Objectives

- Java Fundamentals
  - Print statements
  - Data types, variables
  - Arithmetic operators
  - Development process

Review

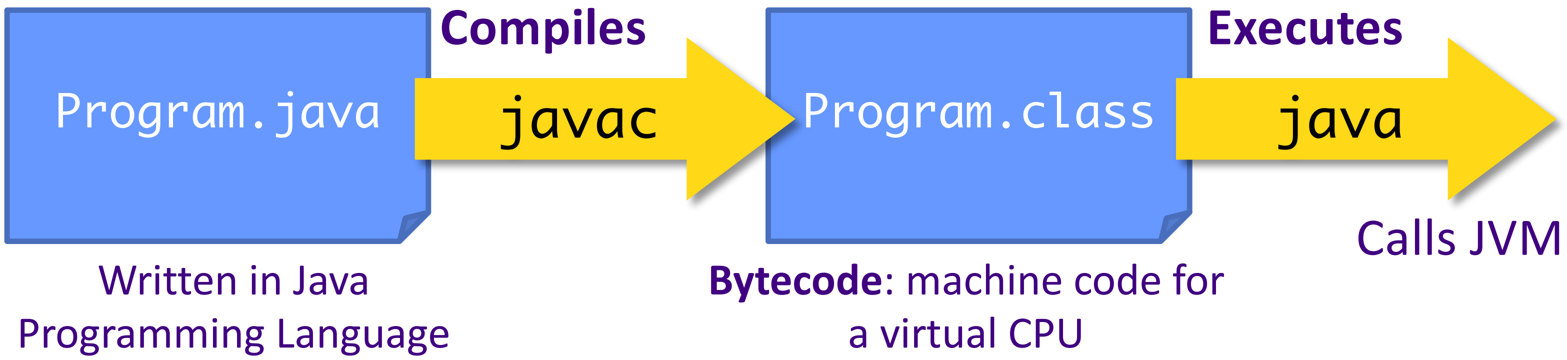
- What are the benefits of Java?
- How do you compile and run Java programs?
- How do you display output in Java?
- What are the modifiers for the `main` method?
  - What are the parameter(s) to `main`?
  - How do you *call* the `main` method?
- How does Java compare to Python (so far)?

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

Review: Benefits of Java

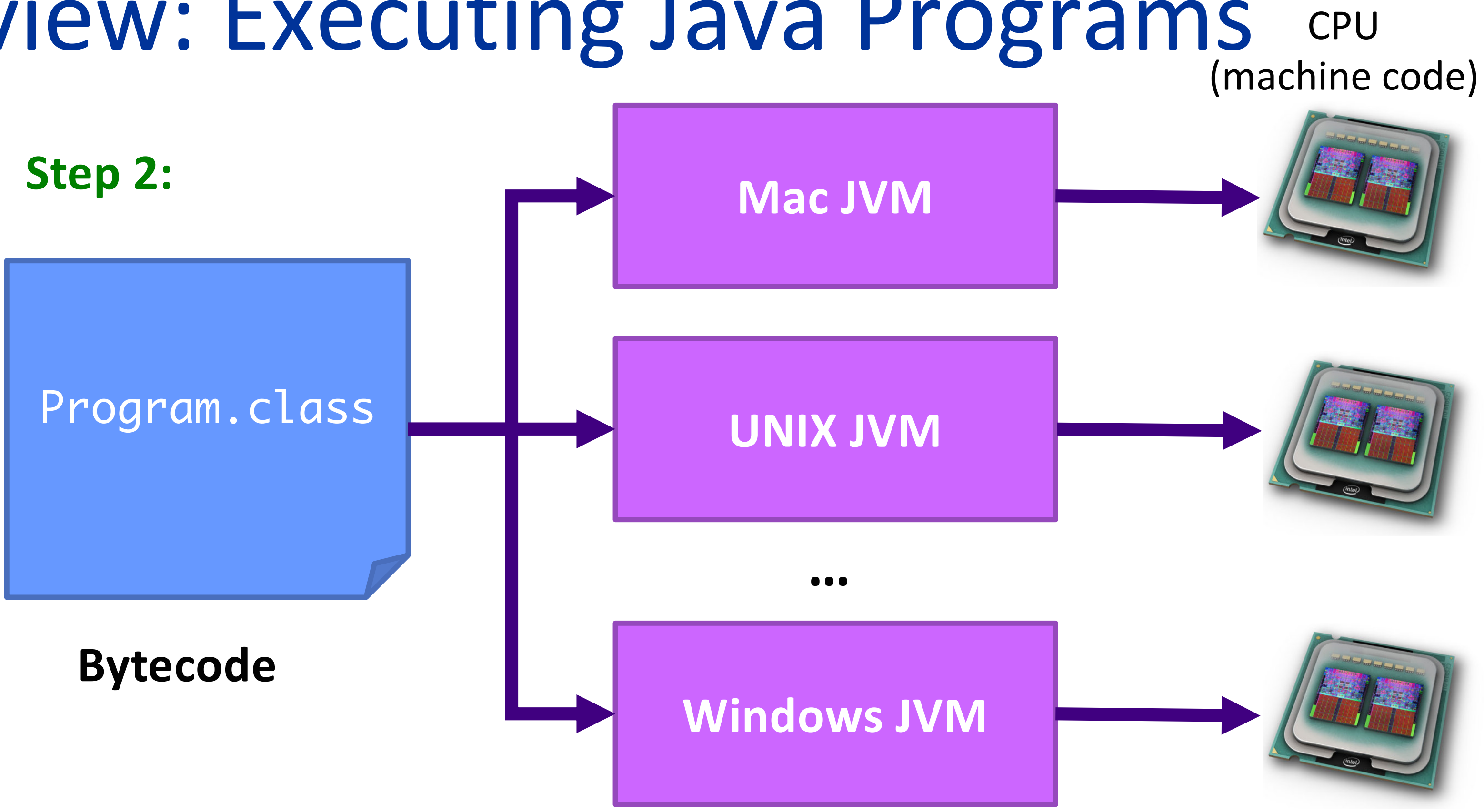
- Rapid development of programs
  - Large library of classes, including GUIs, Enterprise-level applications, Web applications
- Portability
  - Run program on multiple platforms without recompiling
- Compiled
  - Find some errors before execution!
    - Statically typed
  - Can give performance boost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

Review: Compiling, Executing Java Programs


Program.java
Compiles
javac
Program.class
Executes
java
Calls JVM
Written in Java Programming Language
Bytecode: machine code for a virtual CPU

```
javac Program.java
java Program
```

Review: Executing Java Programs



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

Review: Example Java Program

```
/**
 * 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!");
   }
}
```

`main` method is automatically called when you run

```
java Hello
```

Aside: JavaScript vs Java

- JavaScript is **not** Java
  - JavaScript is a *scripting* language, primarily embedded in HTML, executed by Web browsers*

```
<script type="text/javascript">
function myFunction() {
   return ("Hello, have a nice day!")
}
</script>
</head>
<body>
<script type="text/javascript">
   document.write(myFunction())
</script>
```

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

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

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

- **Comments** describing class
  - Sprenkle CSCI209 requirements:
    - **Must** include high-level description of program
    - **Must** include your name as author
- 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!");
    }
}
```

Java Code Style

- **Comments** describing class
  - Sprenkle CSCI209 requirements:
    - **Must** include high-level description of program
    - **Must** include your name as author
- Proper **indentation**
  - Similar to Python
  - Everything within pairs of {} is indented the same
  - Not required by compiler but for readability

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

Tags must be *last* in Javadoc

```
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

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

main()
```

```java
/**
 * 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…

JAVA FUNDAMENTALS

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

String Concatenation

- If a string is concatenated with something that is not a string, the other thing is converted to a string automatically.

Note the +

```
System.out.println("The answer is " + 42);
```

Automatically converted to a String

WinPercentage.java

Java keywords/reserved words

- Case-sensitive
- Can’t be used for variable or class names
- Reserved words seen so far …
  - `public`
  - `class`
  - `static`
  - `void`
- Exhaustive list
  - http://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

Data Types

- Java is ***strongly*** and ***statically typed***
  - Every variable must have a ***declared* type**
- All data in Java is an ***object*** – except for the ***primitive data types***:

`int`	4 bytes (-2,147,483,648 -> 2,147,483,647)
`short`	2 bytes (-32,768 -> 32,767)
`long`	8 bytes (really big integers)
`byte`	1 byte (-128 -> 127)
`float`	4 bytes (floating point)
`double`	8 bytes (floating point)
`char`	2 bytes (Unicode representation), **single** quotes
`boolean`	`true` or `false`

Fun fact: Python *unified* ints and longs → no longer has long

Variables

- Need to specify variable's type, i.e., it must be ***declared***, before used
  - **Syntax**: `<datatype> <name> [= value];`

Optional assignment

- Naming conventions:
  - Variable names (identifiers) typically start with *lowercase* letter
    - _ (underscore) also a valid first character
  - Subsequent words are capitalized
    - Examples: myFile, firstCousinOnceRemoved
    - Called "Camel Casing"

Variable Examples

- Need to specify variable's type, i.e., it must be ***declared***, before use
  - **Syntax**: `<datatype> <name> [= value];`

- Examples:
  - `int x;`
  - `double pi = 3.14;`
  - `char exit = 'q';`
  - `boolean isValid = false;`

Note ***must*** use *single* quotes for `chars`

Camel Casing

Python Transition **Warning**

You can**not** redeclare a variable name in the same scope

- OK:

```
int x = 3;
x = -3;
… // more code
x = 7;
```

Declaration

Definition

Python Transition **Warning**

You can**not** redeclare a variable name in the same scope

- OK:

```
int x = 3;        ← Declaration
x = -3;           ← Definition
… // more code
x = 7;
```

- Not OK:

```
int x = 3;
int x = -3;       ← Compiler errors

boolean x = true;
```

More Data Type-Related Information

- Result of integer division is an `int`
  - Same as C
  - Example: `4/3  =  ??`
- Casting
  - Similar to Python for primitive types
  - Example: `4/(double) 3`

`WinPercentage.java`

Floats in Java

- Decimal literals are considered *doubles*
- This code won't compile:

```
float f = 3.14;
```

Compiler reads `3.14` as a *double*

- Compiler error message:

```
Float.java:15: error: incompatible types: possible lossy
conversion from double to float
       float f = 3.14;
                 ^
1 error
```

- To fix code, add an `f` to specification of number or declare as `double`

`Float.java`

Arithmetic, Relational Operators

- Java has most of the same operators as Python:
  - Arithmetic operators: `+, -, *, /, %`
    - **No power operator: `**`**
  - Relational operators: `==, !=, <, >, <=, >=`
    - Evaluate to a `boolean` value
  - Increment and decrement
    - `+= x, -= y,` etc.
    - Additional shortcut for += 1, -=1: `++ , --`

`Conversion.java`

Escape Sequences

- Combination of characters to represent something else
- Escape character: \

Same as Python:

Meaning	Sequence
Newline character (carriage return)	\n
Tab	\t
Quote	\"
Backslash	\\

- In Java, you can represent a ' without escaping
- What does the following display?

```
System.out.println("To print a \\, you must use
  \"\\\\\"");
```

EscapeCharacters.java

Demo: Compiling and Running Programs

- Compiler errors:
  - Errors in the program's syntax
- Logic errors
  - Errors in your logic/coding
  - Found at runtime
  - After fixing program, need to go back and recompile

Unix Output Redirection: >

- We can redirect output to a file
  - For example

```
ls *.java > java_files.out
```

  - Above command saves the output from the `ls` command into the file named `java_files.out`
- This is how you will save output from your Java programs initially
  - For example `java Intro > out`

Please follow instructions on names in assignments

Policy: Using the Web and Others

- I provide a lot of online resources
- Most of what I ask you to do is similar to my slides or examples
  - Exception: machine/software configuration
- Use my resources first
  - Example programs are on the course web site
- Search online/ask someone else as a last resort
  - Need more experience to sort through the results you get in search engine
    - How do you get experience? More practice in CSCI209!

If it's taking more than ~3 minutes to get an answer, check in with me

Reminder: Design for Sustainability on Earth and in Space

Danielle Wood

Assistant Professor of Media Arts and Sciences, Aeronautics and Astronautics; Director of the Space Enabled Research Group, Massachusetts Institute of Technology

Thursday, September 21, 5:00 pm

Stackhouse Theater

Extra credit opportunity:

Post in Canvas discussion forum

https://my.wlu.edu/mudd-center/programs-and-events/2023-2024-ethics-of-design/danielle-wood

To Do

- Textbook: Read “Java Data Types”, up to but not including String
- Assign 0
  - Part 1: First Java Program
  - Part 2: Fix compiler and logic errors from program
  - Due before Friday’s class