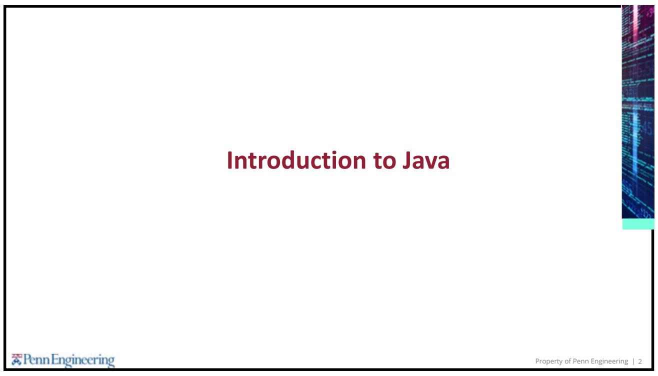
Introduction to Java

Brandon Krakowsky

Penn Engineering

1

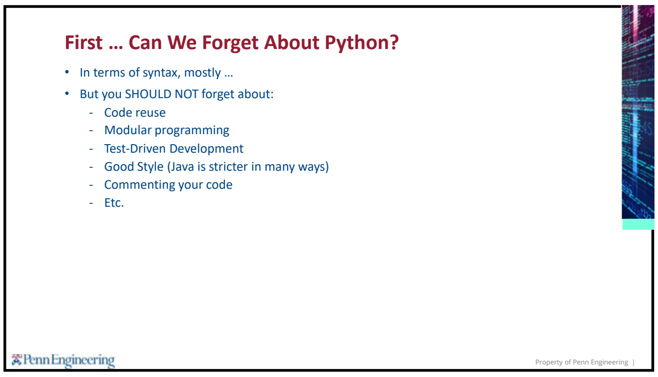


Introduction to Java

Penn Engineering

Property of Penn Engineering | 2

2



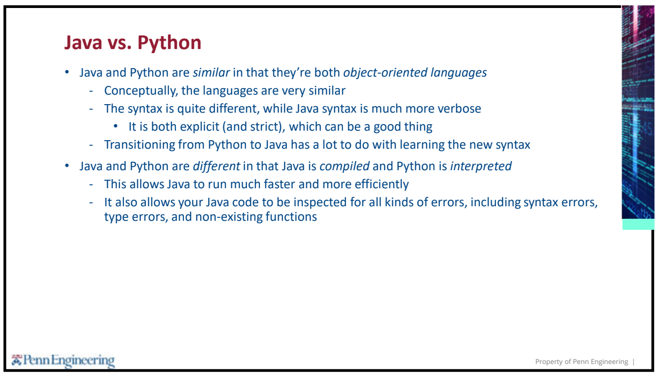
First … Can We Forget About Python?

- 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.

Penn Engineering

Property of Penn Engineering |

3



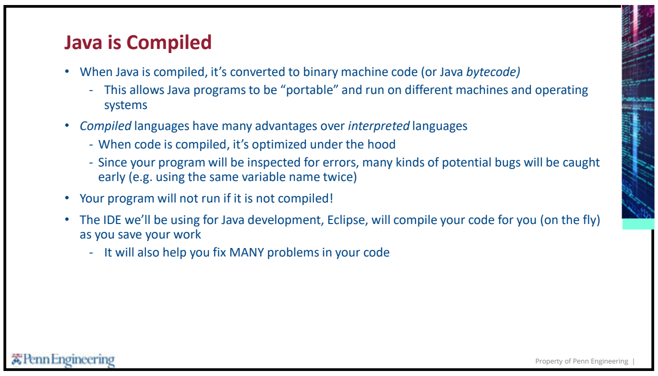
Java vs. Python

- 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

Penn Engineering

Property of Penn Engineering |

4



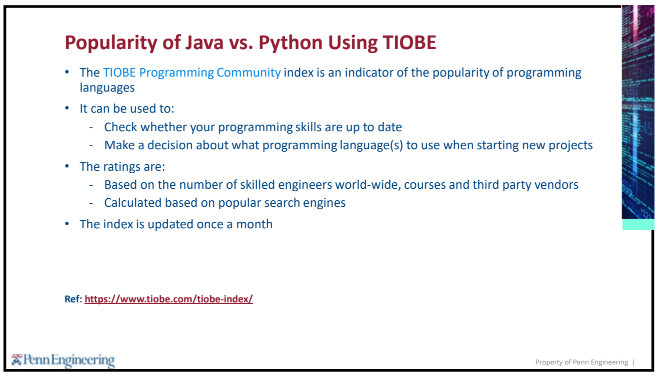
Java is Compiled

- 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

Penn Engineering

Property of Penn Engineering |

5



Popularity of Java vs. Python Using TIOBE

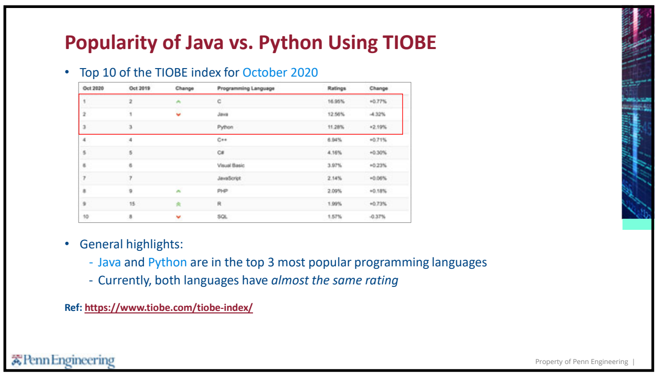
- The TIOBE Programming Community index is an indicator of the popularity of programming languages
- It can be used to:
  - Check whether your programming skills are up to date
  - Make a decision about what programming language(s) to use when starting new projects
- The ratings are:
  - Based on the 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/

Penn Engineering

Property of Penn Engineering |

6



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	▼	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	▼	SQL	1.57%	-0.37%

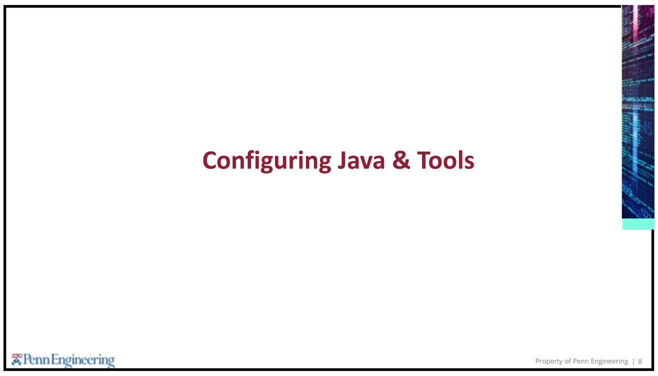
- General highlights:
  - Java and Python are in the top 3 most popular programming languages
  - Currently, both languages have *almost the same rating*

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

Penn Engineering

Property of Penn Engineering |

7

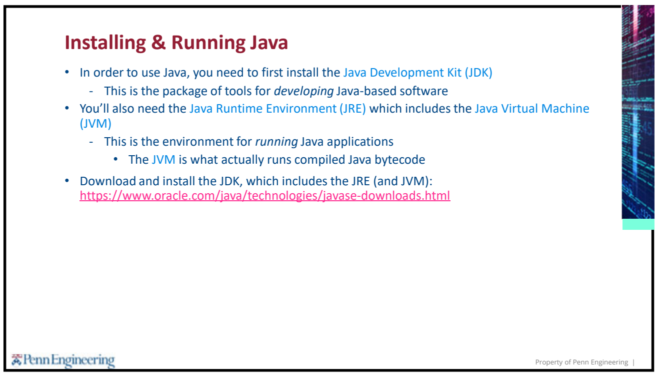


Configuring Java & Tools

Penn Engineering

Property of Penn Engineering | 8

8



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

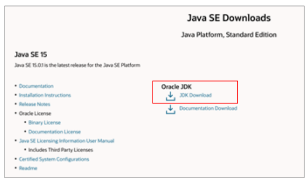
Penn Engineering

Property of Penn Engineering |

9

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



Java SE Downloads

Java Platform, Standard Edition

Java SE 15

Java SE 15.0.1 is the latest release for the Java SE Platform

- Documentation
- Installation Instructions
- Release Notes
- Oracle License
  - Binary License
  - Documentation License
- Java SE Licensing Information User Manual
  - Includes Third Party Licenses
- Certified System Configurations
- Readme

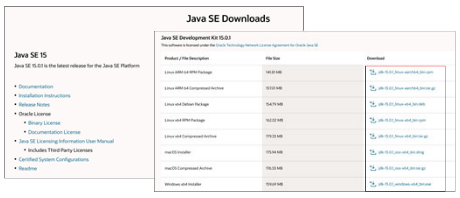
Oracle JDK
JDK Download
Documentation Download

Penn Engineering   Property of Penn Engineering |

10

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



Java SE Downloads

Java SE 15

Java SE 15.0.1 is the latest release for the Java SE Platform

- Documentation
- Installation Instructions
- Release Notes
- Oracle License
  - Binary License
  - Documentation License
- Java SE Licensing Information User Manual
  - Includes Third Party Licenses
- Certified System Configurations
- Readme

Java SE Development Kit 15.0.1

This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE

Product / File Description	File Size	Download
Linux ARM 64 RPM Package	141.81 MB	jdk-15.0.1_linux-aarch64_bin.rpm
Linux ARM 64 Compressed Archive	157.01 MB	jdk-15.0.1_linux-aarch64_bin.tar.gz
Linux x64 Debian Package	154.79 MB	jdk-15.0.1_linux-x64_bin.deb
Linux x64 RPM Package	162.02 MB	jdk-15.0.1_linux-x64_bin.rpm
Linux x64 Compressed Archive	179.33 MB	jdk-15.0.1_linux-x64_bin.tar.gz
macOS Installer	175.94 MB	jdk-15.0.1_osx-x64_bin.dmg
macOS Compressed Archive	176.33 MB	jdk-15.0.1_osx-x64_bin.tar.gz
Windows x64 Installer	159.69 MB	jdk-15.0.1_windows-x64_bin.exe

Penn Engineering   Property of Penn Engineering |

11

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

Penn Engineering   Property of Penn Engineering |

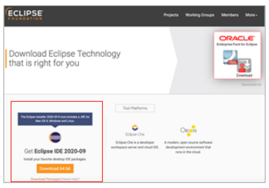
12

0:00

L'indice della comunità di programmazione TIOBE è un indicatore della popolarità dei linguaggi di programmazione. Potrebbe essere usato per verificare se le tue capacità di programmazione sono aggiornate o da fare una decisione su quali linguaggi di programmazione da utilizzare quando si avviano nuovi progetti. Le valutazioni si basano sul numero di ingegneri qualificati in tutto il mondo, corsi e fornitori di terze parti, e calcolato in base ai motori di ricerca più diffusi. L'indice viene aggiornato una volta al mese. Ecco i primi 10 linguaggi di programmazione da una recente istantanea dell'indice TIOBE. Per quanto riguarda i punti salienti generali, Java è il più popolare linguaggio di programmazione in primo luogo. Python è al terzo posto e sta diventando sempre più popolare. Vai a questo URL per l'indice più recente.

Installing & Configuring Eclipse

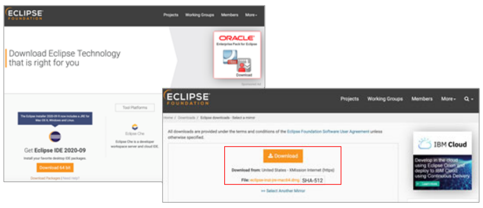
- Install Eclipse via https://www.eclipse.org/downloads/
  - Scroll down to get the latest version of Eclipse


ECLIPSE FOUNDATION
Projects
Working Groups
Members
More
Download Eclipse Technology that is right for you
ORACLE Enterprise Pack for Eclipse
Download
Tool Platforms
Get Eclipse IDE 2020-09
Install your favorite desktop IDE packages.
Download 64 bit
Eclipse Che
Eclipse Che is a developer workspace server and cloud IDE.
A modern, open source software development environment that runs in the cloud.

13

Installing & Configuring Eclipse

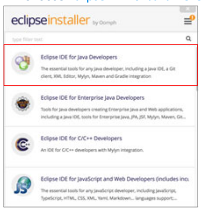
- Install Eclipse via https://www.eclipse.org/downloads/
  - Click to download the latest version of the IDE for your OS



14

Installing & Configuring Eclipse

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



Eclipse IDE for Java Developers
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle integration

Eclipse IDE for Enterprise Java Developers
Tools for Java developers creating Enterprise Java and Web applications, including a Java IDE, tools for Enterprise Java, JPA, JSF, Mylyn, Maven, Git...

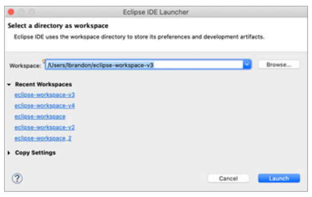
Eclipse IDE for C/C++ Developers
An IDE for C/C++ developers with Mylyn integration.

Eclipse IDE for JavaScript and Web Developers (includes inc
The essential tools for any JavaScript developer, including JavaScript, TypeScript, HTML, CSS, XML, Yaml, Markdown... languages support;...

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"



Eclipse IDE Launcher

Select a directory as workspace

Eclipse IDE uses the workspace directory to store its preferences and development artifacts.

Workspace: /Users/lbrandon/eclipse-workspace-v3 Browse...

Recent Workspaces

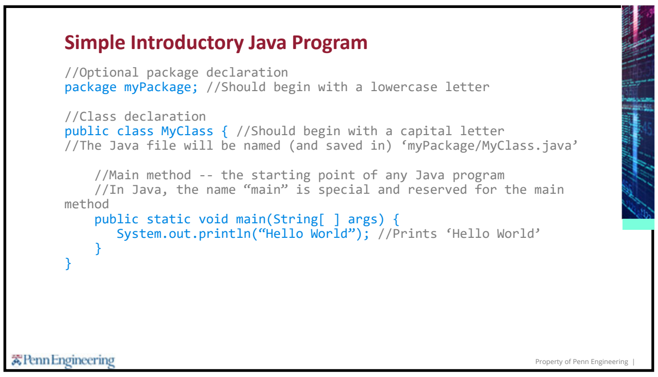
eclipse-workspace-v3
eclipse-workspace-v4
eclipse-workspace
eclipse-workspace-v2
eclipse-workspace 2

Copy Settings

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

Java Language

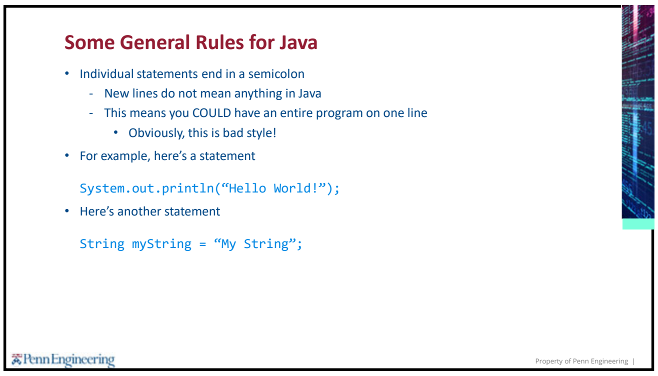


Simple Introductory Java Program

```
//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’
    }
}
```

Penn Engineering

19



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

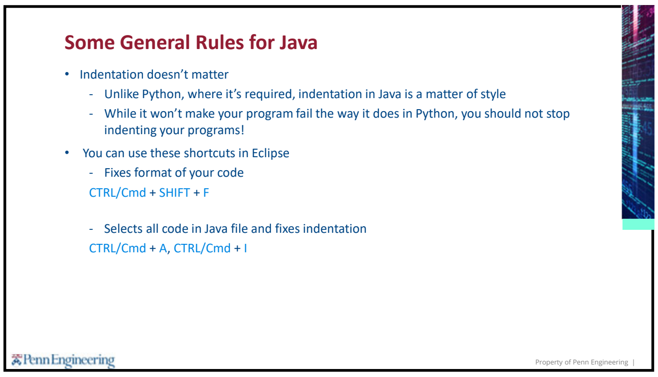
```
System.out.println(“Hello World!”);
```

- Here’s another statement

```
String myString = “My String”;
```

Penn Engineering

20



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

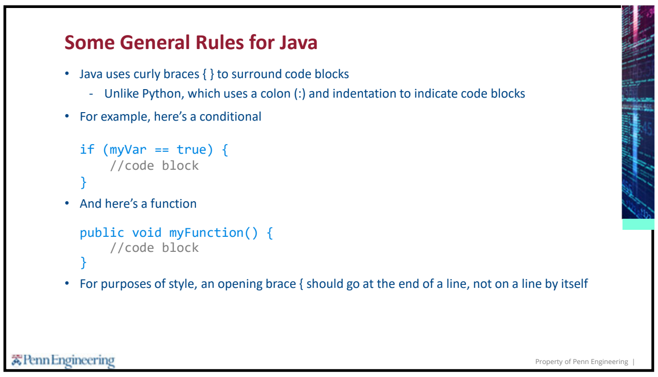
    CTRL/Cmd + SHIFT + F

  - Selects all code in Java file and fixes indentation

    CTRL/Cmd + A, CTRL/Cmd + I

Penn Engineering

21



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

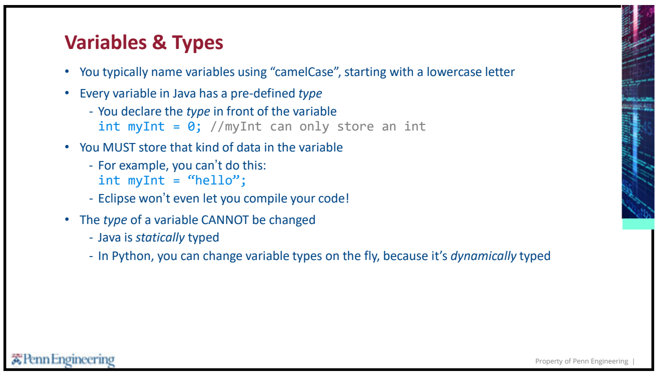
```
if (myVar == true) {
    //code block
}
```

- 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

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

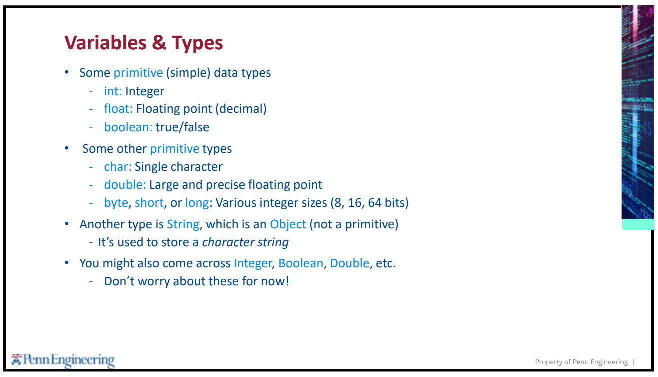
```
int myInt = 0; //myInt can only store an int
```

- 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

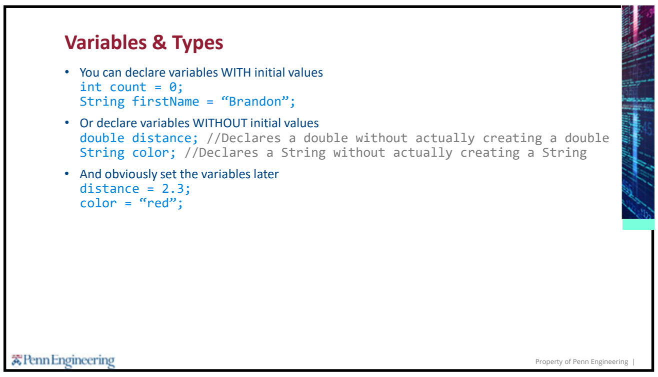
23



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!

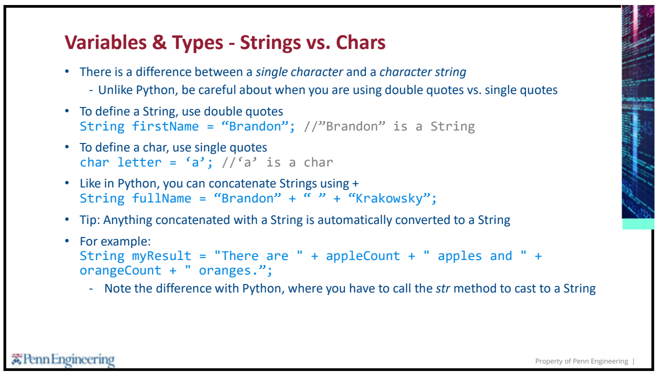
24



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";
```

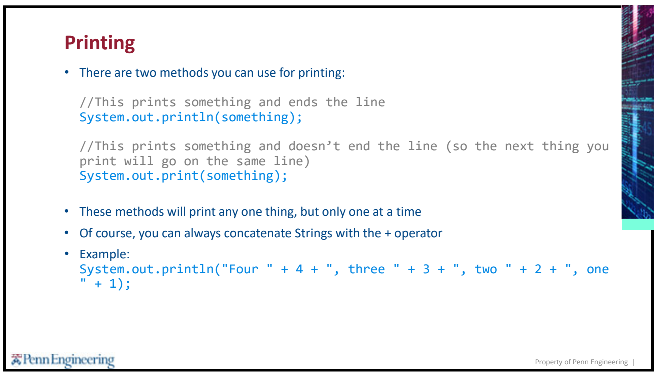
25



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

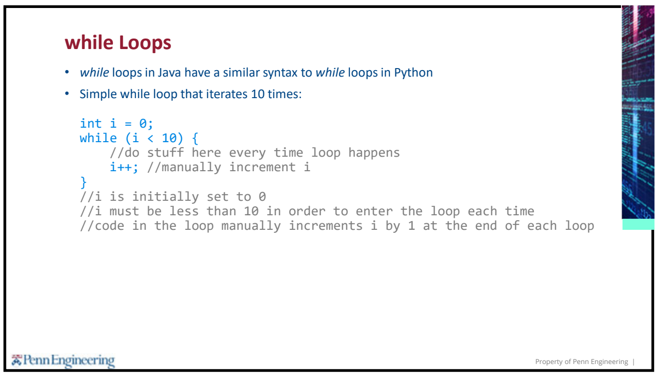
26



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);
```

27

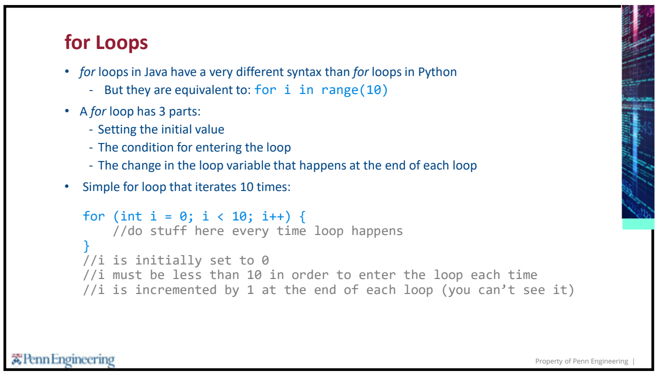


while Loops

- *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
```

28

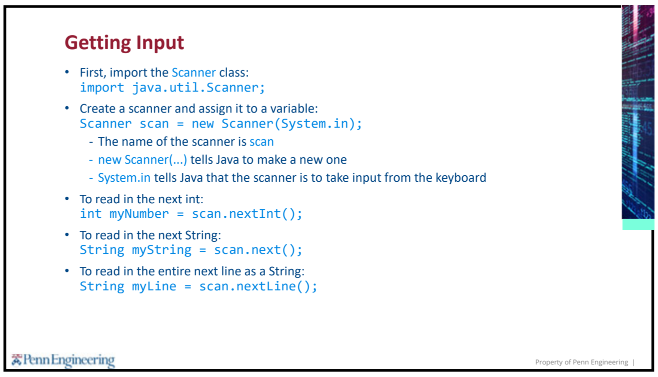


for Loops

- *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)
```

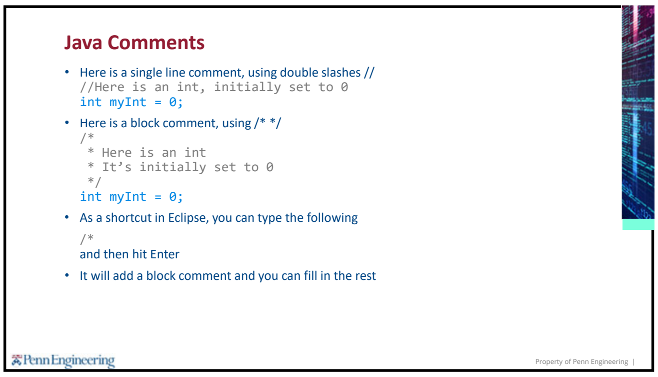
29



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();`

30



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

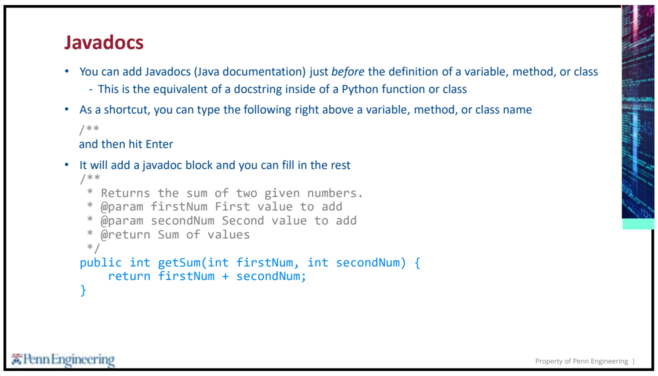
```
/*
```

and then hit Enter

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

Penn Engineering

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

```
/**
```

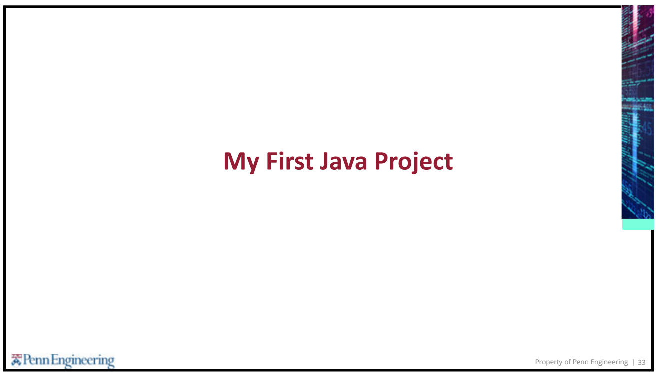
and then hit Enter

- 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;
}
```

Penn Engineering

32



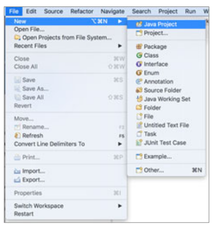
My First Java Project

Penn Engineering

33

My First Java Project

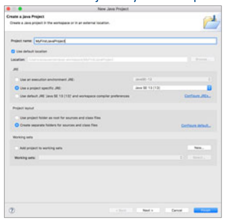
- In Eclipse, go to “File” → “New” → “Java Project”



34

My First Java Project

- Create a Java Project in your workspace


New Java Project
Create a Java Project
Create a Java project in the workspace or in an external location.
Project name: MyFirstJavaProject
[✓] Use default location
JRE
Use an execution environment JRE: JavaSE-13
[●] Use a project specific JRE: Java SE 13 [13]
Use default JRE 'Java SE 13 [13]' and workspace compiler preferences
Project layout
Use project folder as root for sources and class files
[●] Create separate folders for sources and class files
Working sets
Add project to working sets
Working sets:

Provide a Project name
- Project names should be capitalized

Use the default location

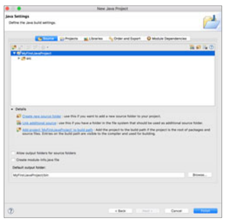
Use the default JRE and project layout

Click “Next”

35

My First Java Project

- Define the compilation/build settings


New Java Project
Java Settings
Define the Java build settings.
MyFirstJavaProject
src
Details
Create new source folder: use this if you want to add a new source folder to your project.
Link additional source: use this if you have a folder in the file system that should be used as additional source folder.
Add project 'MyFirstJavaProject' to build path: Add the project to the build path if the project is the root of packages and source files. Entries on the build path are visible to the compiler and used for building.
Allow output folders for source folders
Create module-info.java file
Default output folder:
MyFirstJavaProject/bin

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

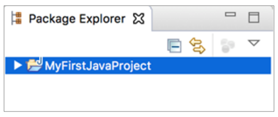
Use the default output folder

Click “Finish”

36

My First Java Project

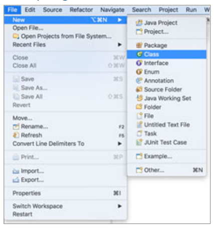
- The project will appear in the Package Explorer on the left hand side in the IDE



PennEngineering

My First Java Project

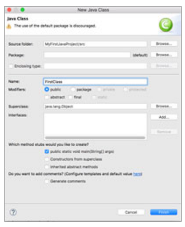
- In Eclipse, go to “File” → “New” → “Class”



PennEngineering

My First Java Project

- Create a Java Class in your Java Project


New Java Class

Java Class

The use of the default package is discouraged.

Source folder: MyFirstJavaProject/src

Package: (default)

Enclosing type:

Name: FirstClass

Modifiers: [✓ public] package private protected

abstract final static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

[✓ public static void main(String[] args)]

Constructors from superclass

Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

Generate comments

Provide a Name
- Class names should be capitalized

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

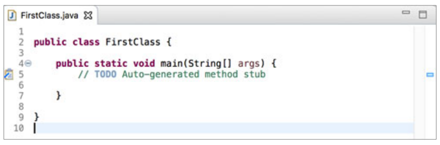
Make sure Inherited abstract methods IS NOT checked

Click “Finish”

PennEngineering

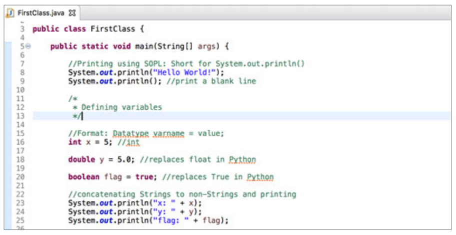
My First Java Project

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



```
FirstClass.java

public class FirstClass {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}
```

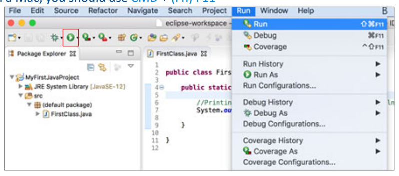
My First Java Project



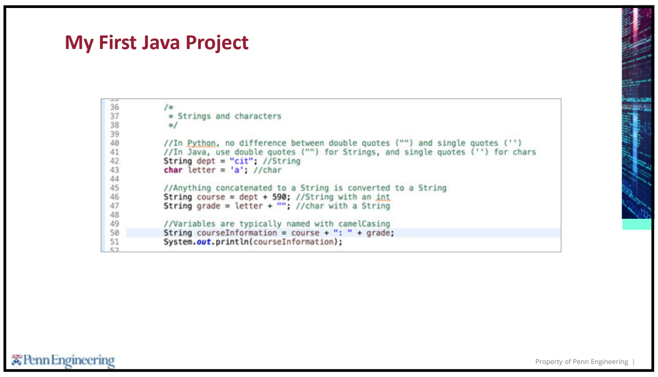
```
FirstClass.java
public class FirstClass {

    public static void main(String[] args) {

        //Printing using SOPL: Short for System.out.println()
        System.out.println("Hello World!");
        System.out.println(); //print a blank line

        /*
         * Defining variables
         */

        //Format: Datatype varname = value;
        int x = 5; //int

        double y = 5.0; //replaces float in Python

        boolean flag = true; //replaces True in Python

        //concatenating Strings to non-Strings and printing
        System.out.println("x: " + x);
        System.out.println("y: " + y);
        System.out.println("flag: " + flag);
```

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



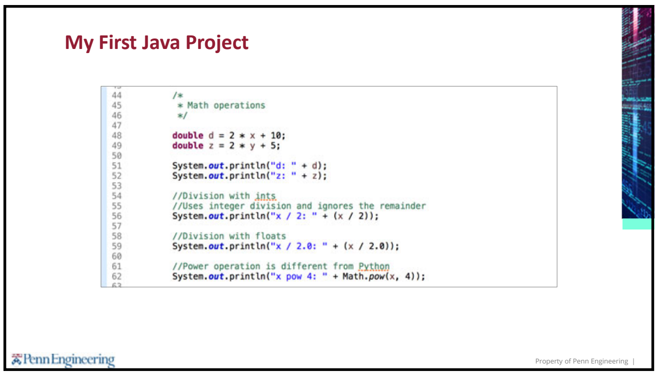
```
public class Firs
    public static
        //Printin
        System.ou
    }
}
```



My First Java Project

```java
/*
 * Strings and characters
 */

//In Python, no difference between double quotes ("") and single quotes ('')
//In Java, use double quotes ("") for Strings, and single quotes ('') for chars
String dept = "cit"; //String
char letter = 'a'; //char

//Anything concatenated to a String is converted to a String
String course = dept + 590; //String with an int
String grade = letter + ""; //char with a String

//Variables are typically named with camelCasing
String courseInformation = course + ": " + grade;
System.out.println(courseInformation);
```

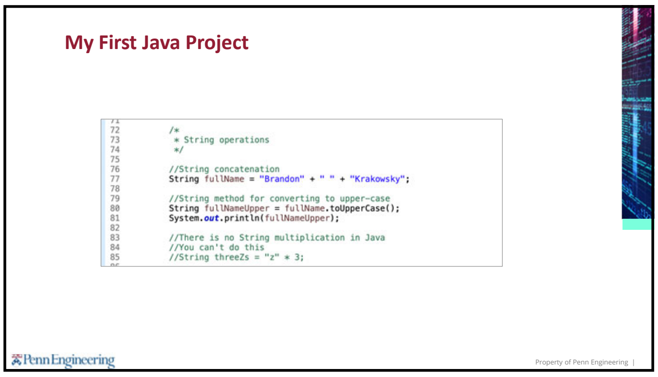
43



My First Java Project

```java
/*
 * Math operations
 */

double d = 2 * x + 10;
double z = 2 * y + 5;

System.out.println("d: " + d);
System.out.println("z: " + z);

//Division with ints
//Uses integer division and ignores the remainder
System.out.println("x / 2: " + (x / 2));

//Division with floats
System.out.println("x / 2.0: " + (x / 2.0));

//Power operation is different from Python
System.out.println("x pow 4: " + Math.pow(x, 4));
```

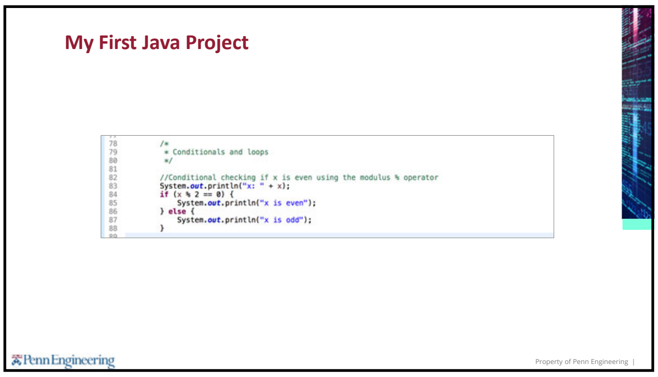
44



My First Java Project

```java
/*
 * String operations
 */

//String concatenation
String fullName = "Brandon" + " " + "Krakowsky";

//String method for converting to upper-case
String fullNameUpper = fullName.toUpperCase();
System.out.println(fullNameUpper);

//There is no String multiplication in Java
//You can't do this
//String threeZs = "z" * 3;
```

45



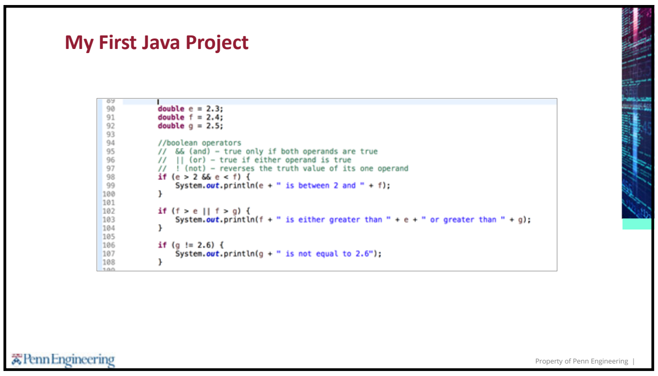
My First Java Project

```java
/*
 * Conditionals and loops
 */

//Conditional checking if x is even using the modulus % operator
System.out.println("x: " + x);
if (x % 2 == 0) {
    System.out.println("x is even");
} else {
    System.out.println("x is odd");
}
```

Penn Engineering

Property of Penn Engineering |

46



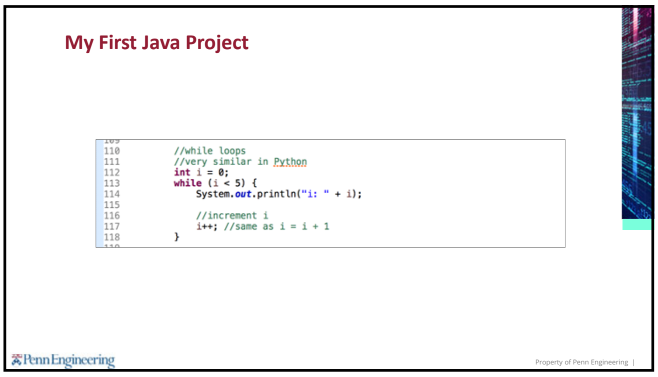
My First Java Project

```java
double e = 2.3;
double f = 2.4;
double g = 2.5;

//boolean operators
//  && (and) - true only if both operands are true
//  || (or) - true if either operand is true
//  ! (not) - reverses the truth value of its one operand
if (e > 2 && e < f) {
    System.out.println(e + " is between 2 and " + f);
}

if (f > e || f > g) {
    System.out.println(f + " is either greater than " + e + " or greater than " + g);
}

if (g != 2.6) {
    System.out.println(g + " is not equal to 2.6");
}
```

Penn Engineering

Property of Penn Engineering |

47



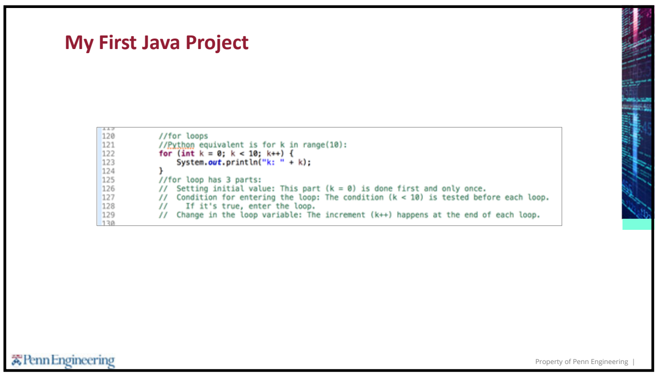
My First Java Project

```java
//while loops
//very similar in Python
int i = 0;
while (i < 5) {
    System.out.println("i: " + i);

    //increment i
    i++; //same as i = i + 1
}
```

Penn Engineering

Property of Penn Engineering |

48



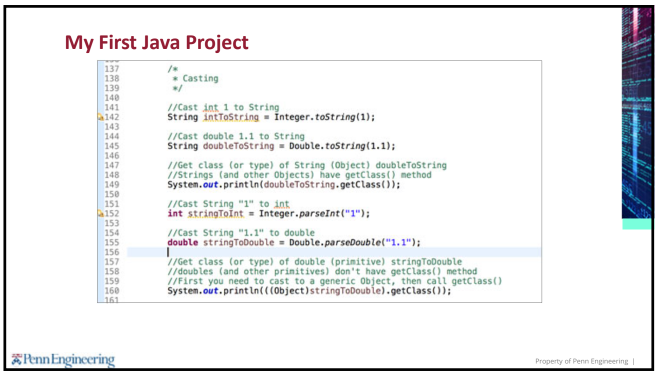
My First Java Project

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

Penn Engineering

Property of Penn Engineering |

49



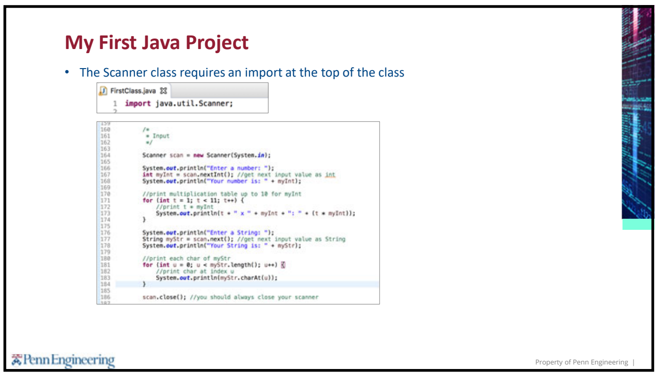
My First Java Project

```java
/*
 * Casting
 */

//Cast int 1 to String
String intToString = Integer.toString(1);

//Cast double 1.1 to String
String doubleToString = Double.toString(1.1);

//Get class (or type) of String (Object) doubleToString
//Strings (and other Objects) have getClass() method
System.out.println(doubleToString.getClass());

//Cast String "1" to int
int stringToInt = Integer.parseInt("1");

//Cast String "1.1" to double
double stringToDouble = Double.parseDouble("1.1");

//Get class (or type) of double (primitive) stringToDouble
//doubles (and other primitives) don't have getClass() method
//First you need to cast to a generic Object, then call getClass()
System.out.println(((Object)stringToDouble).getClass());
```

Penn Engineering

Property of Penn Engineering |

50



My First Java Project

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

FirstClass.java

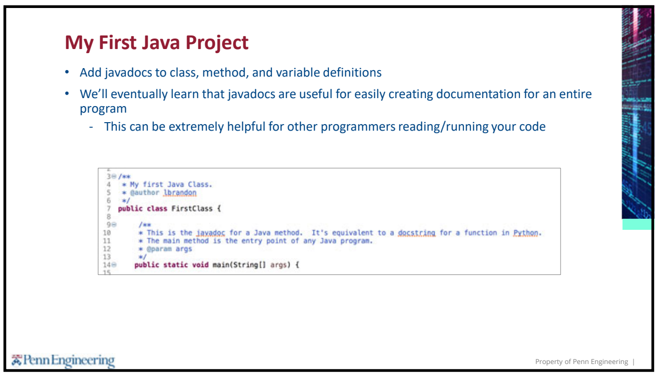
```java
import java.util.Scanner;
```

```java
/*
 * Input
 */

Scanner scan = new Scanner(System.in);

System.out.println("Enter a number: ");
int myInt = scan.nextInt(); //get next input value as int
System.out.println("Your number is: " + myInt);

//print multiplication table up to 10 for myInt
for (int t = 1; t < 11; t++) {
    //print t * myInt
    System.out.println(t + " x " + myInt + ": " + (t * myInt));
}

System.out.println("Enter a String: ");
String myStr = scan.next(); //get next input value as String
System.out.println("Your String is: " + myStr);

//print each char of myStr
for (int u = 0; u < myStr.length(); u++) {
    //print char at index u
    System.out.println(myStr.charAt(u));
}

scan.close(); //you should always close your scanner
```

Penn Engineering

Property of Penn Engineering |

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

```java
/**
 * 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) {
```

Property of Penn Engineering |