the file.

```
public class Distance
{
    private java.awt.Point point0, point1;

    public Distance(int x0, int y0, int x1, int y1)
    {
        point0 = new java.awt.Point(x0, y0);
        point1 = new java.awt.Point(x1, y1);
    }

    public void printDistance()
    {
        System.out.println("Distance between " + point0 + " and " + point1
                           + " is " + point0.distance(point1));
    }

    public static void main(String[] args)
    {
        Distance dist = new Distance(
            intValue(args[0]), intValue(args[1]),
            intValue(args[2]), intValue(args[3]));
        dist.printDistance();
    }

    private static int intValue(String data)
    {
        return Integer.parseInt(data);
    }
}
```

At this point, you may wish to review the source to see how much you might be able to understand. While perhaps not being the most literate of programming languages, someone with understanding of other procedural languages such as C, or other OO languages such as C++ or C#, will be able to understand most if not all of the sample program.

Once you save the file, compile the program:

```
javac Distance.java
```

(If the `javac` command fails, review the Java installation instructions.)

To run the program, you supply it with the *x* and *y* coordinates of two points on a plane. (For this version of *Distance*, only integer points are supported.) The command sequence is

```
java Distance x0 y0 x1 y1
```

to compute the distance between the points (x_0, y_0) and (x_1, y_1)

For example, the command

```
java Distance 0 3 4 0
```

will compute the distance between the points (0,3) and (4,0) and print the following:

```
Distance between java.awt.Point[x=0,y=3] and java.awt.Point[x=4,y=0] is
5.0
```

The command

```
java Distance -4 5 11 19
```

will compute the distance between the points (-4,5) and (11,19):

```
Distance between java.awt.Point[x=-4,y=5] and java.awt.Point[x=11,y=19]
is 20.518284528683193
```

We'll explain this strange looking output, and also show how to improve it, later.

Detailed Program Structure and Overview

As promised, we will now provide a detailed description of this Java program. We will discuss the syntax and structure of the program and the meaning of that structure.

Introduction to Java Syntax

The *syntax* of a Java class is the characters and symbols and their structure used to code the class using Unicode characters. A fuller treatment of the syntax elements of Java may be found at [Syntax](#). We will provide here only enough description of the syntax to grasp the above program.

Java programs consist of a sequence of tokens. There are different kinds of tokens. For example, there are word tokens such as `class` and `public` which represent *keywords* - special words with reserved meaning in Java. Other words (non keywords such as `Distance`, `point0`, `x1`, and `printDistance`) are *identifiers*. Identifiers have many different uses in Java but primarily they are used as names. Java also has tokens to represent numbers, such as `1` and `3`; these are known as *literals*. *String literals*, such as `"Distance between "`, consist of zero or more characters embedded in double quotes, and *operators* such as `+` and `=` are used to express basic computation such as addition or String concatenation or assignment. There are also left and right braces (`{` and `}`) which enclose *blocks*. The body of a class is one such block. Some tokens are punctuation, such as periods `.` and commas `,` and semicolons `;`. You use *whitespace* such as spaces, tabs, and newlines, to separate tokens. For example, whitespace is required between keywords and identifiers: `publicstatic` is a single identifier with twelve characters, not two Java keywords.