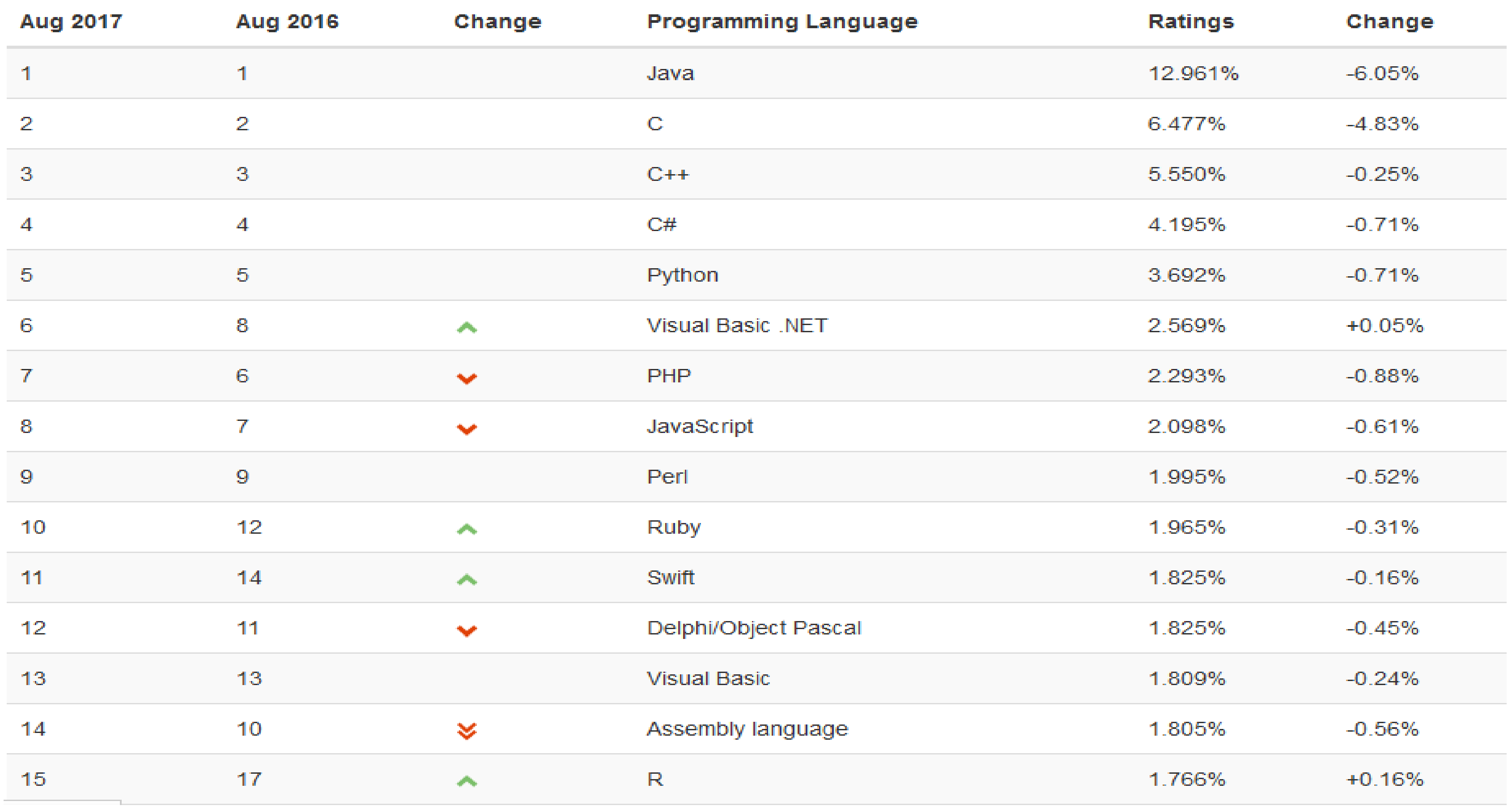
# B.Sc. In Software Development. Year 3. Applications Programming. Introduction To Java.

LIMERICK INSTITUTE OF TECHNOLOGY
SCHOOL OF SCIENCE, ENGINEERING & I.T.
*Department of Information Technology*

## History

- Developed by James Gosling & Bill Joy (Sun).
- Originally called Oak.
- Initially developed out of a need to supply software for consumer electronics.
- Renamed Java early 1995.
    - Redesigned for Internet Applications.
    - Demonstrated for the first time at SunWorld May 10 1995.
- On January 27, 2010, Sun was acquired by Oracle for $7.4 bn.
- Java 9 is expected to be released in September 2017.

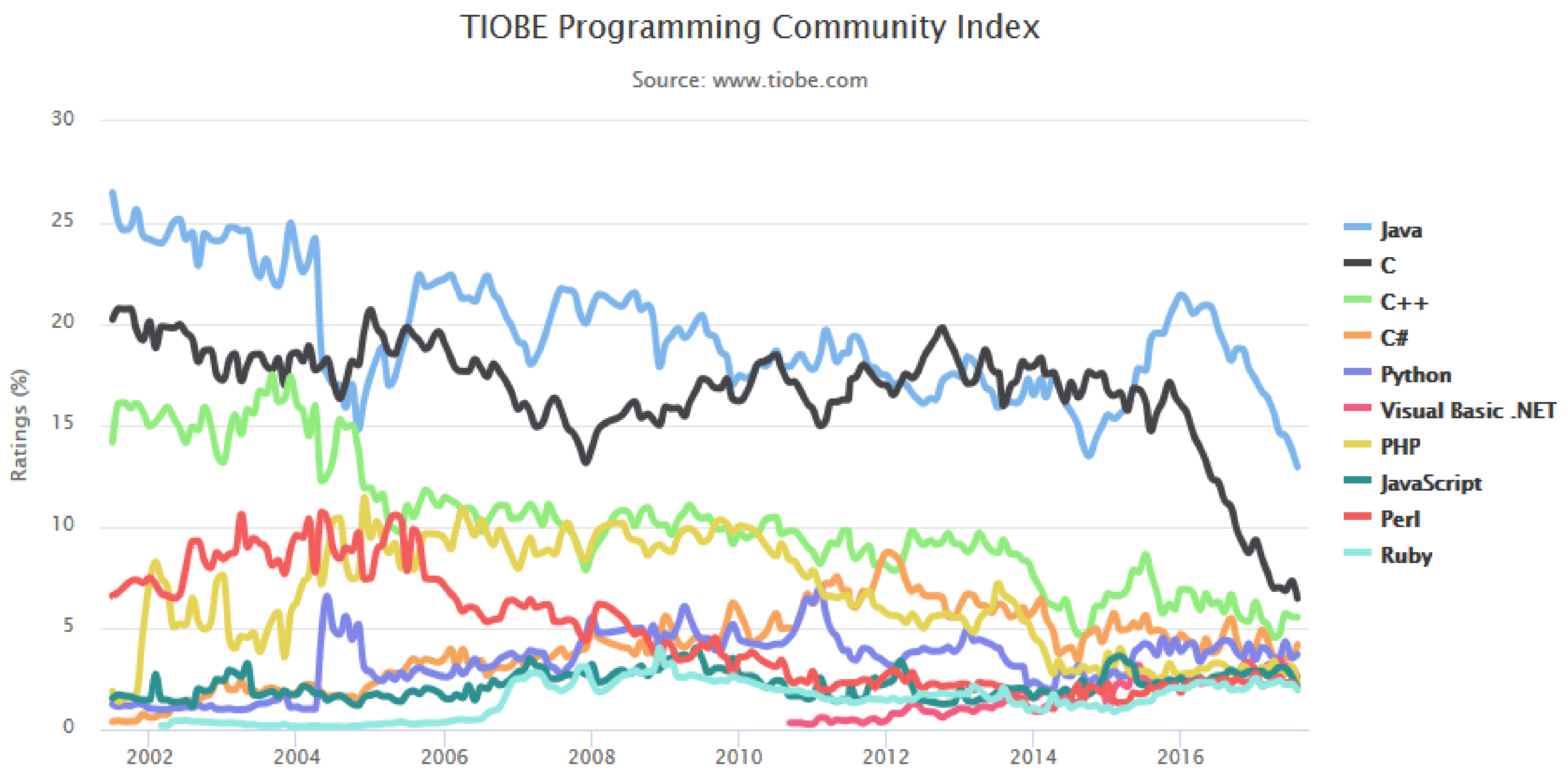
# Relevance of Java



| Aug 2017 | Aug 2016 | Change | Programming Language | Ratings | Change |
|---|---|---|---|---|---|
| 1 | 1 | | Java | 12.961% | -6.05% |
| 2 | 2 | | C | 6.477% | -4.83% |
| 3 | 3 | | C++ | 5.550% | -0.25% |
| 4 | 4 | | C# | 4.195% | -0.71% |
| 5 | 5 | | Python | 3.692% | -0.71% |
| 6 | 8 | ▲ | Visual Basic .NET | 2.569% | +0.05% |
| 7 | 6 | ▼ | PHP | 2.293% | -0.88% |
| 8 | 7 | ▼ | JavaScript | 2.098% | -0.61% |
| 9 | 9 | | Perl | 1.995% | -0.52% |
| 10 | 12 | ▲ | Ruby | 1.965% | -0.31% |
| 11 | 14 | ▲ | Swift | 1.825% | -0.16% |
| 12 | 11 | ▼ | Delphi/Object Pascal | 1.825% | -0.45% |
| 13 | 13 | | Visual Basic | 1.809% | -0.24% |
| 14 | 10 | ▼▼ | Assembly language | 1.805% | -0.56% |
| 15 | 17 | ▲ | R | 1.766% | +0.16% |

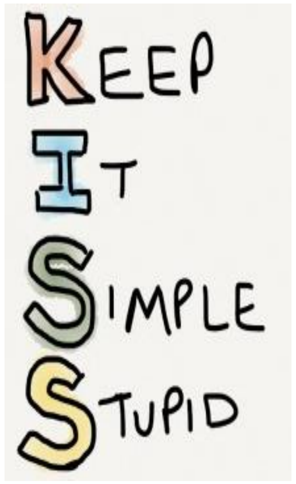
*Source: https://www.tiobe.com*

# Relevance of Java



## Characteristics of Java: *Simplicity*

- Java is similar syntactically to C++ and C#.
- Java omits many rarely used, poorly understood, confusing features of C++ that tend to bring more grief than benefit. For example:
    - No pointers in Java.
    - No multiple inheritance in Java.
    - Uses automatic memory allocation & garbage collection.



- This is one of the reasons why Java was so quickly adopted by industry.

## Characteristics of Java: *Distributed*

- Distributed computing involves several computers working together on a network.
- Since networking capabilities are inherently integrated into Java, writing networked applications with Java is relatively easy.

## Characteristics of Java: *Interpreted*

- You need an interpreter to run Java programs. The programs are compiled into Java Virtual Machine code called bytecode. The bytecode is machine-independent and can run on any machine that has a Java interpreter

- Usually, a compli                    er such as a C++ compiler, translates a program in a high-level language to machine code. The code can only run on the native machine. If you wish to run the program on other machines, it has to be recompiled on the native machine.

- For example, if you compile a C++ program on Windows, the executable code generated by the compiler can only run on Windows. With Java, the bytecode generated by the compiler can run on any platform for which a Java interpreter exists.

## Characteristics of Java: *Robustness*

- Strongly typed language.
- Java compilers can detect many errors that would first show up at execution time in other languages.
- Java has eliminated certain types of error-prone programming constructs found in other languages.

# Characteristics of Java: *Security*

- Used predominately as a server-side and mobile device programming language, Java is used in a networked and distributed environments.
- Security is of paramount importance.
- Cryptography, Authentication and Access Control, Platform Security (strong data typing, memory management, bytecode verification, etc) are all inherent in Java.

*"If you're concerned about privacy issues, put on some pants and close the windows."*

## Characteristics of Java: *Performance*

- It is Java's performance which often gets the most criticism.
- In the early days Java used to run up to 20 times slower than C.
- However, this figure is became more negligible as processing speeds of computers improved.
- Oracle have addressed this issue by improving the performance of its JVM  and its compiler.
- Java's performance is on a par with mostly commercially successful languages.

## Characteristics of Java: *Multithreaded*

- Multithreading is intrinsic to the Java language.

- The benefits of multithreading are better interactive responsiveness and real-time behaviour.

# JDK Versions

- JDK 1.01 (1995)
- JDK 1.1 (1996)
- :
- :
- JDK 6 (December 2006)
- JDK 7 (July 2011)
- JDK 8 (March 2014)
- JDK 9 (due September 2017)

## Java IDE Tools

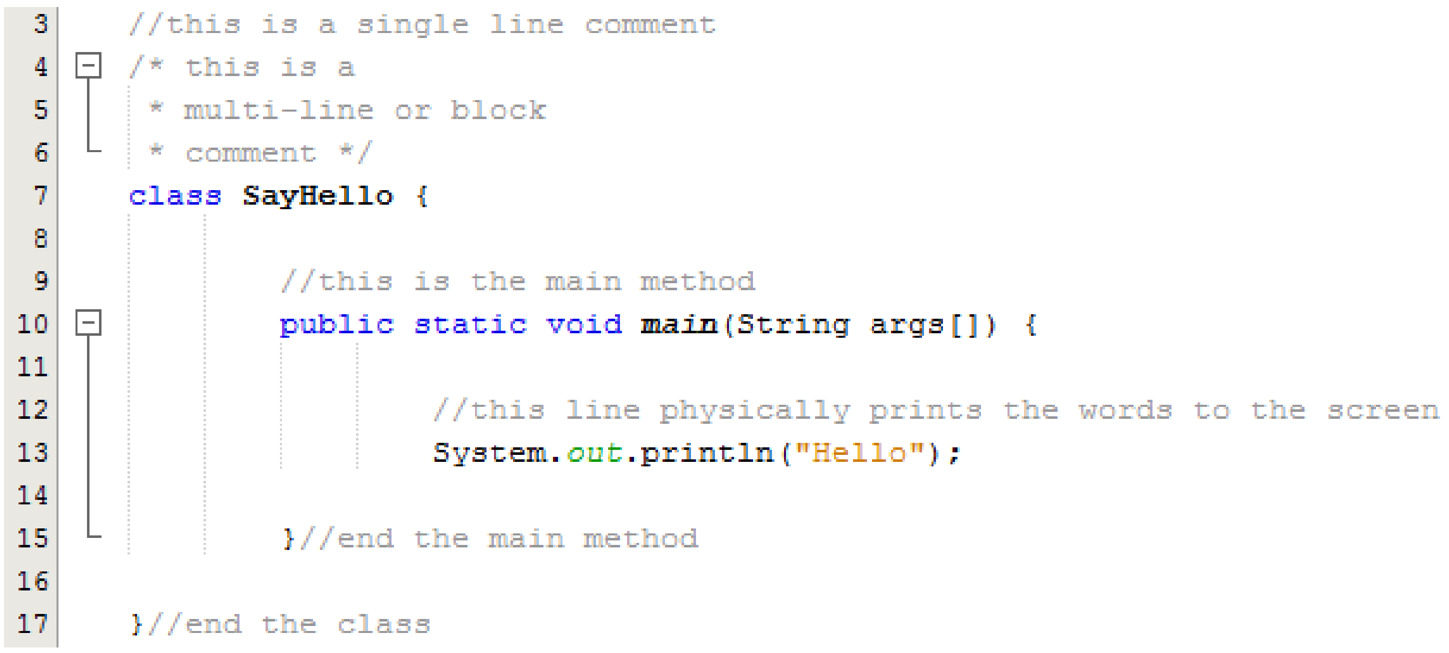
- Netbeans (http://netbeans.org/)
- Eclipse (https://www.eclipse.org/)
- IntelliJ (https://www.jetbrains.com/idea/)
- JCreator (http://www.jcreator.com/)
- Jedit (http://www.jedit.org/)

## Anatomy Of A Java Program

- Programs generally have the following constructs.
  - Comments.
  - Reserved words.
  - Modifiers.
  - Statements.
  - Blocks.
  - Classes.
  - The main method.

# Comments

- By carefully commenting your code, a tool called Javadoc can parse your code and automatically generate HTML documentation.



```
//this is a single line comment
/* this is a
 * multi-line or block
 * comment */
class SayHello {

        //this is the main method
        public static void main(String args[]) {

                //this line physically prints the words to the screen
                System.out.println("Hello");

        }//end the main method

}//end the class
```

## Reserved Words

- Reserved words (keywords), are words that have a specific meaning to the compiler and cannot be used for other purposes in the program.
- For example, when the compiler sees the word `class`, it understands that the word immediately after it is actually the name for the `class`.

## Modifiers

- Java uses several reserved words called modifiers that specify the properties of data.
- Modifiers govern how data can be accessed.
- Examples of modifiers are `public` and `static`.
- Other modifiers are `private`, `abstract`, `final` and `protected`.

## Statements

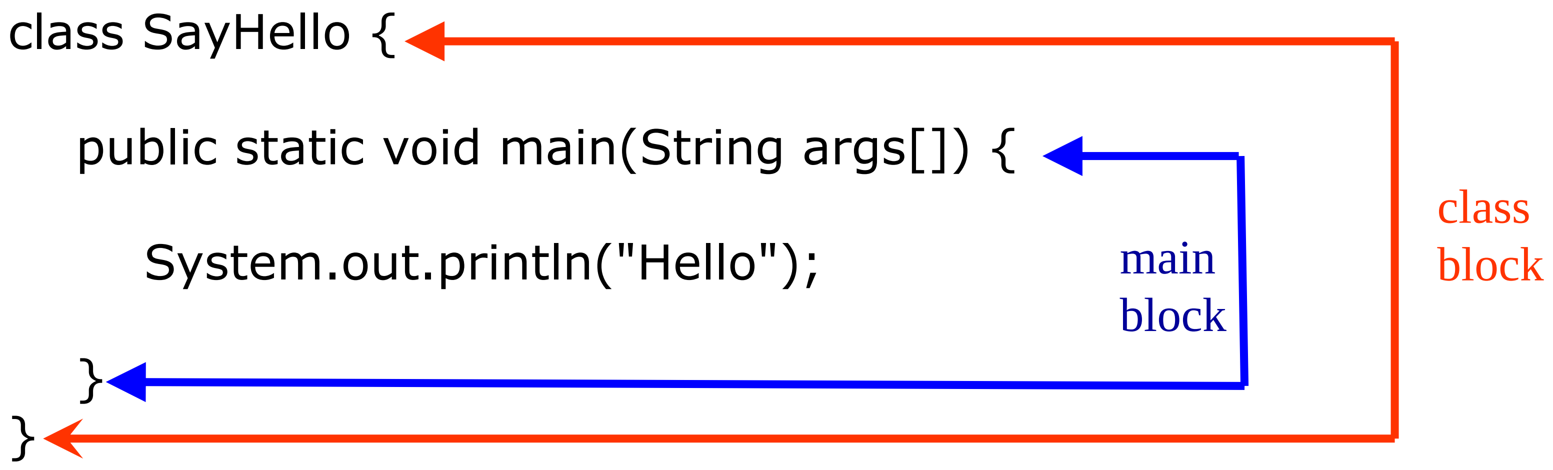
- A statement represents an action or a sequence of actions.
- The following statement is used to print the word Hello to the screen.

```
System.out.println("Hello");
```

- Every statement in Java ends with a semicolon.

# Blocks

- The braces/brackets in the program for a block that groups the components of a program.
- In Java each block begins with an opening brace { and ends with a closing brace }.
- Blocks can be nested:



```
class SayHello {

    public static void main(String args[]) {

        System.out.println("Hello");

    }
}
```

## Classes

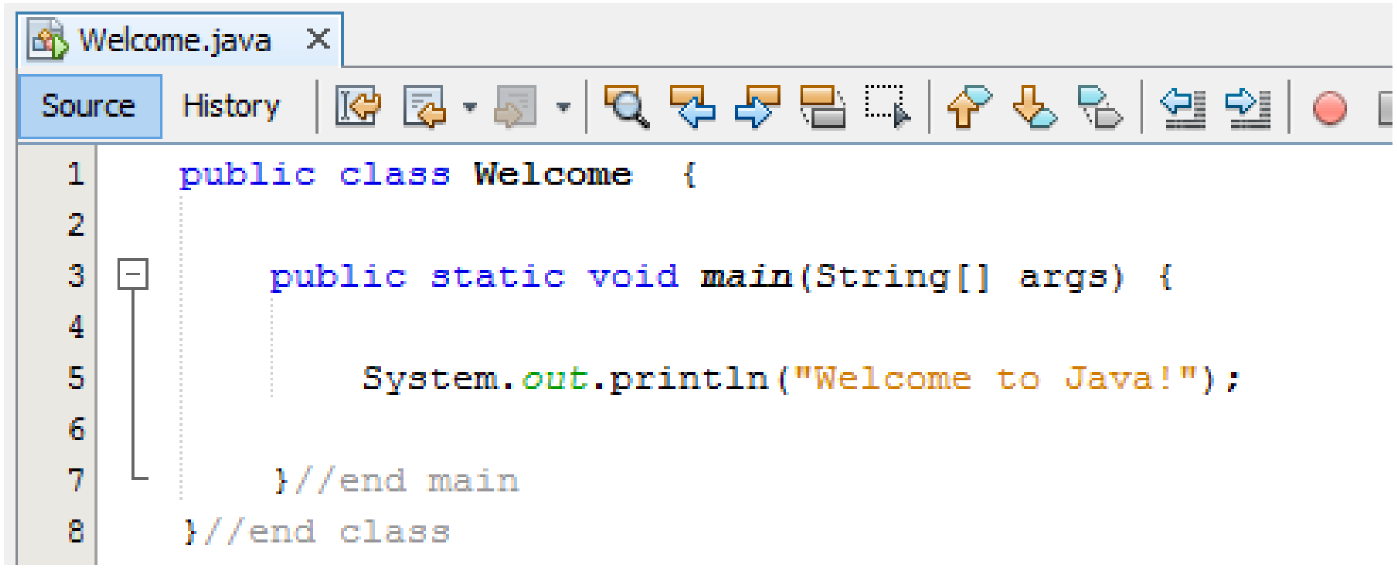
- The class is the essential Java construct.
- To program in Java you must understand classes and be able to write and use them.
- As mentioned earlier the Java API contains predefined programs that are bundled with the language which developers are free to use.
- These predefined programs are actually classes that somebody in Sun Microsystems has written.
  - They aid program development.

## The main method

- All programs that you intend running must have a `main` method.
- A method in Java is a collection of statements.

# A Simple Application

**Program Specification:** Write a program which will print the text *"Welcome To Java"* to standard output.



```
public class Welcome  {

    public static void main(String[] args) {

        System.out.println("Welcome to Java!");

    }//end main
}//end class
```

## A Simple Program

- In this program,

  println(“Welcome To Java”);

  Is the statement that prints the required message.
- Every Java program must have at least one class. Each class begins with a class declaration that defines data and methods for the class.
- In this example, the class name is Welcome.
- A Java file can contain more than 1 class, but only one of them can be public.
- The name you save the file under, must be the name of the public class.
- In this example, the file would be saved under the name “Welcome.java”.

## A Simple Application

- This class contains a method named `main`.

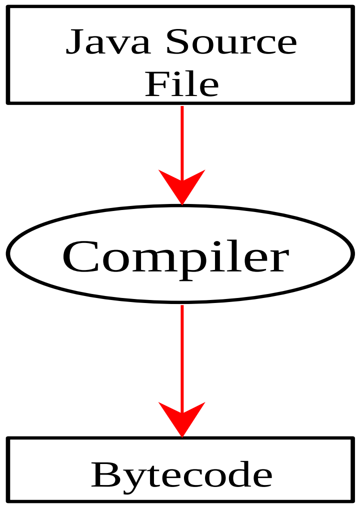
- The `main` method in this program contains the `println` statement.

- The `main` method is invoked by the interpreter.

## Compiling Programs

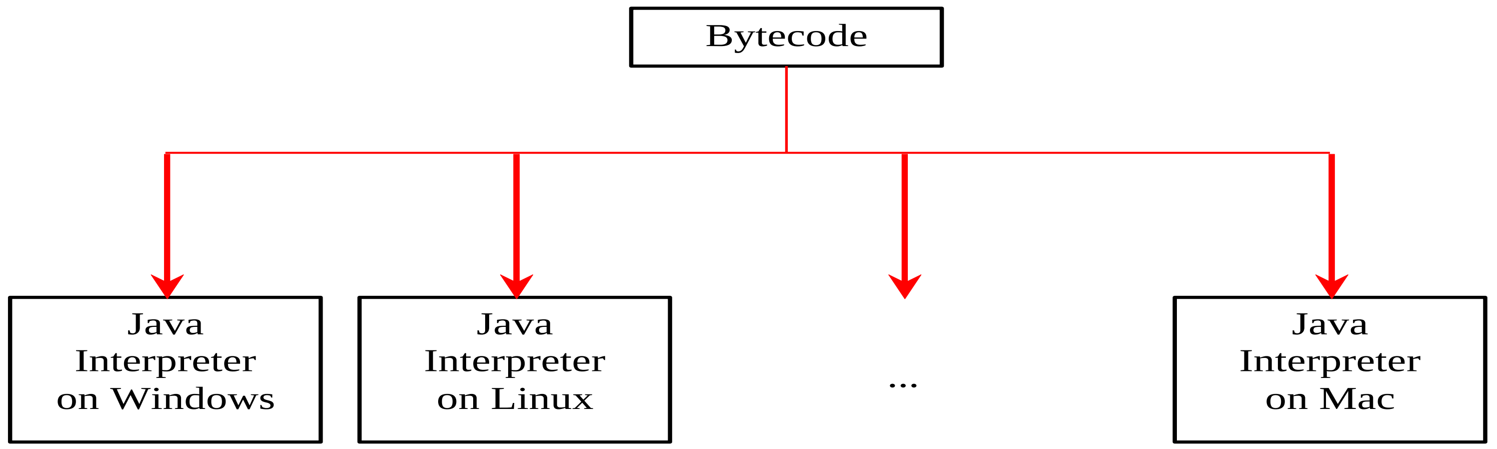
- Most of the time you will build your applications from within Netbeans.
- However, Java apps. can also be compiled from the command line

```
javac file.java
```

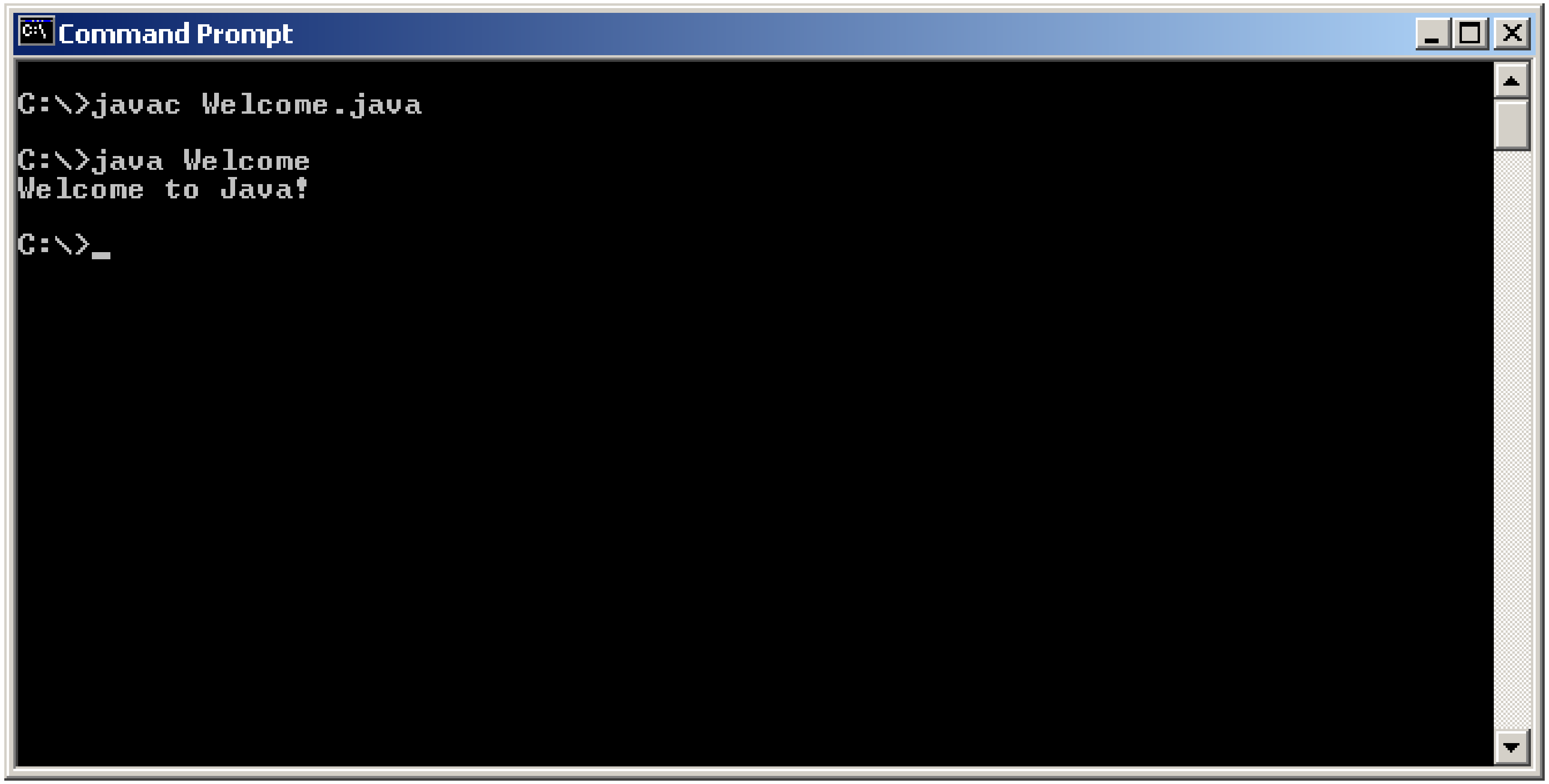


## Executing Applications

- Again, the majority of the time you will execute/run your applications from inside Netbeans.
- However, they can also be run from the command line.
- The syntax is: java *className*



## Example of using the Command Line



```
C:\>javac Welcome.java

C:\>java Welcome
Welcome to Java!

C:\>_
```