Objectives

- Introduction to Java
- Basics of Java Syntax
- Java fundamentals
  - Print statements

Office Hours

- Zoom – Tuesday 9:30-11:30

Review: Version Control

- What are the features/functionality/benefits of version control?
- What is git? Vs what is GitHub?
- What are some of the common Git commands and what do they do?
  - Put them together: what is a typical work flow for using git?

You can and *should* review slides if you don't remember answers

INTRODUCTION TO JAVA

What is Java?

… and, why should I learn it?

- From Sun Microsystems
  - 1995, James Gosling and Patrick Naughton
  - Specifications

- Object-oriented
- Rich and **large** library
- Develop cross-platform applications
  - Web, desktop, embedded
- Widely used
  - Frameworks to enable easier development

Feedback from Alumnus

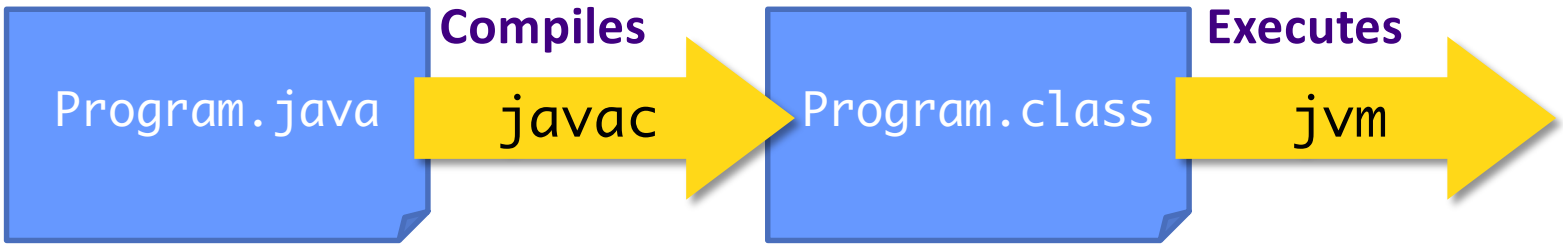
Also thank you for the Software Engineering course in Java! My team codes in C++ and java knowledge have been very helpful in ramping up 🙂

Goal: Transferrable skills

What is Java?

- Java Programming Language
- Java Virtual Machine
- Java Class Libraries

Overview:
Compiling, Executing Java Programs

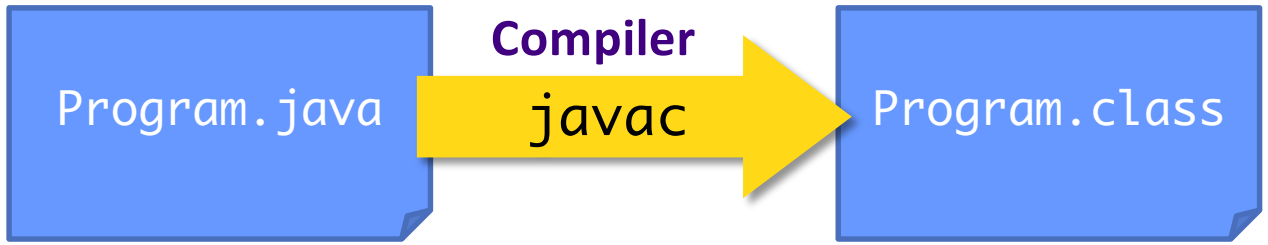


Written in Java Programming Language

Bytecode: machine code for a virtual CPU

Compiling Java Programs

Step 1:



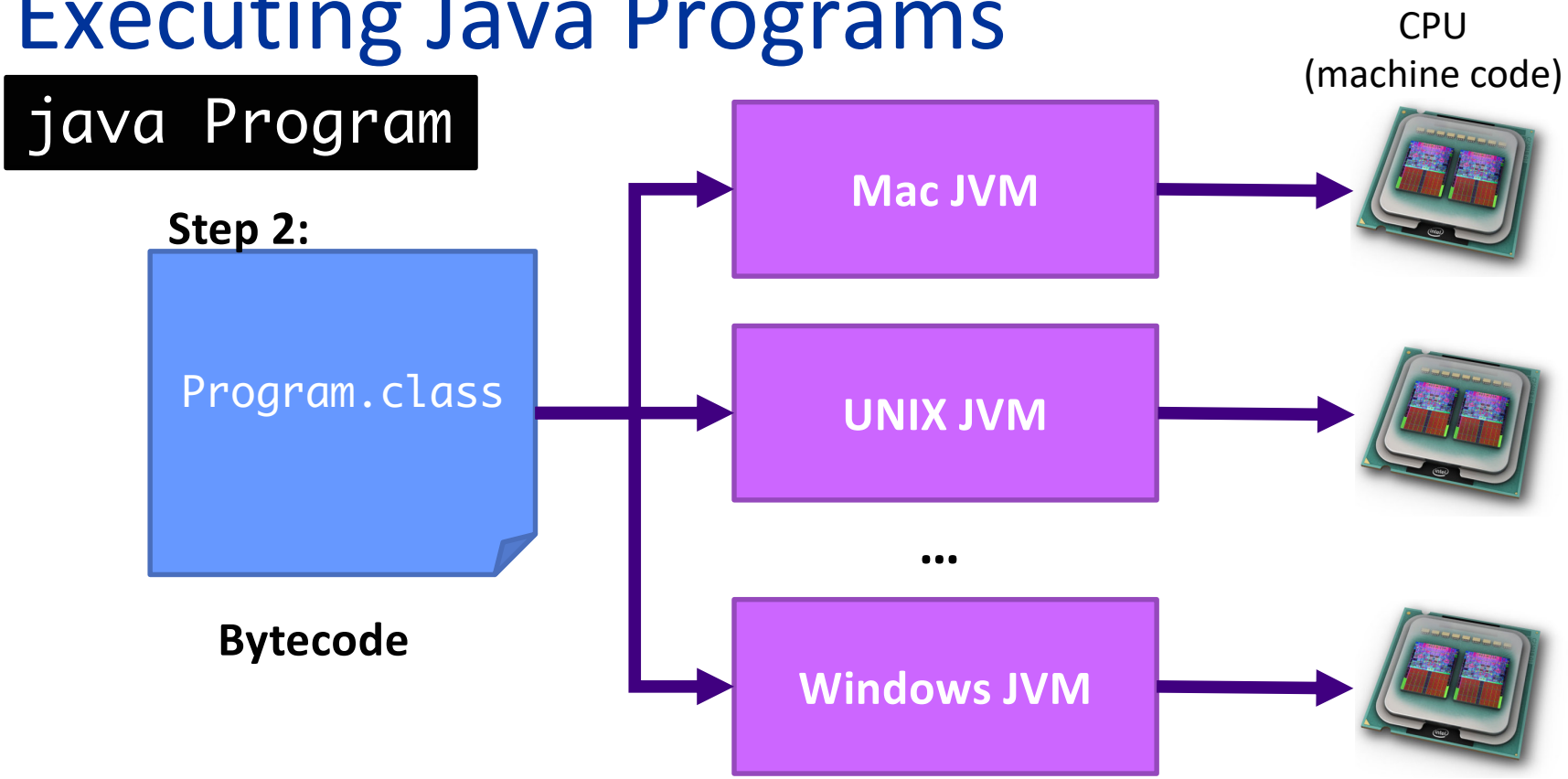
Written in Java Programming Language

Bytecode: machine code for a *virtual* CPU

```
javac Program.java
```

- Compiler catches some errors
- Need to fix those errors and recompile

Executing Java Programs

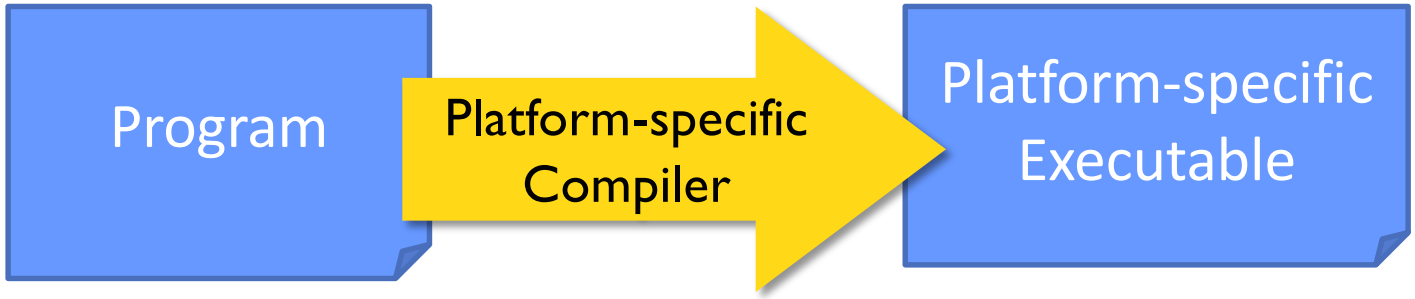


- Same **bytecode** is executed on each platform
- Don't need to provide the source code

Java Virtual Machine (JVM)

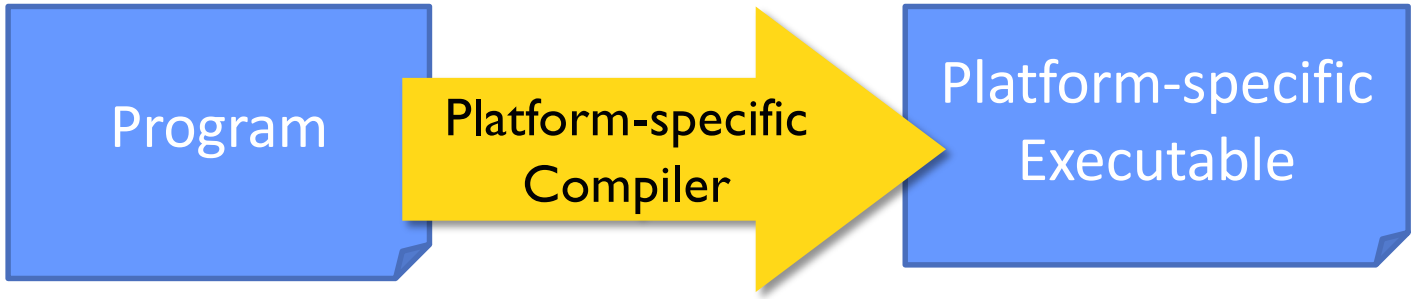
- Emulates the CPU
  - Usually specified in software (rather than hardware)
- Executes the program's ***bytecode***
  - Bytecode: virtual machine code
- JVMs available for each Java-supported platform
  - Enables program *portability*

Traditional (C/C++) Program Execution



- Example: I use my Mac-specific compiler to compile program into a Mac-specific executable
- Limitation: Executable is not portable

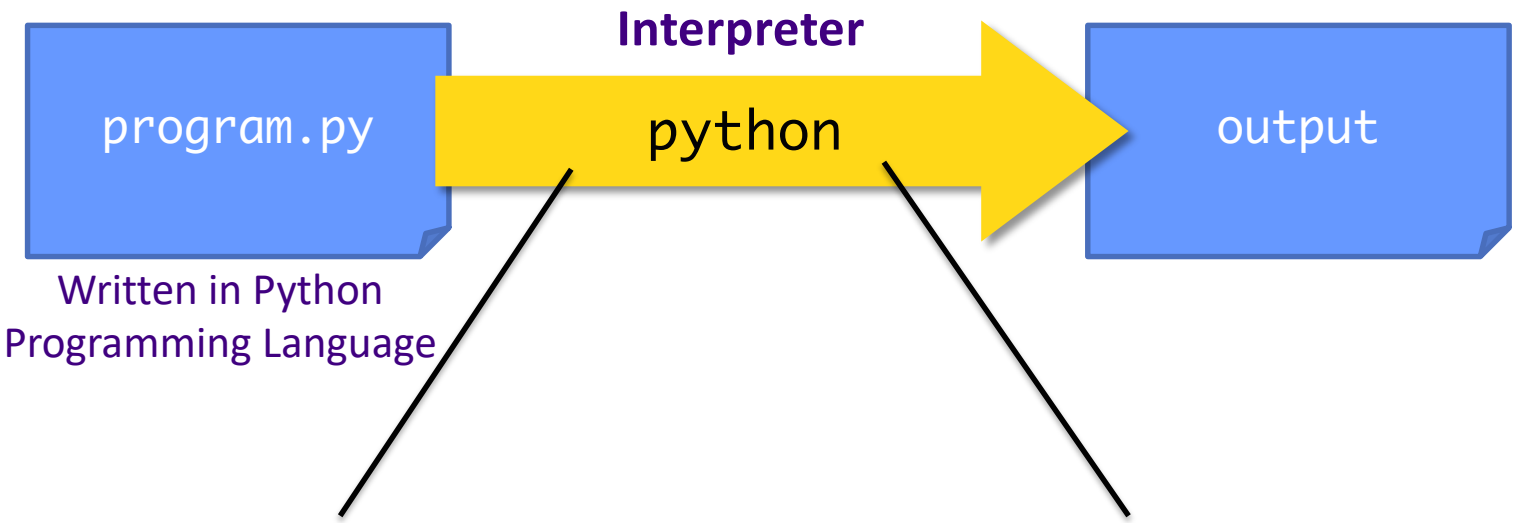
Traditional (C/C++) Program Execution



- Example: I use my Mac-specific compiler to compile program into a Mac-specific executable
- Limitation: Executable is not portable to any OS

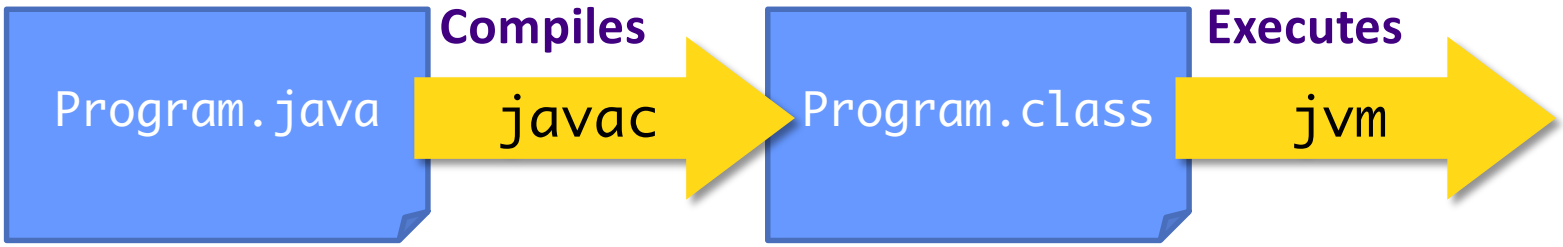
What is Python's approach?
How are (I) Java and (II) the traditional approach the same and different from Python's approach?

Executing Python Programs



1. Syntax validation
   - exit if not valid
2. Translate Python code to Python bytecode
   - Bytecode stored in __pycache__ directory
3. Python virtual machine execute Python bytecode

Overview:
Compiling, Executing Java Programs

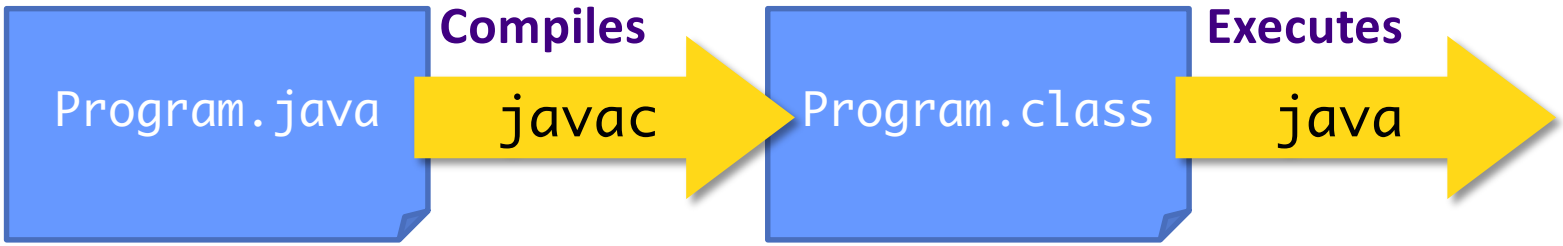


Written in Java Programming Language

Bytecode: machine code for a virtual CPU

JDK: Java Development Kit

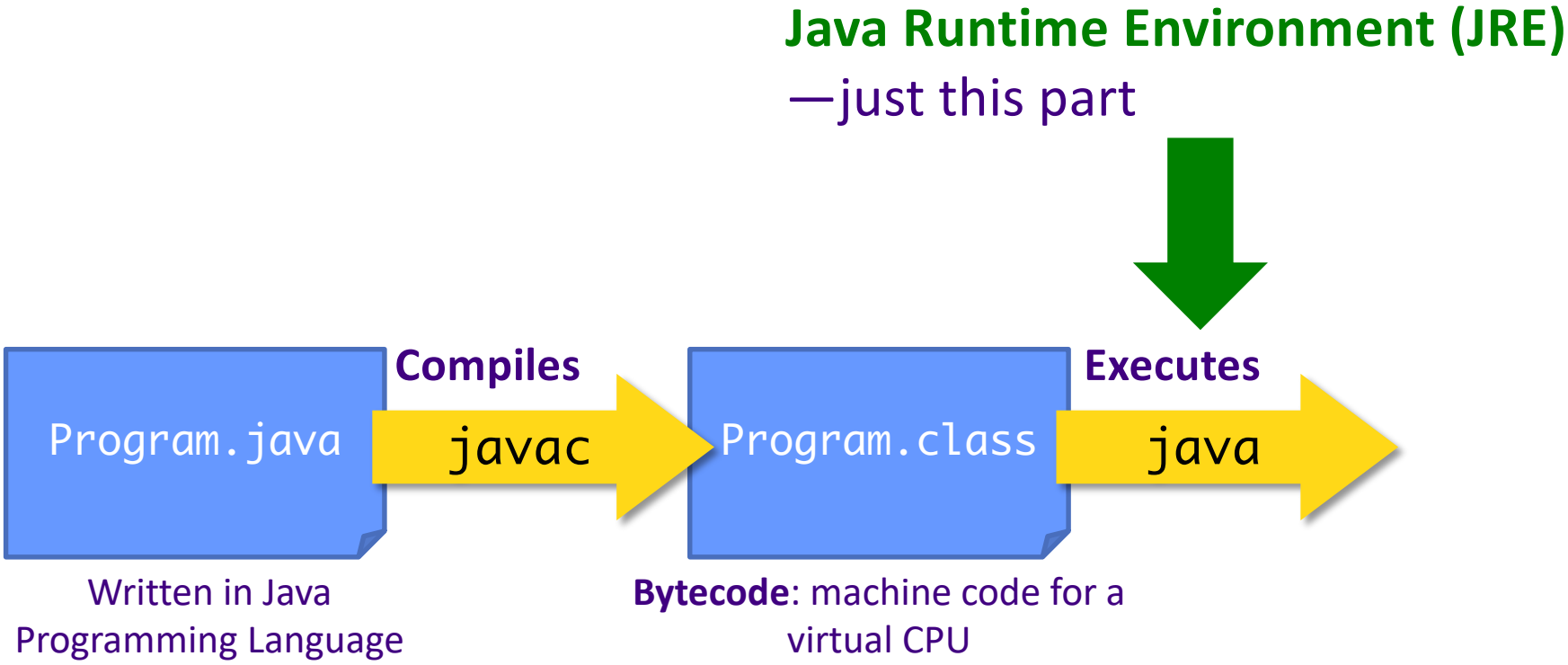
- Contains
  - `javac`: Java compiler
  - `java`: Java Virtual Machine
  - Java class libraries



Written in Java Programming Language

Bytecode: machine code for a virtual CPU

JRE: Java Runtime Environment



Java Class Libraries

- Pre-defined classes
  - Included with Java Development Kit (JDK) and Java Runtime Environment (JRE)
  - View the available classes online:

https://docs.oracle.com/en/java/javase/21/docs/api/index.html

- Similar in purpose to *modules* available for Python

What is Java?

- Java Programming Language
- Java Class Libraries

} What this course is about

- Java Virtual Machine
  - Use the JVM but won't learn about how it works
  - For more information on JVM:
    https://docs.oracle.com/javase/specs/

Bringing It Together: Benefits of Java

- Rapid development of programs
  - Large library of classes, including GUIs, Enterprise-level applications, Web applications
- Portability
  - Run program on multiple platforms without recompiling
- Compiled
  - Find some errors before execution!
    - Statically typed
  - Can give performance boost by doing optimizations

LET'S PROGRAM!

Example Java Program: Hello.java

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

What are your observations about this program?
What can you figure out?

Example Java Program

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

- Everything in Java is inside a **`class`**
  - Java is *entirely* object-oriented*
  - This class is named **`Hello`**

Java: Files and Class Definitions

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

Defines **the class** `Hello`

`Hello.java`

- Name of the file **must** match the name of the class
  - E.g., `Hello.java`

- In general, each Java program file contains **one** class definition

Java: Blocks of Code

Blocks of code marked with { }

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

Defines **the class** `Hello`

`Hello.java`

Java: Access Modifiers

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

Access Modifier:

controls if other classes can use code in this class

Java: Method Definitions

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

method

- This class contains one *method* definition: `main`

The `main` Method

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

- Similar to `main` in Python
  - But *must be* associated with a *class*
- Must take one parameter: an *array* of Strings
  - For command-line arguments
- Must be `public static`
- Must be `void`: data type of what method returns (nothing)
- `main` is *automatically* called when program is executed

Example Java Program

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

- Method contains one line of code
  - What do you think it does?

Java: Print Statements

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

- Calls the **`println`** method on the **`System.out`** object
- **`println`** takes one parameter, a `String`
- Displays string on terminal, terminates the line with new line (`\n`) character

Print Statement

- Syntax:

```
System.out.println(<String>);
System.out.print(<String>);
```

No newline at end

- Closer to how you use Python's `file.write()` method
  - Need to combine parameter into *one* `String` using `+`'s
    - Recall: Python's `print` used *commas*
  - More on `String` operations later

Java: Comments

```
/**
 * Our first Java class: displays Hello!
 * @author Sara Sprenkle
 */
public class Hello {
   public static void main(String[] args) {
      //print a message
      System.out.println("Hello!");
   }
}
```

- Comments: /* */ or //
  - /** */ are special JavaDoc comments

Java Code Style

- **Comments** describing class
  - Sprenkle CSCI209 requirements:
    - **Must** include high-level description of program
    - **Must** include your name as author

```
/**
 * Displays "Hello!"
 * @author Sara Sprenkle
 */
```

Java Code Style

- **Comments** describing class
  - Sprenkle CSCI209 requirements:
    - **Must** include high-level description
    - **Must** include your name as author

```
/**
 * Displays "Hello!"
 * @author Sara Sprenkle
 */
```

Tags must be *last* in Javadoc

Java Code Style

- **Comments** describing class
  - Sprenkle CSCI209 requirements:
    - **Must** include high-level description of program
    - **Must** include your name as author

```
/**
 * Displays "Hello!"
 * @author Sara Sprenkle
 */
```

Tags must be *last* in Javadoc

- Proper **indentation**
  - Similar to Python
  - Everything within pairs of {} is indented the same
  - Not required by compiler but for readability

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

A Note About Examples' Comments

- The example code that I provide is often "over" commented
- I'm providing information for you that isn't needed in your submissions
  - However, if it's helpful for you, you can keep "over"commenting

Translate to Python Program?

```
/**
 * Our first Java class
 * @author Sara Sprenkle
 */
public class Hello {
   public static void main(String[] args) {
      //print a message
      System.out.println("Hello");
   }
}
```

Translation to Python Program

```
print("Hello")
```

Literal translation:

```
class Hello:
   """Our first Python class"""

   @staticmethod
   def main():
        print("Hello")
```

Compare Python and Java

```
# a Python program
def main():
      print("Hello")

main()
```

```
/**
 * Our first Java class
 * @author Sara Sprenkle
 */
public class Hello {
   public static void main(String[] args) {
      //print a message
      System.out.println("Hello");
   }
}
```

Java vs. Python, so far...

- ***Semantics*** the same, ***syntax*** different
  - Blocks of code
  - End statements
- Access modifiers
- Data type declarations
- Class-based programs
- Compiled

We'll see more differences as we go…

New in Java 21

- A simpler option for main

```
public class Hello {
    void main() {
        System.out.println("Hello!");
    }
}
```

- But feature is only in *preview*
  - to allow for developer feedback based on real-world uses before becoming permanent in a future release (25!)

Looking Ahead

- Git Lab Wednesday before class
- Read textbook: up to 2.2: Lets look at a Java Program
- Complete Java compilation and execution before next class