CHAPTER 1

INTRODUCTION

Source of Slides: Donald W. Smith
TechNeTrain.com

Java History

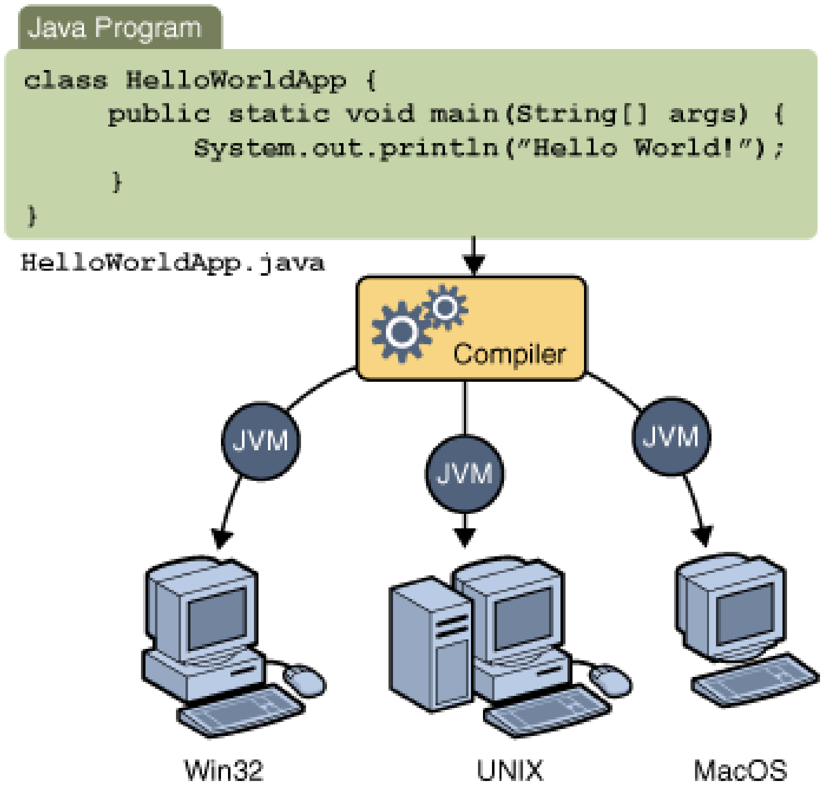
- Java Design Goals
  - Safe: Can run inside browser and will not attack your computer
  - Portable: Runs on many Operating Systems
    - Windows
    - Mac OS
- Java programs are distributed as instructions for a 'Virtual Machine,' (JVM) making them platform-independent
  - Virtual machines are available for most Operating Systems.

Interesting Recent Discussion of OO Programming

- https://medium.com/@cscalfani/goodbye-object-oriented-programming-a59cda4c0e53

Java Virtual Machines

- Source code
  - e.g. HelloWorldApp.java
- Portable 'byte code'
  - The compiler (**javac**) generates *byte code* in a '***.class***' file which can be run on any Java Virtual Machine
    - e.g. HelloWorldApp.class
  - JVM efficiently interprets* byte code in the .class file into native machine code (binary) and executes it
    - *can also compile intonative machine code



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

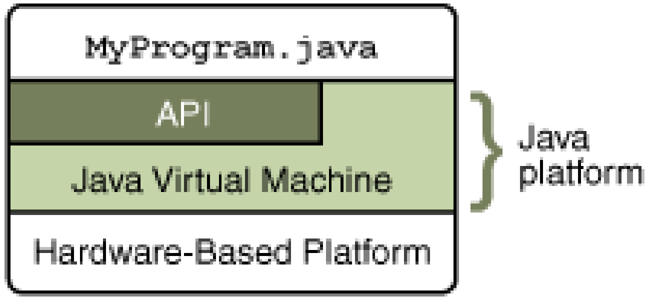
The Java API

- The Java Platform consists of two parts:

  1) Java Virtual Machine

  2) Java API

  -- also called libraries



- The Application Programming Interface (API) is a huge collection of handy software packages that programmers can use:
  - Graphics, user interface, networking, sound, database, math, and many more

The Java SDK

- Install Java SDK (**S**oftware **D**evelopment **K**it)
  - I am using version 14.0.2
  - Google 'Java SDK download,' Get SE (Standard Edition) version
  - Location after installed on Windows will be:
    - `C:\Program Files\Java\jdk-14.0.2`
      - last few numbers may vary with releases

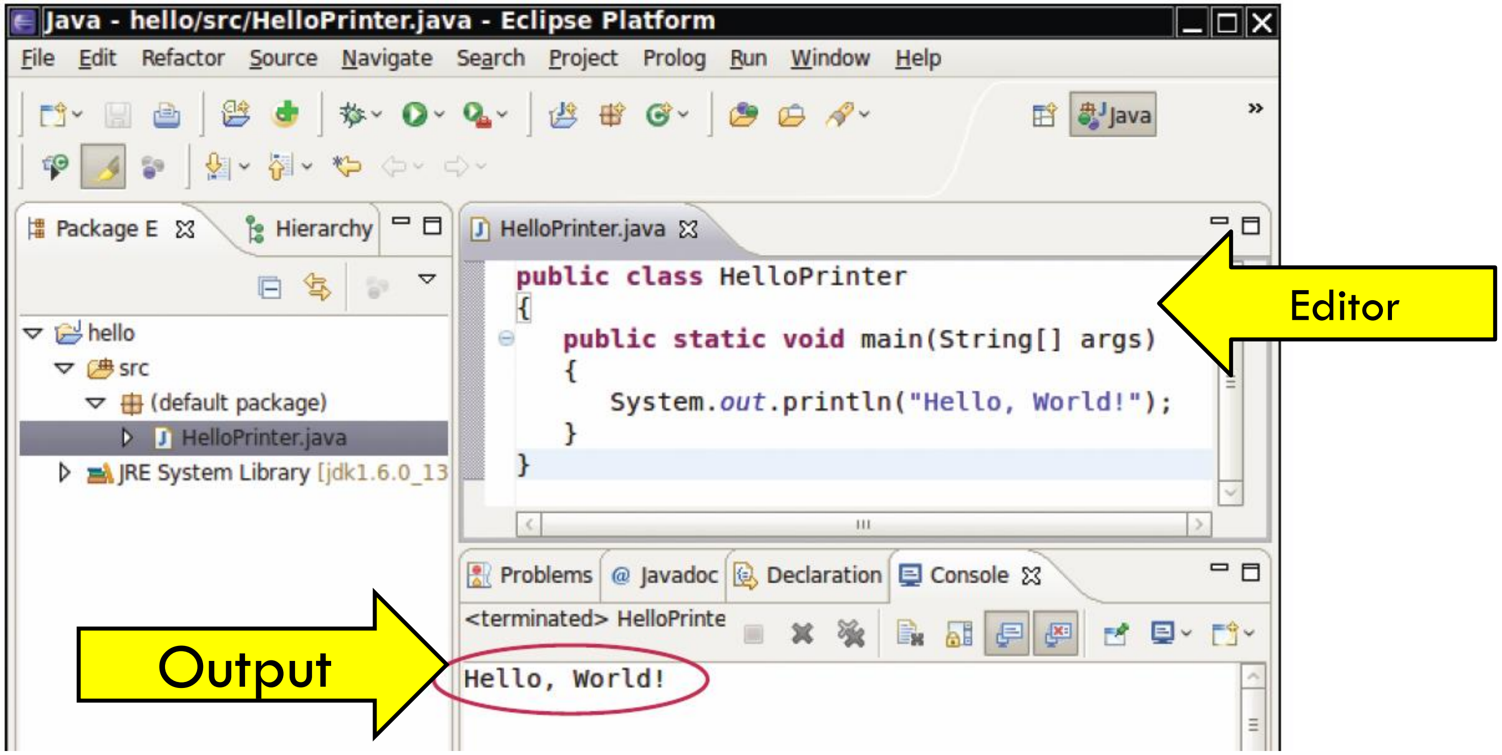
- The SDK includes programs such as:
  - `java.exe` (Executes Java applications via JVM)
  - `javac.exe` (Java compiler)
  - `javadoc.exe` (Javadoc generator)

Programming Environments

- There are many free programming tools available for Java
  - I am using ***Eclipse*** - Eclipse is an **IDE**
  - You can also use an editor (e.g. emacs) but they are not as powerful

- Integrated Development Environment (IDE) components:
  - Source code editor helps programming by:
    - Listing line numbers of code
    - Color lines of code (comments, text…)
    - Auto-indent source code
  - Output window
  - Debugger

An Example IDE


Java - hello/src/HelloPrinter.java - Eclipse Platform
File Edit Refactor Source Navigate Search Project Prolog Run Window Help
Java
Package E
Hierarchy
hello
src
(default package)
HelloPrinter.java
JRE System Library [jdk1.6.0_13
HelloPrinter.java
Problems
@ Javadoc
Declaration
Console
<terminated> HelloPrinte
Editor
Output

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

```
Hello, World!
```

- Many IDEs are designed specifically for Java programming

Editors and IDEs on Lab Computers

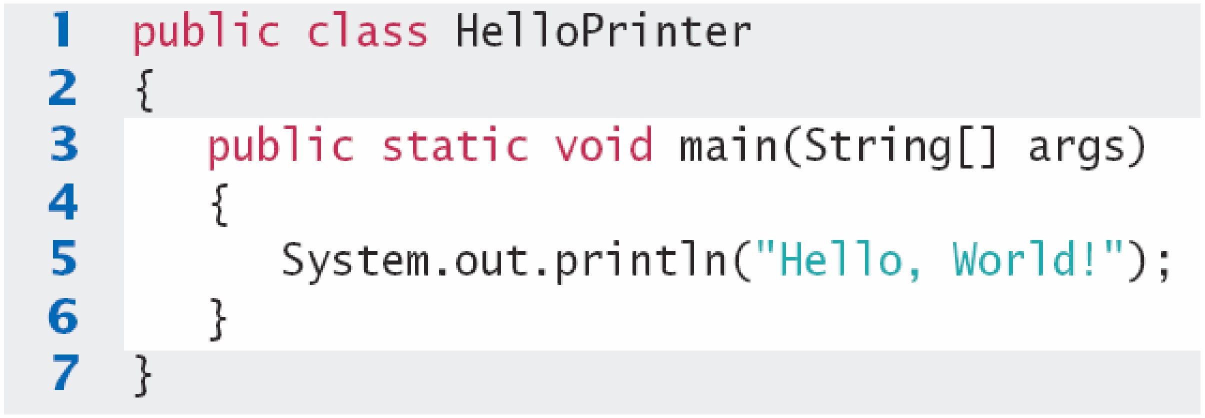
- gedit (editor)
- emacs (editor)
- Eclipse (IDE)
- BlueJ (IDE)
- Visual Studio Code (IDE)

Other Good IDEs

- IntelliJ
- Netbeans
- jGrasp

Your First Java Program

- Traditional ‘Hello World’ program in Java

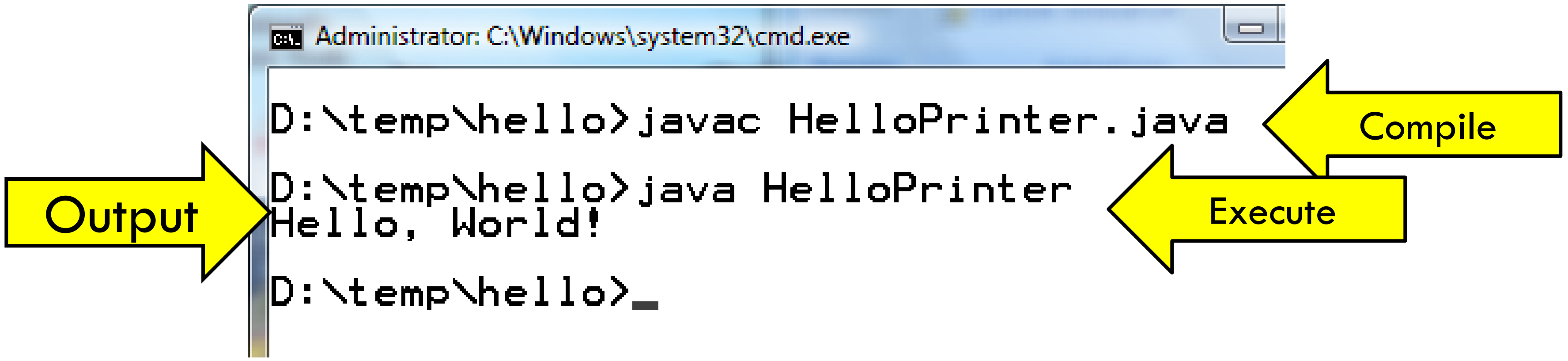


```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello, World!");
   }
}
```

- We will examine this program in the next section
  - JaVa iS CaSe SeNsItiVe
  - Java uses special characters, e.g. { } ( ) ;

Text Editor Programming

- Compile the program with javac in terminal window
- Run the program with java

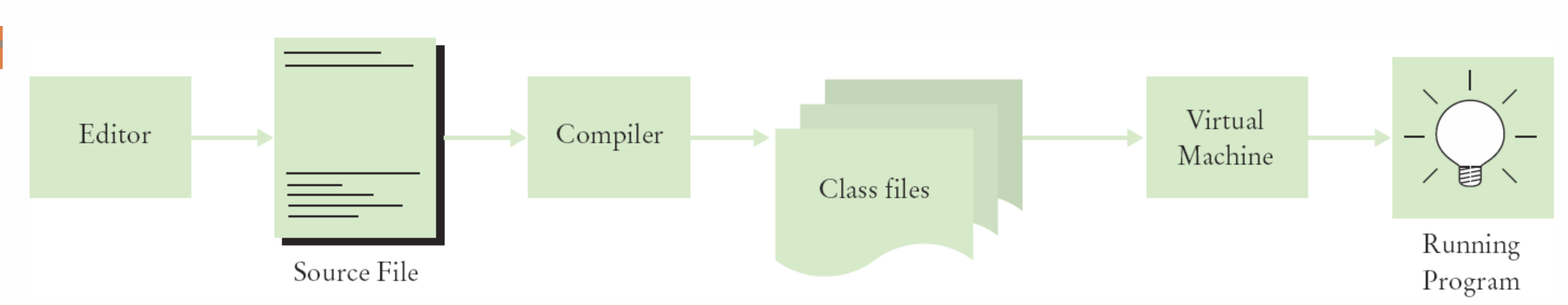


```
D:\temp\hello>javac HelloPrinter.java

D:\temp\hello>java HelloPrinter
Hello, World!

D:\temp\hello>_
```

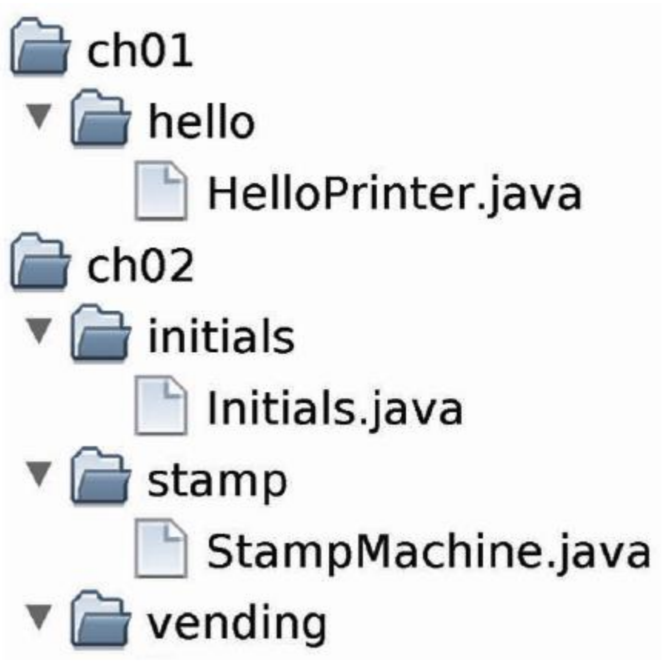
Source Code to Running Program



- The compiler generates the `.class` file which contains instructions for the Java Virtual Machine
- `.class` files contain ‘byte code’ that you cannot edit
  - `D:\temp\hello>Type HelloPrinter.class`
  - `╨■║╛    2 ↔ ♠ ☼     ► ↕ ‼ ¶ § ▬☺ ♠<init>☺ ♥()V☺ ♦Code☺ ☼LineNumberTable☺ ♦main▬([Ljava/lang/String;)V☺`
  - `Hello, World! elloPrinter.java♀ ↨♀ ↑ ↓☺`

Organize your work

- Your 'source code' is stored in `.java` files
- Create one folder per program
  - Can be many `.java` files
- Be sure you know where your IDE stores your files!



```
ch01
  ▼ hello
      HelloPrinter.java
ch02
  ▼ initials
      Initials.java
  ▼ stamp
      StampMachine.java
  ▼ vending
```

Your First Program

```
1  public class HelloPrinter
2  {
3      public static void main(String[] args)
4      {
5          System.out.println("Hello, World!");
6      }
7  }
```

Line 1: Declares a '`class`' `HelloPrinter`

-- Every Java program has one or more classes.

Line 3: Declares a method called '`main`'

-- Every Java application has exactly one '`main`' method

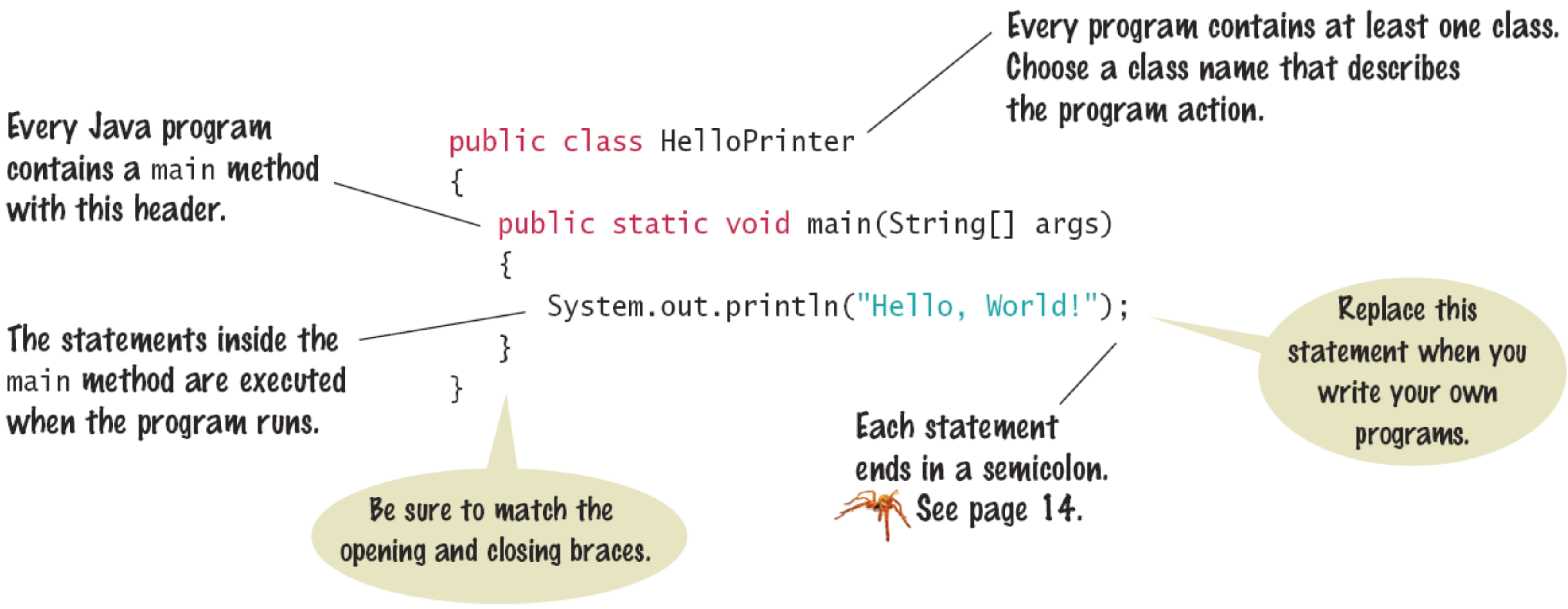
-- Entry point where the program starts

Line 5: Method `System.out.println` outputs 'Hello, World!'

-- A statement must end with a semicolon (`;`)

Syntax: The Java Program

- Every application has the same basic layout
  - Add your ‘code’ inside the `main` method



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

Every program contains at least one class. Choose a class name that describes the program action.
Every Java program contains a main method with this header.
The statements inside the main method are executed when the program runs.
Be sure to match the opening and closing braces.
Each statement ends in a semicolon. See page 14.
Replace this statement when you write your own programs.

Calling Java Library methods

```
5        System.out.println("Hello, World!");
```

- Line 5 shows how to 'call' a 'method' from the Java API: `System.out.println`
  - Notice the dots (periods)
  - Parenthesis surround the arguments that you 'pass' to a method
  - We are passing a String "Hello World"
    - Note the double quotes which denote a String inside
  - You can also print numerical values
    - `System.out.println(3 + 4);`

Getting to know `println`

- The `println` method prints a string or a number and then starts a new line.

```
System.out.println("Hello");
System.out.println("World!”);
```

```
Hello
World!
```

- The `println` method has a ‘cousin’ method named `print` that does not print a new line.

```
System.out.print("00");
System.out.println(3+4);
```

```
007
```

A method is called by specifying the method and its agruments

Common Error

- Omitting Semicolons!!
  - In Java, every statement **must end in a semicolon.**
  - For example, the compiler sees this:

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

  - As this:

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

  - It doesn't understand this statement, because it does not expect the word `System` following the closing parenthesis after `Hello`.