Objectives

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

1

Weekends

- Sorry responses so short
- Emailing through voice-to-text

2

Review

- What are qualities of good software?
- What are the benefits of version control?
- What are some of the common Git commands and what do they do?

Purpose of questions?

3

Git Notes

- Typical Git workflow
  1. Branch from `main` to a work-in-progress branch
  2. Work on feature/next step/…
  3. When complete, merge branch back into `main`
     - Optionally, push `main`
  4. Switch back to and continue in work-in-progress branch
  5. Repeat
- Typically, only push *main* branch
  - ➢ won't push your work-in-progress branches unless need debugging help

4

Why the Command Line?

- Because you *should* know it
  - Alumni feedback
- It can make your development process quicker
  - After you get used to it
- Because you look so badass using it

5

Suggestion

- Reload assignment pages whenever you return to them
  - Get most recent updates
  - I may have addressed issues that students alerted me to

6

INTRODUCTION TO JAVA

7

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

http://www.tiobe.com/tiobe-index/

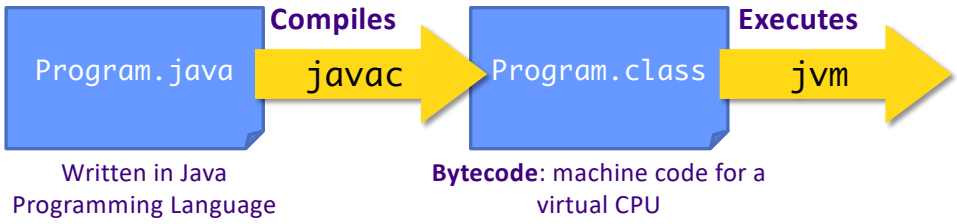
8

What is Java?

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

9

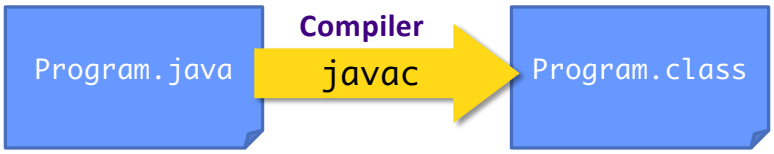
Overview: Compiling, Executing Java Programs



10

Compiling Java Programs

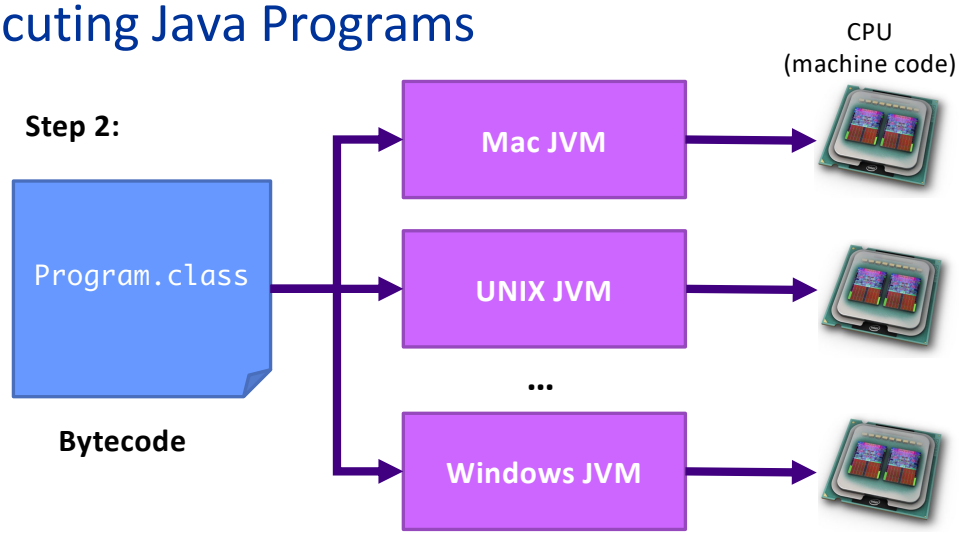
Step 1:



Written in Java Programming Language

Bytecode: machine code for a virtual CPU

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*
- HotSpot VM
  - Code dynamically compiled to machine code
- Garbage Collection

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

How does Java's approach affect distribution of software?

2 - How does software being Java-based affect its distribution?

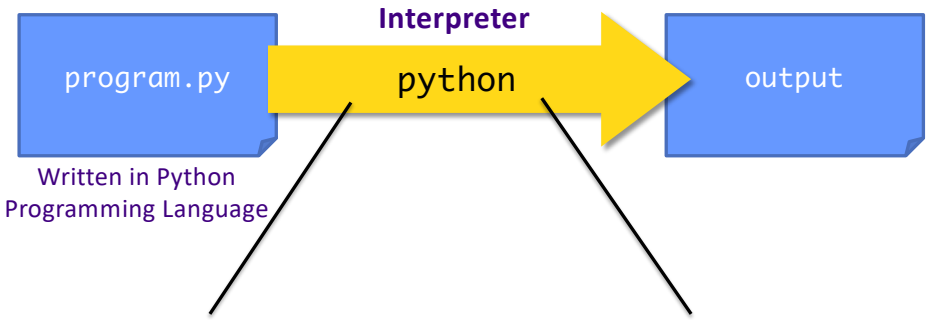
Makes it harder because you have to install a JVM on every machine	A
Makes it more secure because you don't provide the source code	B
Makes it easier because same bytecode can be run on multiple platforms	C
Makes it easier because many machines already have Java installed	D
None of the above	E

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

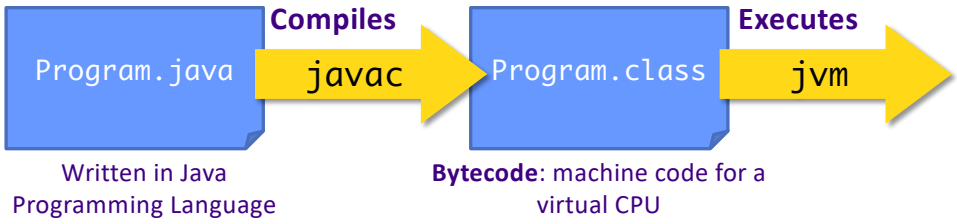
How are (I) Java and (II) the traditional approach the same and different from Python's approach?

Executing Python Programs



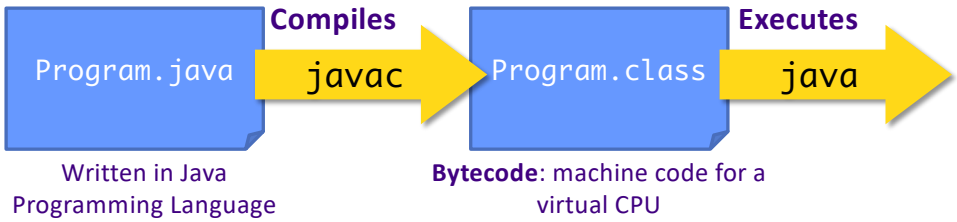
1. Syntax validation
   - exit if not valid
2. Translate Python code to Python bytecode
   - Bytecode stored in __pycache__ directory
3. Python virtual machine execute Python bytecode

Overview: Compiling, Executing Java Programs



JDK: Java Development Kit

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



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/16/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:
    http://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!

Python Review

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

main()
```

What does this program do?

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**

Example Java Program

Blocks of code marked with { }

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

Defines the class "Hello"

- In general, each Java program file contains **one** class definition*
- Name of the class is name of file
  - E.g., `Hello.java`

Example Java Program

```
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

Example Java Program

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

- Class contains one *method*: `main`

Example Java Program: `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?

Example Java Program: 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

Example Java Program: 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

Code Style

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

- **Comments** at top of program
  - Sprenkle CSCI209 requirements:
    - **Must** include your name
    - **Must** include high-level description of program
- Proper **indentation**
  - Similar to Python
  - Everything within pairs of {} is indented the same

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

A Note About 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

What are the Differences?

```
# 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…

Literal Translation 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

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

   def __init__(self):
       # fill in later…

   def main(self):
       print("Hello")
```

Semi-literal translation

JAVA FUNDAMENTALS

Print Statement

- Syntax:

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

No newline

- Similar to Python's `file.write()` method
  - Need to combine parameter into one `String` using +'s
    - 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 variable is converted to a string.

Note the +

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

Automatically converted to a String

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
- 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

Looking Ahead

- Register for Text Book
  - Start reading Chapter 1 through 1.4: Lets look at a Java Program
- Complete Assignment 0 by 11:59 p.m. Tuesday
- Official Office Hours: 1-2 p.m.
  - BUT email me and we can Zoom to resolve the git issues