

Introduction to Java

Java

- ▶ Developed for home appliances - cross-platform VM a key feature
- ▶ Originally called Oak
- ▶ Gained notariety with HotJava web browser that could run “programs over the internet” called applets
- ▶ Gained popularity when Netscape included Java VM in Navigator web browser
- ▶ JavaScript is purely a marketing label meant to capitalize on Java hype - there is no relationship between Java and JavaScript
- ▶ Java is a general-purpose application programming language.
- ▶ Java applets are now very rare. The bulk of Java code runs on (web) servers.

The Java Programming Language

- ▶ Java is part of the C family. Same syntax for variable declarations, control structures
- ▶ Java came at a time when C++ was king. C++ was a notoriously complex object-oriented extension to C.
- ▶ Java improved on several key aspects of C++, greatly simplifying software development
- ▶ Two most compelling features of Java were cross-platform deployability (“write once, run anywhere”) and automatic garbage collection
- ▶ These two advantages, especially garbage collection, drove Java adoption

The Java Platform

Three components of the Java platform:

- ▶ The Java programming language
- ▶ The Java Virtual Machine (JVM)
- ▶ The Java standard library

Java is both compiled and interpreted:

- ▶ Java source files (ending in `.java`) are compiled to java bytecode files (ending in `.class`)
- ▶ Java bytecode is then interpreted (run) by the JVM
- ▶ Compiling and running can be done on different machines – bytecode is portable (more precisely, the JVM on each platform accepts the same bytecode).

The enormous Java standard library (containing many Classes notably missing from C++) greatly reduces software development effort.

Object-Oriented Programming in Java

Java is an imperative programming language (we'll learn what that means next class) with support for object-oriented programming.

- ▶ All Java code resides within classes
- ▶ Classes define state (member variables) and behavior (methods)
- ▶ Objects are instantiated from ~class~es (we'll see examples soon)
- ▶ An object-oriented program models some system as a collection of objects
- ▶ Each object communicates with other objects by sending them messages, or *invoking* methods on those objects

The Java SDK

Follow the instructions on the [Resources](#) page of the course web site to install the JDK. Installing the JDK on your computer provides you with several command-line tools, the most important of which are:

- ▶ `javac` - the Java compiler, which compiles `.java` files to `.class` files. You can tell you have correctly installed your SDK like this:

```
$ javac -version
javac 1.8.0_111
```

- ▶ `java` - the Java runtime program, which runs compiled `.class` files. You can tell you have a correctly installed JRE (Java Runtime Environment) like this:

```
$ java -version
java version "1.8.0_111"
Java(TM) SE Runtime Environment (build 1.8.0_111)
Java HotSpot(TM) 64-Bit Server VM (build 25.11-b03, mixed mode)
```

The JRE is included in the JDK, but they can be installed separately.

The Anatomy of a Java Program

It is customary for a programmer's first program in a new language to be "Hello, World." Here's our [HelloWorld.java](#) program:

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

- ▶ The first line declares our HelloWorld class. `class` is the syntax for declaring a class, and prepending with the `public` modifier means the class will be visible outside HelloWorld's package. For now just think of them as boilerplate.
- ▶ Because we didn't declare a package explicitly, HelloWorld is in the *default* package. More on that in a few lectures.
- ▶ The code between the curly braces, `{ ... }` define the contents of the HelloWorld class, in this case a single method, `main`

public static void main(String[] args)

In order to make a class executable with the java command, it must have a main method:

```
public static void main(String[] args) { ... }
```

- ▶ The `public` modifier means we can call this method from outside the class.
- ▶ The `static` modifier means the method can be called without instantiating an object of the class. Static methods (and variables) are sometimes called *class* methods.
- ▶ `void` is the return type. In particular, `main` returns nothing. Sometimes such subprograms are called *procedures* and distinguished from *functions*, which return values.
- ▶ After the method name, `main`, comes the parameter list. `main` takes a single parameter of type `String[]` - an array of `String`'s. `args` is the name of the parameter, which we can refer to within the body of `main`

Compiling Java Programs

Compile Java programs with `javac`, which stands for “Java compiler”

```
$ javac HelloWorld.java
```

With no command line options, `javac` will look in the present working directory (`pwd`) for any `.java` files you pass to `javac` and produce corresponding `.class` files. After compiling `HelloWorld.java` you should have a `HelloWorld.class` in the same directory.

```
$ ls  
HelloWorld.class HelloWorld.java
```

Running Java Programs

Run Java programs with java

```
$ java HelloWorld  
Hello, world!
```

- ▶ The HelloWorld argument tells the java command to find the .class file named HelloWorld (which could be a file or in a JAR archive) and execute its main method.

This is all you need to know for now.