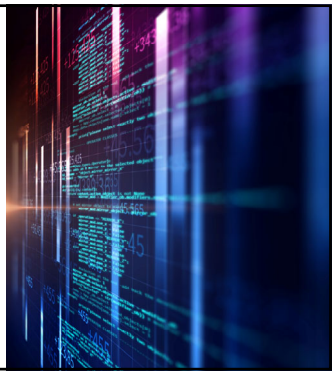
Introduction to Java

Brandon Krakowsky

Penn Engineering



1

Introduction to Java

2

First … Can We Forget About Python?

- In terms of syntax, mostly …
- But you SHOULD NOT forget about:
  - Code reuse
  - Modular programming
  - Test-Driven Development
  - Good Style (Java is stricter in many ways)
  - Commenting your code
  - Etc.

3

Java vs. Python

- Java and Python are *similar* in that they're both *object-oriented languages*
  - Conceptually, the languages are very similar
  - The syntax is quite different, while Java syntax is much more verbose
    - It is both explicit (and strict), which can be a good thing
  - Transitioning from Python to Java has a lot to do with learning the new syntax
- Java and Python are *different* in that Java is *compiled* and Python is *interpreted*
  - This allows Java to run much faster and more efficiently
  - It also allows your Java code to be inspected for all kinds of errors, including syntax errors, type errors, and non-existing functions

Java is Compiled

- When Java is compiled, it's converted to binary machine code (or Java *bytecode)*
  - This allows Java programs to be "portable" and run on different machines and operating systems
- *Compiled* languages have many advantages over *interpreted* languages
  - When code is compiled, it's optimized under the hood
  - Since your program will be inspected for errors, many kinds of potential bugs will be caught early (e.g. using the same variable name twice)
- Your program will not run if it is not compiled!
- The IDE we'll be using for Java development, Eclipse, will compile your code for you (on the fly) as you save your work
  - It will also help you fix MANY problems in your code

Popularity of Java vs. Python Using TIOBE

- The TIOBE Programming Community index is an indicator of the popularity of programming languages
- It can be used to:
  - Check whether your programming skills are up to date
  - Make a decision about what programming language(s) to use when starting new projects
- The ratings are:
  - Based on the number of skilled engineers world-wide, courses and third party vendors
  - Calculated based on popular search engines
- The index is updated once a month

Ref: https://www.tiobe.com/tiobe-index/

Popularity of Java vs. Python Using TIOBE

- Top 10 of the TIOBE index for October 2020

Oct 2020	Oct 2019	Change	Programming Language	Ratings	Change
1	2	^	C	16.95%	+0.77%
2	1	v	Java	12.56%	-4.32%
3	3		Python	11.28%	+2.19%
4	4		C++	6.94%	+0.71%
5	5		C#	4.16%	+0.30%
6	6		Visual Basic	3.97%	+0.23%
7	7		JavaScript	2.14%	+0.06%
8	9	^	PHP	2.09%	+0.18%
9	15	^	R	1.99%	+0.73%
10	8	v	SQL	1.57%	-0.37%

- General highlights:
  - Java and Python are in the top 3 most popular programming languages
  - Currently, both languages have *almost the same rating*

Ref: https://www.tiobe.com/tiobe-index/

Configuring Java & Tools

Installing & Running Java

- In order to use Java, you need to first install the Java Development Kit (JDK)
  - This is the package of tools for *developing* Java-based software
- You'll also need the Java Runtime Environment (JRE) which includes the Java Virtual Machine (JVM)
  - This is the environment for *running* Java applications
    - The JVM is what actually runs compiled Java bytecode
- Download and install the JDK, which includes the JRE (and JVM): https://www.oracle.com/java/technologies/javase-downloads.html

Downloading and Installing the JDK

- Download and install the JDK, which includes the JRE (and JVM): https://www.oracle.com/java/technologies/javase-downloads.html
  - Locate the main link for the JDK

Downloading and Installing the JDK

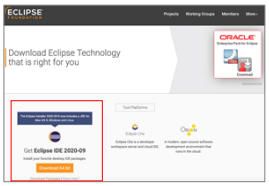
- Download and install the JDK, which includes the JRE (and JVM): https://www.oracle.com/java/technologies/javase-downloads.html
  - Download the latest version of the JDK for your OS

Eclipse

- Eclipse is one of two main IDEs for Java development
  - The other IDE is IntelliJ
  - I'll work with Eclipse
- Eclipse makes it very easy to write well-formatted Java, with good style
  - Like Python's PyCharm, it has a TON of features
  - It compiles code on the fly, provides autocomplete suggestions, and fixes simple bugs
  - Overall, Eclipse greatly speeds up Java programming
- Getting Eclipse:
  - Go to https://www.eclipse.org/downloads/ and download the latest version

Installing & Configuring Eclipse

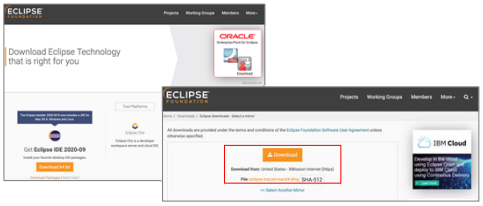
- Install Eclipse via https://www.eclipse.org/downloads/
  - Scroll down to get the latest version of Eclipse



13

Installing & Configuring Eclipse

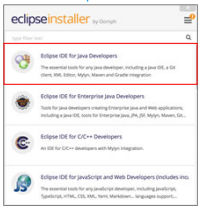
- Install Eclipse via https://www.eclipse.org/downloads/
  - Click to download the latest version of the IDE for your OS



14

Installing & Configuring Eclipse

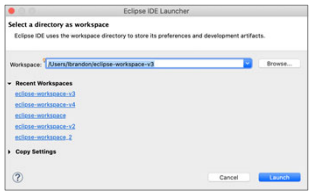
- When you extract and run the Eclipse Installer
  - Choose Eclipse IDE for Java Developers



15

Installing & Configuring Eclipse

- When you launch Eclipse, you need to specify a workspace location
  - You can use the default option (unless you have a really strong need to change it)
  - Click "Launch"



16

Java & Eclipse

- Eclipse stores projects in a workspace
- When you use Eclipse to create a project (a single "program"), it creates a directory with that name in your workspace
- Within the project, you create an *optional* package (a sub-directory)
- Finally, within the package, you create a class (a file)
- For the simplest program, you'll only need a single package (or the default "no" package), and only one (or a very few) classes
  - Java is object-oriented and class-based, which means you have to create *at least one class* to write a Java program

17

Java Language

18

Simple Introductory Java Program

```
//Optional package declaration
package myPackage; //Should begin with a lowercase letter

//Class declaration
public class MyClass { //Should begin with a capital letter
//The Java file will be named (and saved in) 'myPackage/MyClass.java'

    //Main method -- the starting point of any Java program
    //In Java, the name "main" is special and reserved for the main
method
    public static void main(String[ ] args) {
       System.out.println("Hello World"); //Prints 'Hello World'
    }
}
```


Some General Rules for Java

- Individual statements end in a semicolon
  - New lines do not mean anything in Java
  - This means you COULD have an entire program on one line
    - Obviously, this is bad style!
- For example, here's a statement

```
System.out.println("Hello World!");
```

- Here's another statement

```
String myString = "My String";
```


Some General Rules for Java

- Indentation doesn't matter
  - Unlike Python, where it's required, indentation in Java is a matter of style
  - While it won't make your program fail the way it does in Python, you should not stop indenting your programs!
- You can use these shortcuts in Eclipse
  - Fixes format of your code

    CTRL/Cmd + SHIFT + F
  - Selects all code in Java file and fixes indentation

    CTRL/Cmd + A, CTRL/Cmd + I

Some General Rules for Java

- Java uses curly braces { } to surround code blocks
  - Unlike Python, which uses a colon (:) and indentation to indicate code blocks
- For example, here's a conditional

```
if (myVar == true) {
    //code block
}
```

- And here's a function

```
public void myFunction() {
    //code block
}
```

- For purposes of style, an opening brace { should go at the end of a line, not on a line by itself

Penn Engineering

22

Variables & Types

- You typically name variables using "camelCase", starting with a lowercase letter
- Every variable in Java has a pre-defined *type*
  - You declare the *type* in front of the variable

```
int myInt = 0; //myInt can only store an int
```

- You MUST store that kind of data in the variable
  - For example, you can't do this:

```
int myInt = "hello";
```

  - Eclipse won't even let you compile your code!
- The *type* of a variable CANNOT be changed
  - Java is *statically* typed
  - In Python, you can change variable types on the fly, because it's *dynamically* typed

Penn Engineering

23

Variables & Types

- Some primitive (simple) data types
  - int: Integer
  - float: Floating point (decimal)
  - boolean: true/false
- Some other primitive types
  - char: Single character
  - double: Large and precise floating point
  - byte, short, or long: Various integer sizes (8, 16, 64 bits)
- Another type is String, which is an Object (not a primitive)
  - It's used to store a *character string*
- You might also come across Integer, Boolean, Double, etc.
  - Don't worry about these for now!

Penn Engineering

24

Variables & Types

- You can declare variables WITH initial values

```
int count = 0;
String firstName = “Brandon”;
```

- Or declare variables WITHOUT initial values

```
double distance; //Declares a double without actually creating a double
String color; //Declares a String without actually creating a String
```

- And obviously set the variables later

```
distance = 2.3;
color = “red”;
```


Variables & Types - Strings vs. Chars

- There is a difference between a *single character* and a *character string*
  - Unlike Python, be careful about when you are using double quotes vs. single quotes
- To define a String, use double quotes

```
String firstName = “Brandon”; //”Brandon” is a String
```

- To define a char, use single quotes

```
char letter = ‘a’; //‘a’ is a char
```

- Like in Python, you can concatenate Strings using +

```
String fullName = “Brandon” + “ ” + “Krakowsky”;
```

- Tip: Anything concatenated with a String is automatically converted to a String
- For example:

```
String myResult = "There are " + appleCount + " apples and " +
orangeCount + " oranges.”;
```

  - Note the difference with Python, where you have to call the *str* method to cast to a String

Printing

- There are two methods you can use for printing:

```
//This prints something and ends the line
System.out.println(something);

//This prints something and doesn’t end the line (so the next thing you
print will go on the same line)
System.out.print(something);
```

- These methods will print any one thing, but only one at a time
- Of course, you can always concatenate Strings with the + operator
- Example:

```
System.out.println("Four " + 4 + ", three " + 3 + ", two " + 2 + ", one
" + 1);
```

while Loops

- *while* loops in Java have a similar syntax to *while* loops in Python
- Simple while loop that iterates 10 times:

```
int i = 0;
while (i < 10) {
    //do stuff here every time loop happens
    i++; //manually increment i
}
//i is initially set to 0
//i must be less than 10 in order to enter the loop each time
//code in the loop manually increments i by 1 at the end of each loop
```


for Loops

- *for* loops in Java have a very different syntax than *for* loops in Python
  - But they are equivalent to: `for i in range(10)`
- A *for* loop has 3 parts:
  - Setting the initial value
  - The condition for entering the loop
  - The change in the loop variable that happens at the end of each loop
- Simple for loop that iterates 10 times:

```
for (int i = 0; i < 10; i++) {
    //do stuff here every time loop happens
}
//i is initially set to 0
//i must be less than 10 in order to enter the loop each time
//i is incremented by 1 at the end of each loop (you can't see it)
```


Getting Input

- First, import the Scanner class:
  `import java.util.Scanner;`
- Create a scanner and assign it to a variable:
  `Scanner scan = new Scanner(System.in);`
  - The name of the scanner is scan
  - new Scanner(…) tells Java to make a new one
  - System.in tells Java that the scanner is to take input from the keyboard
- To read in the next int:
  `int myNumber = scan.nextInt();`
- To read in the next String:
  `String myString = scan.next();`
- To read in the entire next line as a String:
  `String myLine = scan.nextLine();`

Java Comments

- Here is a single line comment, using double slashes //

```
//Here is an int, initially set to 0
int myInt = 0;
```

- Here is a block comment, using /* */

```
/*
 * Here is an int
 * It's initially set to 0
 */
int myInt = 0;
```

- As a shortcut in Eclipse, you can type the following

```
/*
```

and then hit Enter

- It will add a block comment and you can fill in the rest

31

Javadocs

- You can add Javadocs (Java documentation) just *before* the definition of a variable, method, or class
  - This is the equivalent of a docstring inside of a Python function or class
- As a shortcut, you can type the following right above a variable, method, or class name

```
/**
```

and then hit Enter

- It will add a javadoc block and you can fill in the rest

```
/**
 * Returns the sum of two given numbers.
 * @param firstNum First value to add
 * @param secondNum Second value to add
 * @return Sum of values
 */
public int getSum(int firstNum, int secondNum) {
    return firstNum + secondNum;
}
```

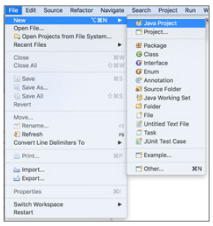

32

My First Java Project

33

My First Java Project

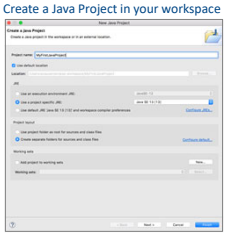
- In Eclipse, go to "File" → "New" → "Java Project"



34

My First Java Project

- Create a Java Project in your workspace



Provide a Project name
- Project names should be capitalized

Use the default location

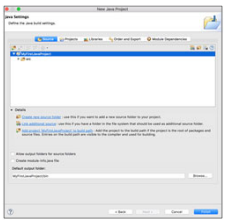
Use the default JRE and project layout

Click "Next"

35

My First Java Project

- Define the compilation/build settings



Make sure Create module-info.java file
IS NOT checked

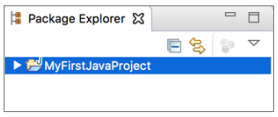
Use the default output folder

Click "Finish"

36

My First Java Project

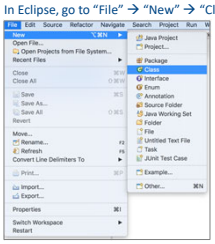
- The project will appear in the Package Explorer on the left hand side in the IDE



37

My First Java Project

- In Eclipse, go to "File" → "New" → "Class"



38

My First Java Project

- Create a Java Class in your Java Project

Provide a Name
- Class names should be capitalized

Make sure public static void main(String[ ] args) IS checked

Make sure Inherited abstract methods IS NOT checked

Click "Finish"

39

My First Java Project

- The entry point of any java program is the *main* method

FirstClass.java

```
1
2  public class FirstClass {
3
4      public static void main(String[] args) {
5          // TODO Auto-generated method stub
6
7      }
8
9  }
10
```


40

My First Java Project

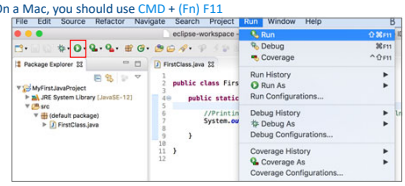
FirstClass.java

```
3  public class FirstClass {
4
5      public static void main(String[] args) {
6
7          //Printing using SOPL: Short for System.out.println()
8          System.out.println("Hello World!");
9          System.out.println(); //print a blank line
10
11         /*
12          * Defining variables
13          */
14
15         //Format: Datatype varname = value;
16         int x = 5; //int
17
18         double y = 5.0; //replaces float in Python
19
20         boolean flag = true; //replaces True in Python
21
22         //concatenating Strings to non-Strings and printing
23         System.out.println("x: " + x);
24         System.out.println("y: " + y);
25         System.out.println("flag: " + flag);
```


41

My First Java Project

- To run your Java program in Eclipse, go to Run → Run
  - Or click the "Run" button
- Keyboard shortcuts will vary based on your install of Eclipse and operating system
  - On a Mac, you should use CMD + (Fn) F11



42

My First Java Project

```
36        /*
37         * Strings and characters
38         */
39
40        //In Python, no difference between double quotes ("") and single quotes ('')
41        //In Java, use double quotes ("") for Strings, and single quotes ('') for chars
42        String dept = "CIT"; //String
43        char letter = 'a'; //char
44
45        //Anything concatenated to a String is converted to a String
46        String course = dept + 590; //String with an int
47        String grade = letter + ""; //char with a String
48
49        //Variables are typically named with camelCasing
50        String courseInformation = course + ": " + grade;
51        System.out.println(courseInformation);
```


My First Java Project

```
44        /*
45         * Math operations
46         */
47
48        double d = 2 * x + 10;
49        double z = 2 * y + 5;
50
51        System.out.println("d: " + d);
52        System.out.println("z: " + z);
53
54        //Division with ints
55        //Uses integer division and ignores the remainder
56        System.out.println("x / 2: " + (x / 2));
57
58        //Division with floats
59        System.out.println("x / 2.0: " + (x / 2.0));
60
61        //Power operation is different from Python
62        System.out.println("x pow 4: " + Math.pow(x, 4));
```


My First Java Project

```
72        /*
73         * String operations
74         */
75
76        //String concatenation
77        String fullName = "Brandon" + " " + "Krakowsky";
78
79        //String method for converting to upper-case
80        String fullNameUpper = fullName.toUpperCase();
81        System.out.println(fullNameUpper);
82
83        //There is no String multiplication in Java
84        //You can't do this
85        //String threeZs = "z" * 3;
```

My First Java Project

```
78        /*
79         * Conditionals and loops
80         */
81
82        //Conditional checking if x is even using the modulus % operator
83        System.out.println("x: " + x);
84        if (x % 2 == 0) {
85            System.out.println("x is even");
86        } else {
87            System.out.println("x is odd");
88        }
```


46

My First Java Project

```
90        double e = 2.3;
91        double f = 2.4;
92        double g = 2.5;
93
94        //boolean operators
95        //  && (and) - true only if both operands are true
96        //  || (or) - true if either operand is true
97        //  ! (not) - reverses the truth value of its one operand
98        if (e > 2 && e < f) {
99            System.out.println(e + " is between 2 and " + f);
100       }
101
102       if (f > e || f > g) {
103           System.out.println(f + " is either greater than " + e + " or greater than " + g);
104       }
105
106       if (g != 2.6) {
107           System.out.println(g + " is not equal to 2.6");
108       }
```


47

My First Java Project

```
110       //while loops
111       //very similar in Python
112       int i = 0;
113       while (i < 5) {
114           System.out.println("i: " + i);
115
116           //increment i
117           i++; //same as i = i + 1
118       }
```


48

My First Java Project

```
120         //for loops
121         //Python equivalent is for k in range(10):
122         for (int k = 0; k < 10; k++) {
123             System.out.println("k: " + k);
124         }
125         //for loop has 3 parts:
126         //  Setting initial value: This part (k = 0) is done first and only once.
127         //  Condition for entering the loop: The condition (k < 10) is tested before each loop.
128         //    If it's true, enter the loop.
129         //  Change in the loop variable: The increment (k++) happens at the end of each loop.
```


49

My First Java Project

```
137         /*
138          * Casting
139          */
140
141         //Cast int 1 to String
142         String intToString = Integer.toString(1);
143
144         //Cast double 1.1 to String
145         String doubleToString = Double.toString(1.1);
146
147         //Get class (or type) of String (Object) doubleToString
148         //Strings (and other Objects) have getClass() method
149         System.out.println(doubleToString.getClass());
150
151         //Cast String "1" to int
152         int stringToInt = Integer.parseInt("1");
153
154         //Cast String "1.1" to double
155         double stringToDouble = Double.parseDouble("1.1");
156
157         //Get class (or type) of double (primitive) stringToDouble
158         //doubles (and other primitives) don't have getClass() method
159         //First you need to cast to a generic Object, then call getClass()
160         System.out.println(((Object)stringToDouble).getClass());
161
```


50

My First Java Project

- The Scanner class requires an import at the top of the class

FirstClass.java

```
1  import java.util.Scanner;
2
```

```
159         /*
160          * Input
161          */
162
163         Scanner scan = new Scanner(System.in);
164
165         System.out.println("Enter a number: ");
166         int myInt = scan.nextInt(); //get next input value as int
167         System.out.println("Your number is: " + myInt);
168
169         //print multiplication table up to 10 for myInt
170         for (int t = 1; t < 11; t++) {
171             //print char at index u
172             System.out.println(t + " x " + myInt + ": " + (t * myInt));
173         }
174
175         System.out.println("Enter a String: ");
176         String myStr = scan.next(); //get next input value as String
177         System.out.println("Your String is: " + myStr);
178
179         //print each char of myStr
180         for (int u = 0; u < myStr.length(); u++) {
181             //print char at index u
182             System.out.println(myStr.charAt(u));
183         }
184
185         scan.close(); //you should always close your scanner
186
```


51

My First Java Project

- Add javadocs to class, method, and variable definitions
- We'll eventually learn that javadocs are useful for easily creating documentation for an entire program
  - This can be extremely helpful for other programmers reading/running your code

```
/**
 * My first Java Class.
 * @author lbrandon
 */
public class FirstClass {

    /**
     * This is the javadoc for a Java method.  It's equivalent to a docstring for a function in Python.
     * The main method is the entry point of any Java program.
     * @param args
     */
    public static void main(String[] args) {
```