Objectives

- Basics of Java Syntax
- Java fundamentals
  - Primitive data types
  - Static typing
  - Arithmetic operators
  - Relational operators

Review

- What are qualities of good software?
- What is Java?
  - Benefits to using Java?
- How do you compile a Java program?
  - How do you run/execute a Java program?

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
- Statically-typed language
  - Compiling - find some errors before execution, performance benefits

Review: Java Programming Language

- Entirely object-oriented
- Similar to Python

Step 1:

- Written in Java Programming Language
- Bytecode: machine code for a virtual CPU
- Compiler: javac
- Program.java
- Program.class

Review: Java Virtual Machine

- Same bytecode executes on each platform
- Don’t need to provide the source code

Step 2:

- Bytecode
- Mac JVM
- UNIX JVM
- Windows JVM
- ...
Python Review

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

What does this program do?

Example Java Program

```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

• Everything in Java is inside a class
  ➢ Java is entirely object-oriented
  ➢ This class is named Hello

Example Java Program

• In general, each Java program file contains one class definition
  ➢ E.g., Hello.java

Example Java Program

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

Blocks of code marked with { }

Example Java Program

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

Defines the class "Hello"

Example Java Program

• Class contains one method: main

Access Modifier:
controls if other classes can use code in this class
Example Java Program: **main Method**

```java
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
- Must be `public static`
- Must be `void`: data type of what method returns (nothing)
- `main` is automatically called when program is executed from command line

Example Java Program: **Print Statements**

```java
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**

```java
/**
 * Our first Java class
 * @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

Example Java Program: **What are the Differences?**

```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");
    }
}
```

- Comments at top of program
  - Must include your name
  - High-level description of program
- Proper indentation
  - Similar to Python
  - Everything within sets of `{}` is indented the same
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?

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

Translation to Python Program

```java
class Hello:
    """Our first Python class""
    def __init__(self):
        # fill in later...
    def main(self):
        print("Hello")
```

Semi-literal translation

Print Statement

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

- 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 `+`

  ```java
  System.out.println("Your score is " + 78);
  ```
  - Automatically converted to a `String`
Escape Sequences

- Same as Python:
<table>
<thead>
<tr>
<th>Meaning</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newline character</td>
<td>\n</td>
</tr>
<tr>
<td>(carriage return)</td>
<td></td>
</tr>
<tr>
<td>Tab</td>
<td>\t</td>
</tr>
<tr>
<td>Quote</td>
<td>&quot;</td>
</tr>
<tr>
<td>Backslash</td>
<td>\</td>
</tr>
</tbody>
</table>

- In Java, you can print a `\` without escaping.
- What does the following display?
  ```
  System.out.println("To print a \\, you must use \\\\\\\\\
  ");
  ```

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://download.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html](http://download.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html)

Data Types

- Java is **strongly-typed**
  - Every variable must be a declared type
- All data in Java is an **object** except for the **primitive data types**:
<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int</code></td>
<td>4 bytes</td>
<td>-2,147,483,648 -&gt; 2,147,483,647</td>
</tr>
<tr>
<td><code>short</code></td>
<td>2 bytes</td>
<td>-32,768 -&gt; 32,767</td>
</tr>
<tr>
<td><code>long</code></td>
<td>8 bytes</td>
<td>(really big integers)</td>
</tr>
<tr>
<td><code>byte</code></td>
<td>1 byte</td>
<td>-128 -&gt; 127</td>
</tr>
<tr>
<td><code>float</code></td>
<td>4 bytes</td>
<td>(floating point)</td>
</tr>
<tr>
<td><code>double</code></td>
<td>8 bytes</td>
<td>(floating point)</td>
</tr>
<tr>
<td><code>char</code></td>
<td>2 bytes</td>
<td>(Unicode representation), single quotes</td>
</tr>
<tr>
<td><code>boolean</code></td>
<td>false or true</td>
<td></td>
</tr>
</tbody>
</table>

Variables

- Must be **declared** before used
  - **Syntax**: `<datatype> <name> [= value];`
  - Optional assignment
- Variable names typically start with lowercase letter
  - `_` (underscore) also a valid first character
  - Convention: Subsequent words are capitalized
    - Called "Camel Casing"

Variable Examples

- Must be **declared** before used
  - **Syntax**: `<datatype> <name> [= value];`
- Examples:
  - `int x;`
  - `double pi = 3.14;`
  - `char exit = 'q';`
  - `boolean isValid = false;`
  - **Camel Casing**

Floats in Java

- Decimal literals are considered **doubles**
- This code won’t compile:
  ```java
  float f = 3.14;
  ```
  - Compiler reads 3.14 as a double
- Compiler error message:
  ```java
  Float.java:13: possible loss of precision
  found    : double
  required: float
  f = 3.14;
  ```
- To fix code, add an `f` to specification of number or declare as `double`
Python Transition Warning

You cannot redeclare a variable name in the same scope

- OK:
  ```java
  int x = 3;
  x = -3;
  ```

- Not OK:
  ```java
  int x = 3;
  int x = -3;
  boolean x = true;
  ```

More Data Type Information

- Default data types
  - Same as Python 2, not Python 3
  - Result of integer division is an `int`
    - Example: `4/3 = ??`

- Casting
  - Similar to Python for primitive types
  - Example: `4/(double) 3`

Formatted Output

- `printf` or `format`
  - `PrintStream` functionality since Java 1.5
  - `System.out` is a `PrintStream` object

```java
double d1=3.14159, d2=1.45, total=9.43;

// simple formatting...
System.out.printf("\%6.5f and \%5.2f ", d1, d2);
// \% is platform-specific line separator,
// e.g., \n or \n
System.out.printf("\%-6s\%5.2f\n", "Tax:", total);
```

Output Redirection: >

- In UNIX, we can redirect output to a file
  - For example
    ```bash
    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
      ```bash
      java Intro > out
      ```

Programming Assignment 0

1. Write a program called `Intro.java`
   - Displays information about yourself
2. Fix compiler and logic errors in a program

- Due Wednesday before class
- Go to Course Web Page
  - Bookmark it!
- Activate your Piazza account and check it out