

Objectives

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

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?
- How did the git lab go?

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

Review: Version Control Systems, in brief

- Track versions, changes
- Collaboration
- Documenting authorship, changes
- Sandbox
- Back up, restore

Review: Common Git Commands

Command	What it does
clone	Clones a repository – sets up your repository so that you can coordinate
add <file>	Adds the <i>file</i> to the staging area
commit	Commits all the staged files (locally)
push	Push all your changes to the remote → You need your code to be pushed so that I can see it.
branch	List all local branches
branch <name>	Creates a new branch named <i>name</i>
checkout <name>	Switches to the branch named <i>name</i>



<https://xkcd.com/1597/>

Typical Git Workflow

1. Clone repository
 - Git's default branch is `main`
2. Branch from `main` to a work-in-progress branch
 - Work on feature/next step/...
 - Add files to staging area
 - Commit with message
3. When complete, merge branch back into `main`
4. Push `main`
5. Switch back to and continue in work-in-progress branch (either same branch or new one)
6. Repeat steps 2-6

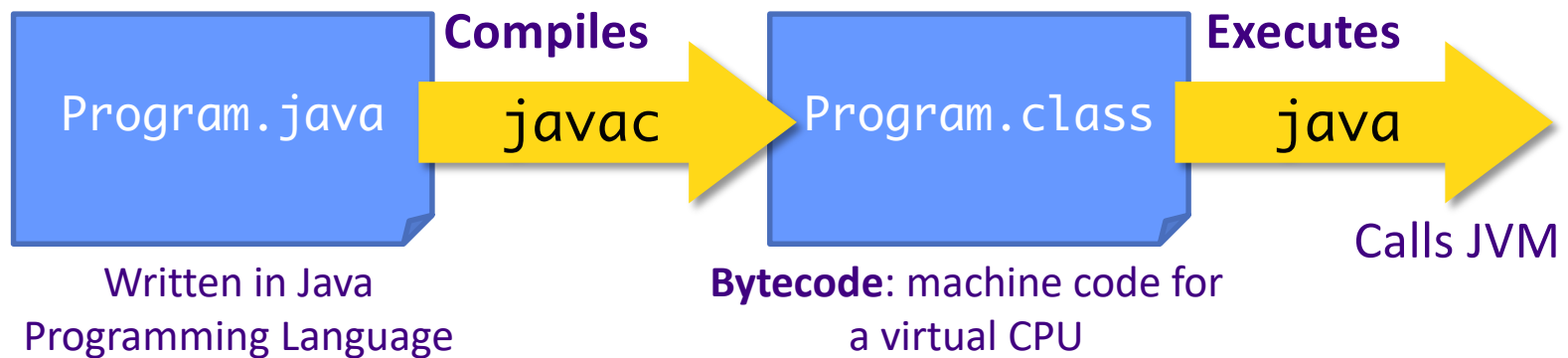
Review

- What are the benefits of Java?
- How do you compile and run Java programs?
- How do you display output in a Java program?
- 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)?

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



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

Compiling & Running a Single-Source Program

- If you want to run a Java class that's in **one** class and doesn't use need other classes, then you can compile *and* run the class using

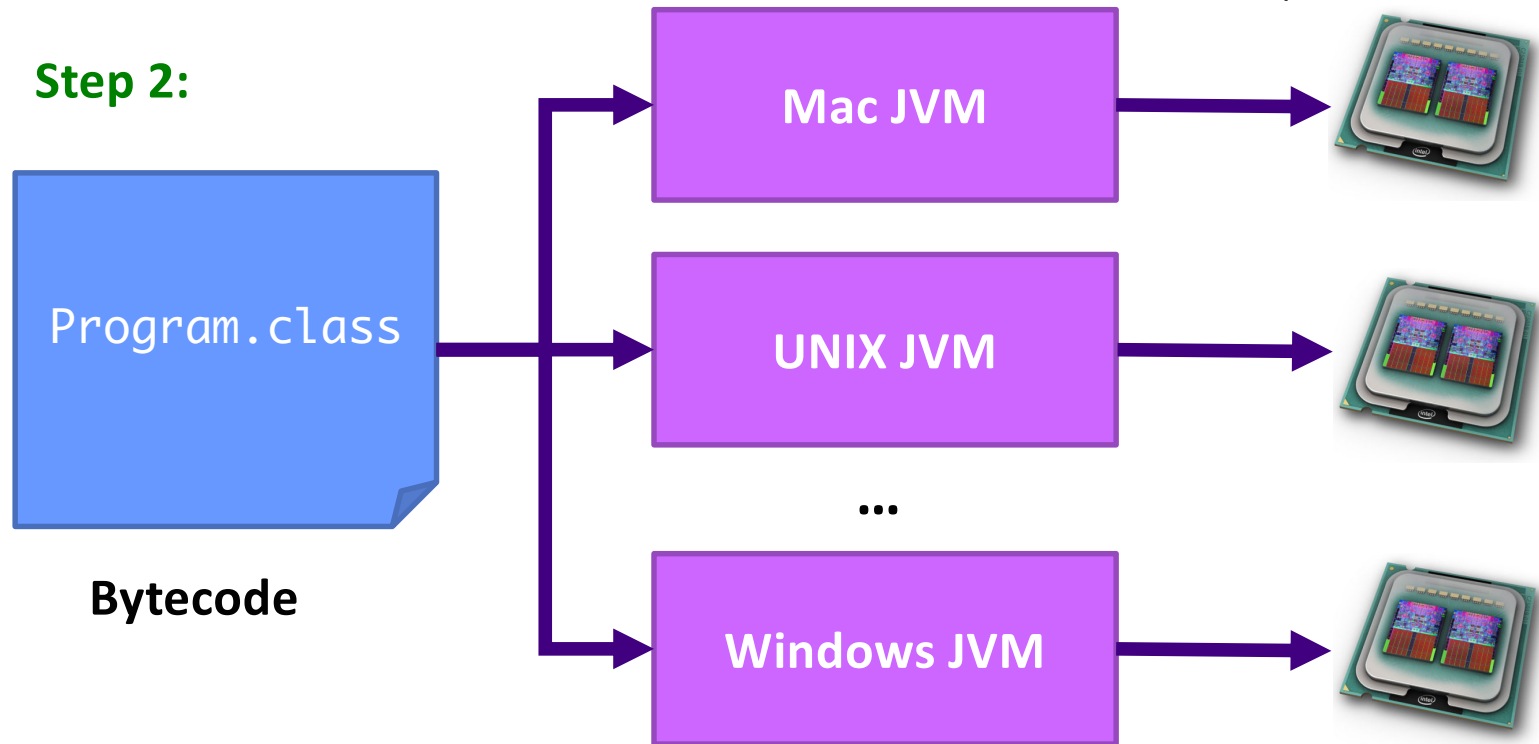
```
java Program.java
```

➤ Generated bytecode is not saved in a file

- BUT single-source program is *not* the common/general case

➤ Instead compile, then run your code

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 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*

String Concatenation

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

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

Note the +



Automatically
converted to a String

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];`

- Naming conventions: Optional assignment

➤ 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';

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

➤ **boolean** isValid = false;

Camel Casing



Python Transition **Warning**

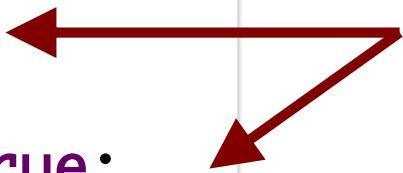
You can**not** redeclare a variable name in the same *scope*

- Not OK:

```
int x = 3;  
int x = -3;
```

```
boolean x = true;
```

Compiler errors



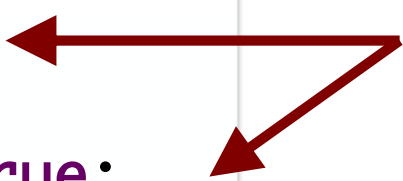
Python Transition **Warning**

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

- Not OK:

```
int x = 3;  
int x = -3;  
  
boolean x = true;
```

Compiler errors



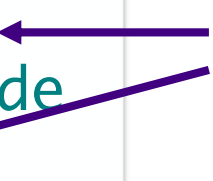
- OK:

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

Declaration



Definition



More Data Type-Related Information

- Result of integer division is an **int**
 - Same as C
 - Example: $5/3 = ??$
- If want a **double** as a result, need to *cast* at least one of numerator and denominator
 - Examples:
 - $5/(\text{double})\ 3$
 - $(\text{double})\ \text{numerator}/\text{denominator}$

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`

Arithmetic, Relational Operators

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

Escape Sequences

Same as Python:

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

- Combination of characters to represent something else

- Escape character: \

- In Java, you can represent a ' without escaping

- What does the following display?

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

Demo: Compiling and Running Programs

- Compiler errors:
 - Errors in the program's syntax or violations of Java rules
- 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 > intro.out`

Policy: Using the Web, AI, 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
- Search online/ask someone else as a last resort
 - Need more experience to sort through the results you get in search engine and to understand results from AI
 - How do you get experience? More practice in CSCI2100!

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

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