Java Programming
Week 1

Alice E. Fischer

January 22 and 29, 2015

Java is...

The Java Development Kit
Byte Code in the Sandbox
The Garbage Collector
Java is Object Oriented

Two Kinds of Programs

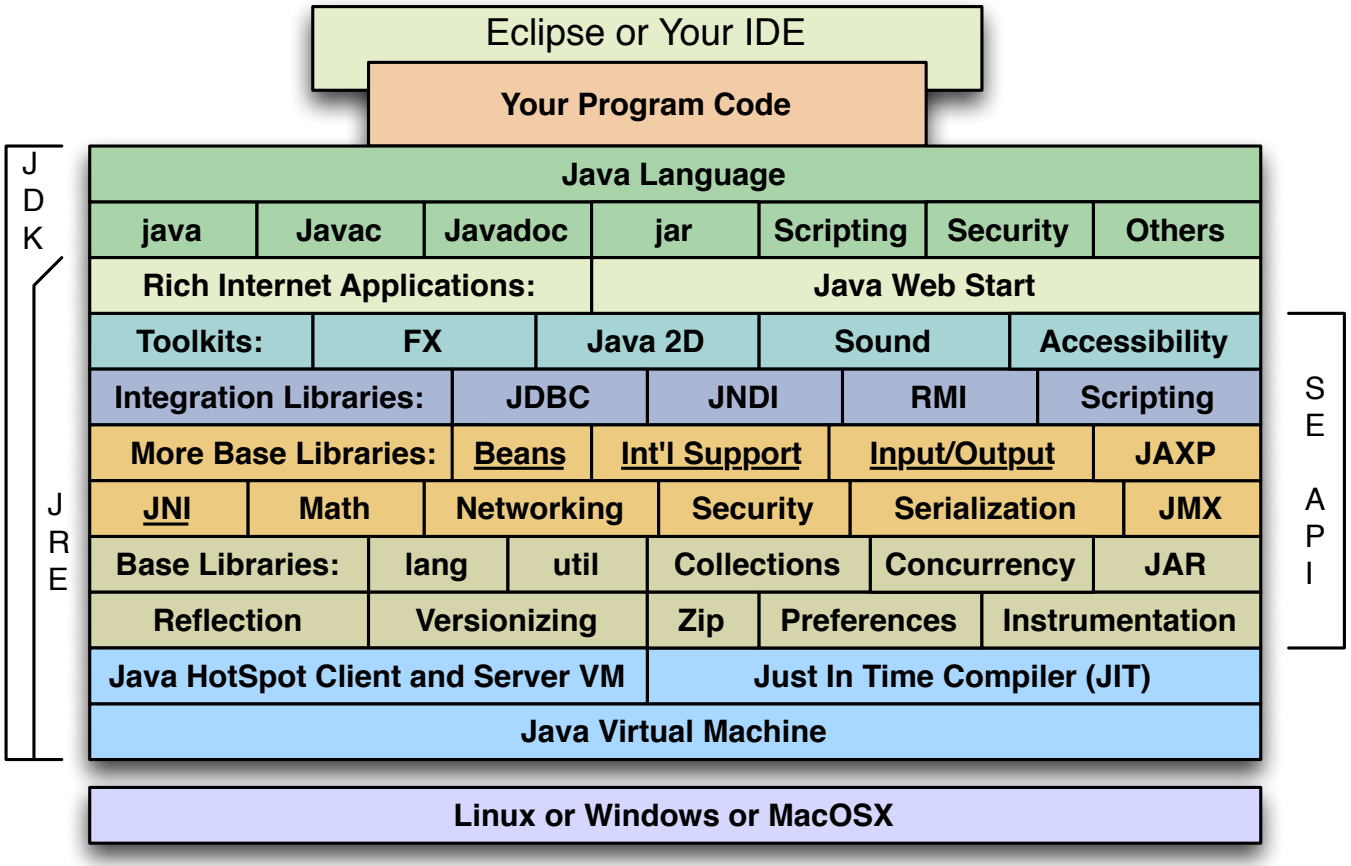
Java is Huge

The parts of Java are:

- A core language that is a lot like C.
- Many many libraries.
- Toolkits for building windowing and network applications.
- A compiler to go from source code to byte code.
- A virtual machine that executes Java byte code.

The Java Development Kit

includes a byte-code compiler and loader, toolkits, libraries, and a byte-code interpreter. The IDE is an independent application.



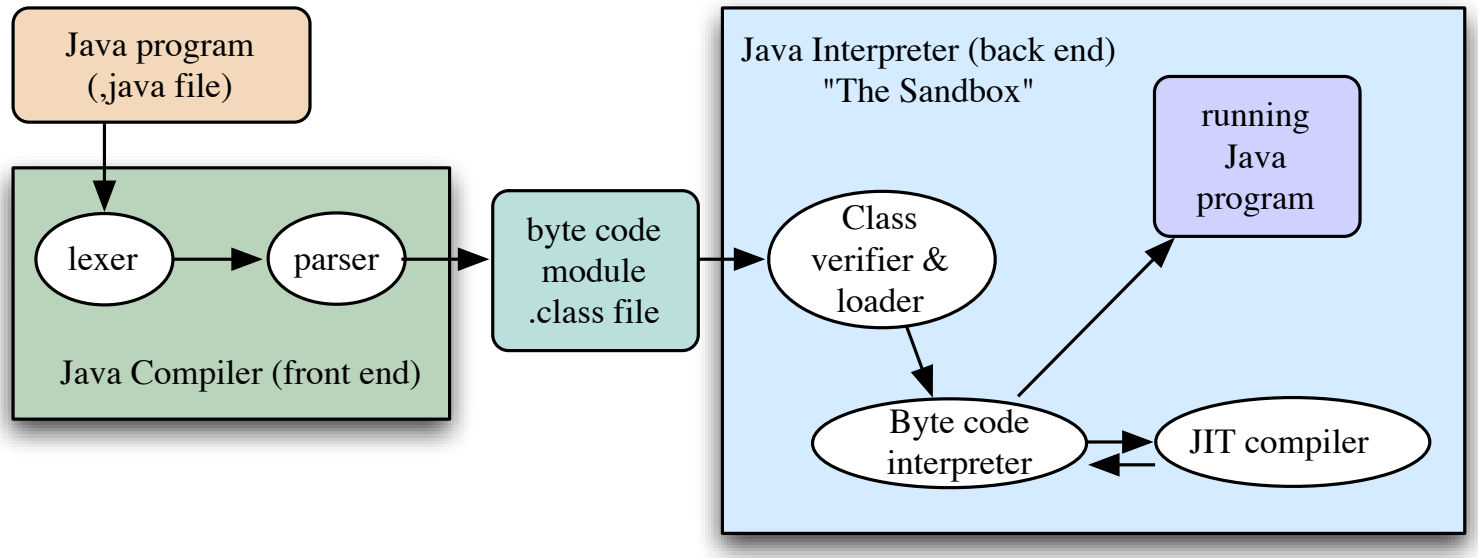
Java systems run on top of many operating systems.

Byte Code in the Sandbox

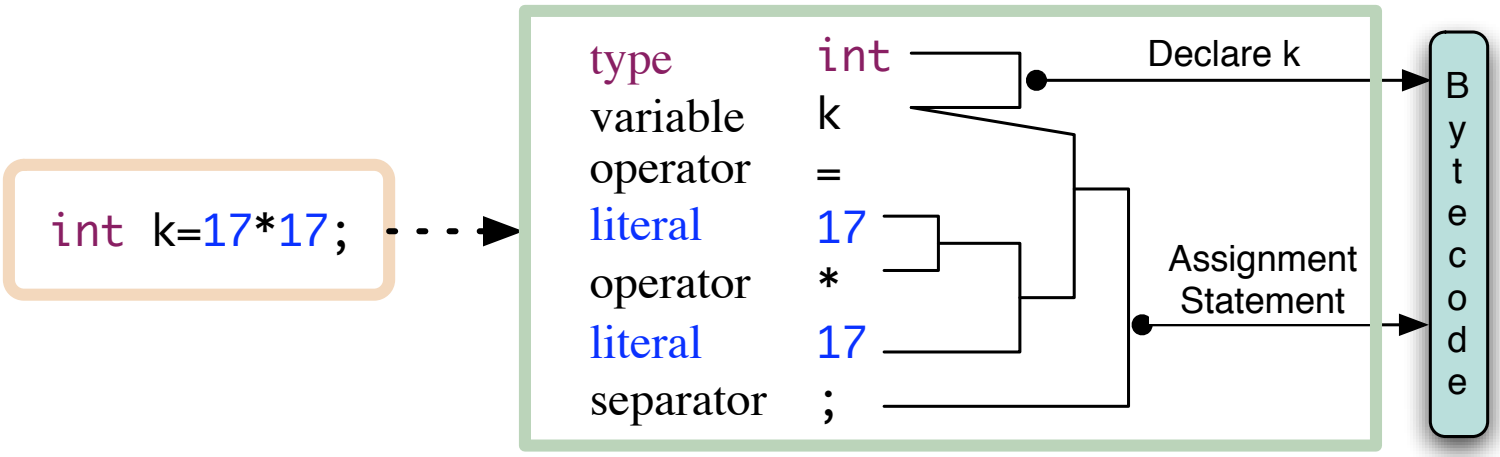
Java uses byte-code to achieve portability (system independence).

- A .java file is translated into a .class file.
- All the .class files for an application are combined into a .jar file or a .war file
- The .jar file can be executed on any machine that has a Java Virtual Machine (JVM)
- The JVM interprets the byte codes by executing the appropriate machine instructions for the local computer.
- Protections are built into the JVM to prevent code from wiping out the system. We say that java programs execute inside a *sandbox*.

The Java Sandbox



From Source to Bytecode



Safe for Beginners?

Java was designed to be a safe language.

- It is safe to run an Applet imported from elsewhere because the Applet runs in a sandbox.
- Java attempts to find and forbid every kind of error that it is possible to find at compile time.
- There are no explicit pointers. That means you can't make the same kind of pointer errors you make in C.
- However, explicit or not, the pointers are there and sometimes cause problems. Java's favorite run-time error comment is "Null Pointer Error".
- Memory areas that can no longer be used (garbage) are identified and collected automatically. This frees the program from the need to manage his own allocations.

The Garbage Collector

Like C, Java has three kinds of storage:

- auto: stored on the run-time stack, created / discarded when a function is called / returns
- static: created and initialized when the program is loaded. Persists until the program terminates.
- dynamic: allocated using `malloc` or `new`.

C and Java handle auto and static storage the same way. However, when dynamic storage is needed,

- In C, it must be explicitly freed by the programmer.
- In Java, it sits there until the garbage collector picks it up for recycling. The cost, of course, is inefficiency and a loss of control over timing.

Java is Object Oriented

But what does Object Oriented mean?

- It is a way of thinking, not a language or set of languages.
- Data representation is designed first, code second.
- A class is a collection of data fields plus the functions that operate on that data.
- Everything is defined around classes; they contain data and related functions.
- Objects are instances of classes.
- Each class should protect itself and take care of itself.
- Much attention is paid to the ways in which classes interact.

Two Kinds of Java Programs

We will study two kinds of Java programs:

- A console program, sometimes called "tool": WinterWork
  You see the output in the console window.
- Application: Winter.java
  The application creates its own graphics window and uses it.
  When you launch the application, the window appears.

WinterWork

A console program can run from the command line and uses the command window for input and output.

- A console program must define a class.
- Inside that class there must be a main function.
- Instead, an application has a `start` function and it runs inside the Java FX environment.
- At the top of the file, a comment should document the purpose of the class, the author, and the date.

We will write console programs for a few weeks until this class has mastered Java basics.

An FX Application: Winter

- A graphics window lets you create output in color, with various fonts (lines 22–24).
- An application has a `start()` function and can be run from a command shell or an IDE.
- It is derived from the class Application, which supplies a graphics window for your content.
- The `start()` function creates the contents of the window (lines 35–38) and makes it visible.
- The class members (line 18–25) define the parts of the window.

Basics

Javadoc Comments
OO Vocabulary
WinterWork

Input and Output

Miles Per Hour
Console Input
Console Output

Javadoc Comments

In addition to the two kinds of comments C, we have a third:

- A Javadoc comment starts with /** and a brief general description of the purpose of the function. It ends with the next */.
- Inside, you write javadoc annotations such as @author, @since, @param, @return. Each one is followed by relevant information.
- Following that is a detailed description of what the function does.
- When your source code is run through a Javadoc application, it generates html documentation pages from your Javadoc comments. This makes it easy to keep the documentation for your program up to date.

OO Vocabulary

- package: A package is a set of classes that work together and are stored in the same subdirectory. When you use a system-defined package, you usually import its public declarations into your namespace.
- When you create your own package, it must have the same name as the directory in which it is stored.
- class: Everything in Java is inside classes. Each class has a name. A public
  class must be in a file with the same name.
- instantiate: To make a new object, we instantiate a class. The object is called and instance of the class.
- object: An instance of a class.
- primitive: An instance of built-in type such as int or double.

Keywords

- public: a class, function, or data object that may be used by any part of your program.
- static: A static function or variable can be used immediately when your program is loaded. Your main function must be public static.
- import: This command brings into your program the names of all the public members of the selected package. You can use these public members without the "import", but then you must write the entire long name each, including the class and package that it is defined in.

WinterWork

```
/** WinterWork.java:  a first Java program.
    @author Alice Fischer
    @version 1/29/07
 */
public class WinterWork {

   // A console program must have a main function.

   public static void main(String[] args) {
      System.out.println( "\n Winter time, Java.");
      System.out.println( " New, yet not new, much to "
              +"learn.\n Hard work brings rewards.\n ");
   }
}
```

Elements to note in WinterWork.java

- The author's name and other identifying information should be in a Javadoc comment at the top of the file.
- Other comments begin with // and go to the end of the line, or begin with /* and end with */
- `import` commands bring the names defined within a package into your namespace. `java.lang` contains the most basic classes in the Java language. It is imported automatically, so this command may be omitted.
- The most basic parts of Java are like C, including statements, string literals, and newline characters.

`public static void main( String[] args );`

- If something is `static`, it can be used without creating any objects. The Java libraries are full of public static functions that can be called any time, any where.
- The main function must always be `public` and `static` These properties make it possible for main to be started up by the Java system.
- The main function must `void`. Unlike a main program in C, it will NOT return any status code or information of any kind to the Java system when the program ends.
- The Java system CAN send command-line arguments into main, although we will not do so this term. These are sent to main in the form of an array of Strings: `(String[] args )`

Streams

Streams are used for input and output. Three streams are predefined in Java, inside the class System:

- `System.in` is the "standard" input stream. Like `stdin` in C, it is connected to the keyboard. Its type is `InputStream`.
- `System.out` is the "standard" output stream. Like `stdout` in C, it is connected to the screen. Its type is `PrintStream`.
- `System.err` is the "standard" stream for error comments, connected to the screen. Like `stderr` in C, it is connected to the screen. Its type is `PrintStream`. Output sent to System.err is mixed in with output sent to System.out.

These streams are all static objects. They are opened for you and are ready to use when your program starts up.

The code inside main()

- `println()` is an output function that can send output to any `PrintStream`, then print a newline.
- `print()` is like `println()` except that it does not add a final newline character.
- The argument to `println()` or `print()` should be a string or something that Java knows how to convert to a string.
- If you want to output a long string, break it onto two lines, as shown. The + on line 21 is the *concatenate* operator. It tells Java that the second part of the string should be glued to the end of the first one, not treated as a separate object.
- A `return` statement is not necessary because main() is a void function.

Miles Per Hour: Scanner

Use the Scanner class, from the `java.util` package, to do input.

- The stream Standard.in is opened automatically when you run a program.
- Instantiate a new Scanner to work on the stream Standard.in.
- Use your scanner to call one of the `next` functions: `nextDouble()`, `nextInt()`, `nextLong()`, `nextBoolean()`, `nextLine()`, and many others.
- `next()` reads the next whitespace-separated string.

Use the Java API documentation to learn details and learn about more possibilities.

Miles Per Hour: Console Output

String output is easy in Java.

- The stream Standard.out is opened automatically when you run a program.
- Use `printf()` to print a formatted number.
- Use `print()` to print a string and keep the cursor on the same line.
- Use `println()` to print a string ending in a newline.
- If your string has more then one part, write a + between every pair of parts.
- If you try to print a non-string: `System.out.print( 35 );` Java will automatically call the `toString()` method.