Variables and Data Types

Module 1 - 02

Day 1 Review

1. The File System
2. Bash Shell
3. GIT

Shell Basics

The **Root** directory is where the file system begins for the computer. All directories and files on the computer are in the root directory.

The **home directory** is where the file system that has been assigned to your user begins. The home directory can always be accessed by the ~ alias.

The **working directory** is the directory that is currently being accessed.

Shells are controlled using **commands** that tell the computer what you want it to do.

Many shell commands take **arguments**, which are extra parts of the command that change its behavior.

In Windows, the Bash Shell is accessed using the Git Bash application.

Basic Shell Commands

. is an alias for the current directory

.. is an alias for the parent directory of the current directory.

Command	Description
cd	change directory. ~ specifies the home directory. (cd ~)
pwd	Tells you where you are in the file system
ls ls -la	list all files and folders in a directory. -a (shows all files and folders) ls -a -l (shows details about the files and folders) ls - l. can be used together as ls -la In file details first letter is d for a directory and - for a file.
mkdir <dir name>	Creates a new directory
code <filename>	Creates a file then opens it in Visual Studio Code, or if the file exists it opens it.
touch <filename>	Creates an empty file
cp <file> <new location>	Copies a file
mv <file> <new location>	Moves or renames a file
cat	print contents of a file on the screen
rm rm -r	Removes a file or folder with -r (recursively remove all contents). -f (force) - make it happen no matter what.

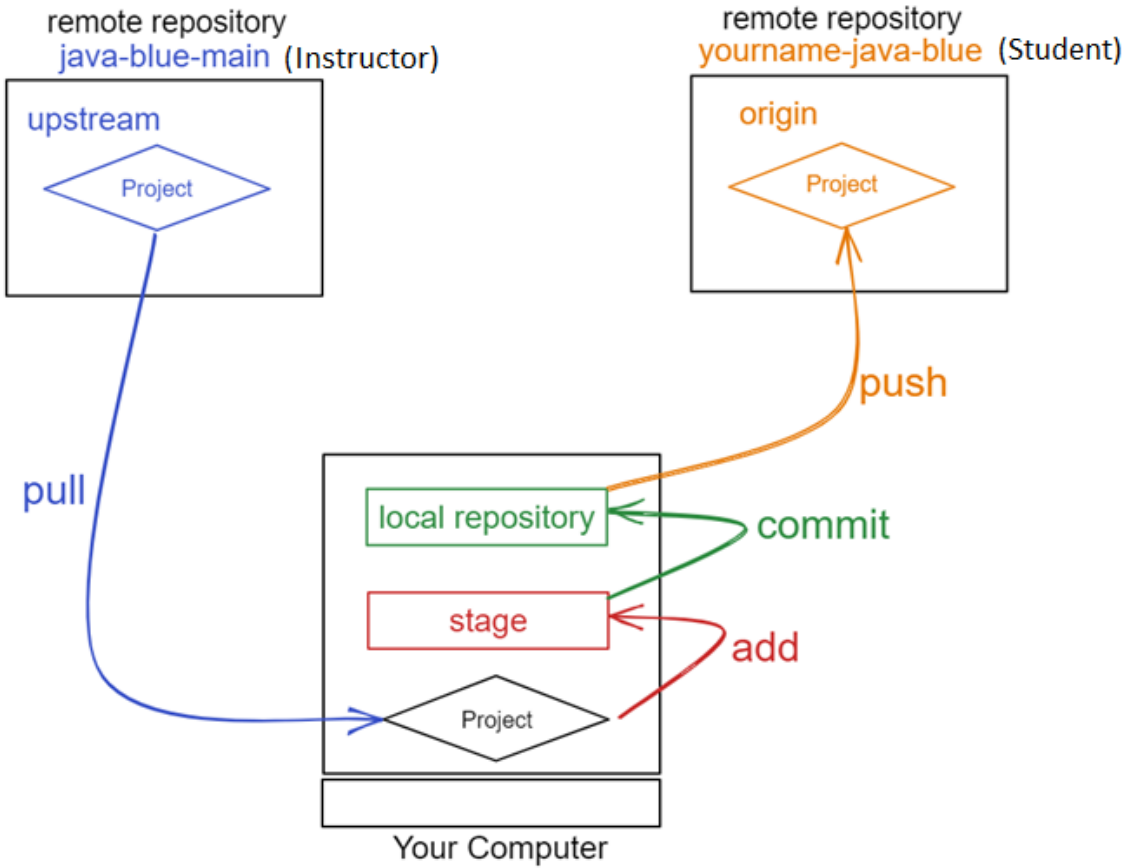
Remote Repositories

upstream
java-blue-main

Instructor repository used to get new materials and exercises. Can only pull.

origin
yourname-java-blue

Student repository used to backup and submit materials and exercises. Can push and pull.



Today’s Objectives

1. Introduction to Java
2. Hello World!
3. Introduction to the IntelliJ Integrated Development Environment (IDE)
4. Variables and Data Types
5. Expressions
6. Arithmetic Operators
7. Data Type Conversion
    a. Widening
    b. Narrowing
    c. Truncation

Introduction to Java

What is Java?

- Object Oriented
- Syntax Derived from C/C++
- Portable
  - Write Once, Run Anywhere
- Virtual Machine Interpreted
- Has a standard Library

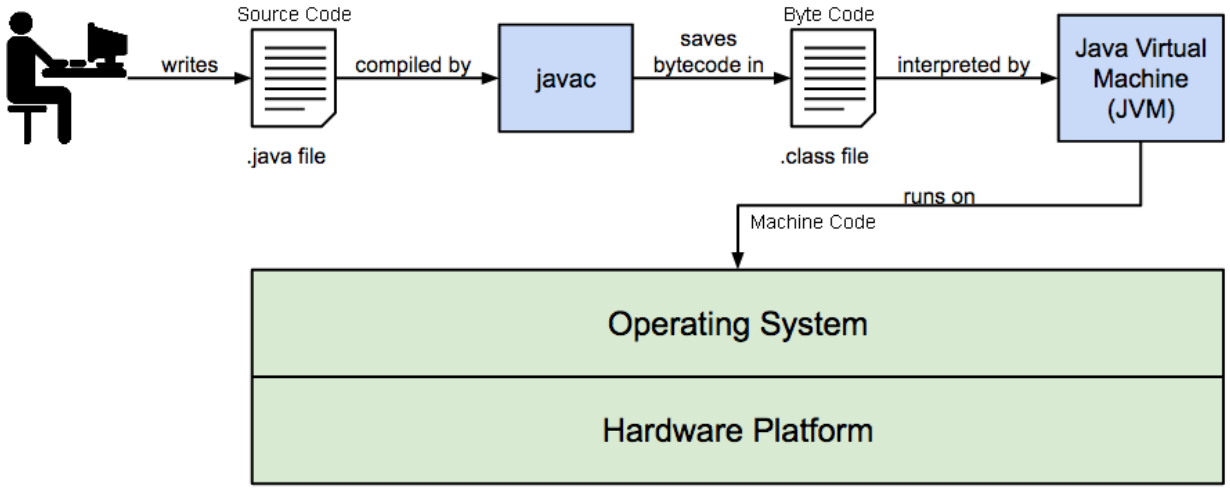
Common Interview Question:

What is the relationship between Java and JavaScript?

Answer: There is no relationship

Common Interview Question

Java Architecture



Source Code - code written by humans. *Stored in .java files.*

Byte Code - compiled code that can be read by the JVM independent of the Operating System (Windows, Mac OS, Linux, etc.). *Stored in .class files.*

Machine Code - compiled code that can be read by a specific operating system. *Created by the JVM when running.*

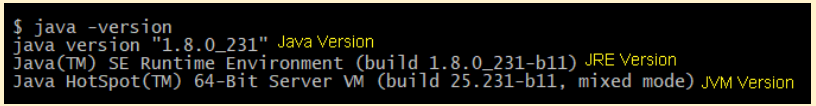
1. Programmer *writes* **Source Code** in **.java** files
2. ***javac*** *compiles* the source code into **Byte Code** in **.class** files
3. ***Java Virtual Machine (JVM)*** *interprets* **Byte Code** into **Machine Code** that can be understood by the computer's operating system.

JRE vs JDK

1. Java Runtime Environment (JRE)
    a. Allows execution of Java applications.
    b. Installed by default on most OSs
    c. Contains the JVM
    d. Does not contain libraries and tools for development
    e. Used by home users
2. Java Development Kit (JDK)
    a. Includes development library and tools
    b. Includes the javac compiler
    c. Includes the JRE and JVM
    d. Used only by developers

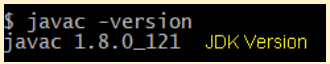
Java Version Check

In terminal: java -version



Java JDK Version Check

In terminal: javac -version



Hello World!

1. Open Terminal
2. `$ cd ~/source`
3. `$ mkdir HelloWorld`
4. `$ cd HelloWorld`
5. `$ code HelloWorld.java`
6. Write the source code in the file
7. File -> Save
8. Compile the Source to Byte Code using *javac*

   `$ javac HelloWorld.java`

1. Run the program using the *JVM*

   `$ java HelloWorld`

Source Code

```
public class HelloWorld {

            public static void main(String[]
args) {
                        // Prints out Hello World
                        System.out.println("Hello
World");
            }

}
```

The class name, **HelloWorld**, must match the java file name, **HelloWorld.java**. Including case.

Parts of Hello World

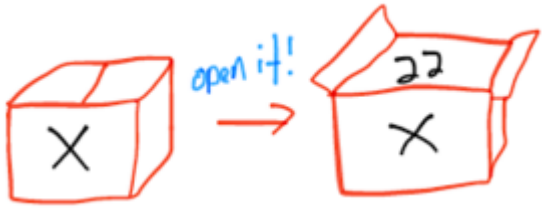
Code	Description
`public class HelloWorld`	Java programs are made up of one or more classes which hold program code. All Java code must be in a class.
`public static void main(String[] args)`	The entry point to the program. When we run the program this contains the code that is executed.
`{...}`	Encloses a set of commands. For each opening brace there is a closing brace
`// Prints out Hello World`	Comment code that is not executed and provides information to the developer
`System.out.println("Hello World");`	Command to write the text, Hello World, to the console. In this case the Terminal.

Introduction to IntelliJ

IntelliJ is an IDE (Integrated Development Environment)

- Organizes code into projects
- Provides immediate feedback on syntax errors
- Assists in code with Intellisense
- Performs many routine tasks and supplies common code snippets
- Allows us to suspend a program and step through using a **Debugger**

Variables



- **Definition**
  Storage container paired with a symbolic name or identifier. It holds some known or unknown amount of information referred to as a value. Variables have a Data Type that defines what type of data that variable can hold.
- **Parts:**
  - Data Type
  - Name
  - Value

A variable allows us to set and hold a value to be used later. They are also used to make code more readable.

Java Primitive Data Types

Data Type	Size (Bytes)	Usage	Example
byte	1	whole numbers -128 to 127	41
char	2	Unicode Characters codes - representing a single letter, number, or symbols. Supports characters from most languages.	‘a’, ‘Φ’
boolean	1	true / false	true
short	2	whole numbers -32,768 to 32,767	-27, 10822, 1
int	4	positive or negative whole numbers +/- 2,147,483,647)	12024, -500042, 1
float	4	floating point numbers with a precision of 7 digit	3.14
double	8	double floating point numbers with a precision of 15 digits	3.14159265358979
long	8	really big whole numbers +/- 9,223,372,036,854,775,807	5267503443, 2, -26

Variable Names

Rules

1. follow camelCase (first character lowercase after that every new word the first character is upper case)
2. Always start with a letter and contain numbers and letters
    a. Can also contain or start with $ but only used by generated code
    b. Can also contain or start with _ but discouraged
3. Can be any length - favor description over length
    a. Isolation test - in isolation can you tell from the name what the variable represents
4. Boolean variables should start with a word that defines as a true/false question or statement (is, has, does, etc.)

Oracle Variable Naming Documentation

Variable Names Continued

Bad Variable Names

name1, name2, name3

number

x

cost

Good Variable Names

numberOfStudents

averageCostOfGasInDollars

checkingAccountBalance

secondsPerMinute

totalCostInCents

Creating a Variable

Created in 2 Parts

1. **Declaration** - defines the **Data Type** and **Name**

```
int numberOfStudentsInClass;
```

1. **Assignment** - sets a value

```
numberOfStudentsInClass = 20;
```

Declaration and Assignment can occur as 2 lines of code:

```
int numberOfStudentsInClass;
numberOfStudentsInClass = 20;
```

Or as a single line of code:

```
int numberOfStudentsInClass = 20;
```

String Data Type

- Holds characters “Hello world”
- Can be assigned with a literal string in double quotes

```
String name = “John Matrix”;
```

- Data Type name is capitalized
- Can contain characters like new line, tab, and double quotes, which must identified with escape characters
  - \n → new line
  - \t → tab
  - \” → double quote
  - Example: `String name = “ \”Let off some steam!\”\n\t-John Matrix”;`

Prints as:
```
“Let off some steam!”
        - John Matrix
```

Arithmetic Operators

Multiplicative : * (multiplication), / (division), % (modulus - returns the remainder from division)

Additive: + (addition), - (subtraction)

Assignment: = (assigns a value)

Order of Operation

Similar to PEMDAS order of operation of mathematics.

1. Parentheses
2. Multiplicative (multiplication, division, and modulus)
3. Addition (addition and subtraction)
4. Assignment

When unsure set Order of operation with Parentheses

Example:
5 + 2 * 2 results in 9
(5 + 2) * 2 results in 14

Float and Double Precision

Significant Digit - the point of precision when a decimal digit is “good enough” to solve the problem.

Simple Explanation

Modern 64bit computers can store 56 significant digits, but this is the number of digits in binary and not base-10 Arabic Numbers! Decimals are also stored in scientific notation and float and double don’t understand recursive numbers (ones with repeating digits like 10 / 3 == 3.3333333333~. This often causes rounding errors when doing math with floating point numbers.

Often we don’t care because the significant digit is small enough not to matter. For example, if 10/3 then an answer of 3.33 may be “good enough”, but what if we care about absolute precision like in a currency or a scientific calculation?

A more detailed explanation can be seen in this short Video

Type Conversion

The Problem: Data types are different sizes. For example, a long type can hold a very large number, but int can only hold a number approximately 2.1billion in size. If we have a long x that contains a 10, then we know it will fit in an int, but how do we use it as an int?

Solution: ***Casting*** allows us to tell Java to treat a variable as a different data type, provided they both hold the same type of data. To ***cast*** the type we want Java to treat the data as is added before it in parentheses.

```
long x = 10;
    int y = (int) x;
```

Literal numbers in Java have a default data type. All whole number literals default to `int`, and all floating point numbers default to `double`.

For example, `float x = 2.0;` will result in an error, since `2.0` defaults to a `double`.

There are shorthand casting available for literal numbers. `L` tells Java you want the literal to be a `long` and `F` tells Java to treat the number as a `float`.

```
float x = 2.0F;
long y = 1000L;
```

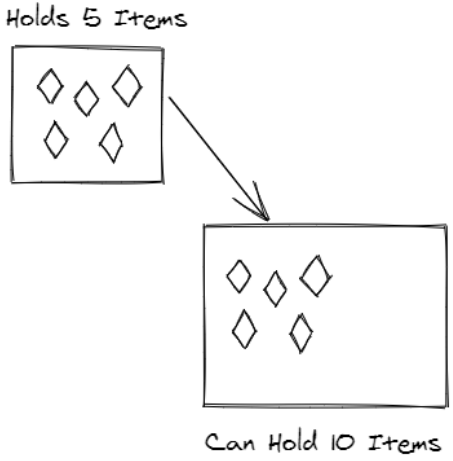
Widening

Cast (converting) from a smaller data type into a larger data type

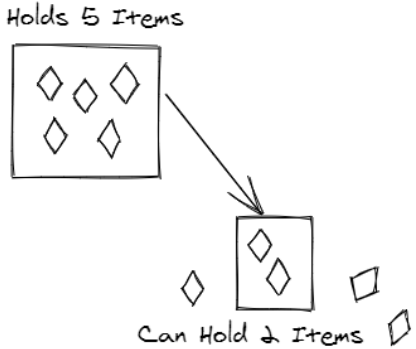
int → long
float → double

Implicitly Cast (converted (cast) automatically)

```
int x = 10;

long y = x;
```



Narrowing



Cast (converting) from a larger data type into a smaller data type

long → int
double → float
int → short

Explicitly Cast (not automatic and code must indicate that it should happen and the data type to convert (cast) it to)

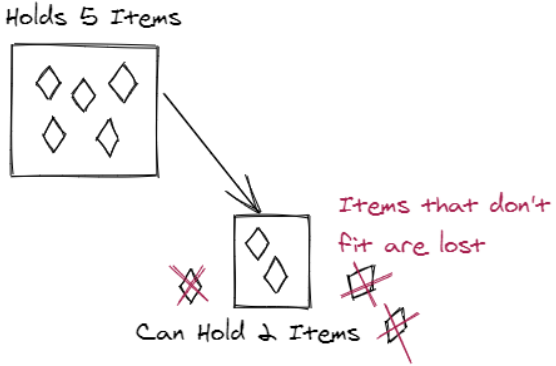
```
long x = 10;

int y = (int) x;
```

Truncation

When a narrowed value doesn't "fit" into the new data type, the "extra" is ignored.

double x = 5 / 2; (double = int / int)

5 / 2 = double 2.5 → int 2



In Review

1. Variables
2. Creating a variable.
3. Casting primitive data types
4. Widening
5. Narrowing
6. Truncation

Module 1- (Deck B)

Variables and Data Types

History of Java

Java is an object oriented language (you will learn what this means later!) developed by Sun Microsystems in 1995. It was originally created by James Gosling. Sun Microsystems was acquired by Oracle Corporation in 2010.

It it is one of the most widely used languages. It consistently ranks in the top 4 in terms of popularity. (Source: https://stackify.com/popular-programming-languages-2018/ and https://www.geeksforgeeks.org/top-10-programming-languages-that-will-rule-in-2021/).

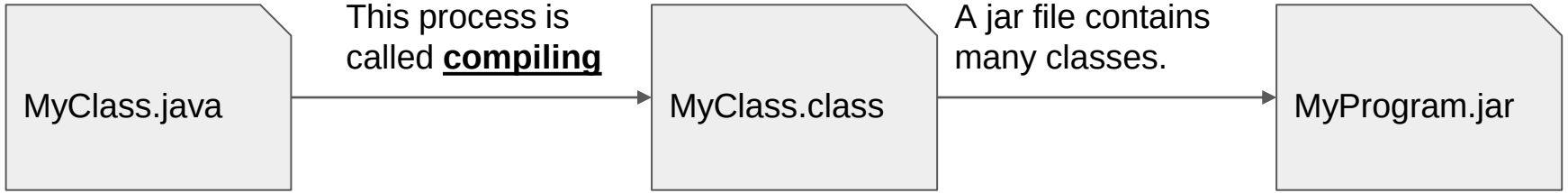
GitHub Language Rankings, 2018-2020

Language	2020 Ranking	2019 Ranking	2018 Ranking
JavaScript	1	1	1
Python	2	2	3
Java	3	3	2
TypeScript	4	7	4
C#	5	5	6
PhP	6	4	4

Java Features

- It shares similar syntax with C/C++
- It can be used to create desktop, mobile, and web applications.
- Unlike other languages, Java is not run natively on a given device, it is instead executed in a Java Virtual Machine (JVM) / Java Runtime Environment (JRE).... think of this as a virtual computer running inside your computer.
- Advantages of this model:
  - Oracle (not you ☺) is responsible for the lifecycle of the JRE / JVM.
  - Developers are therefore freed from the idiosyncrasies of each individual platform! (i.e. pc's, macs, mobile devices, refrigerators, etc)
  - WORA – write once, run anywhere

Java Development Workflow



Compiling is the process by which source code (in this case the file MyClass.java) is transformed into a language the computer can understand.

In the past, the command **javac** was used to compile code, since the advent of modern development tools, manually typing this command is no longer needed.

Java Bytecode Example

```
outer:
for (int i = 2; i < 1000; i++) {
    for (int j = 2; j < i; j++) {
        if (i % j == 0)
            continue outer;
    }
    System.out.println (i);
}
```

This is what we will write

A Java compiler might translate the Java code above into byte code as follows, assuming the above was put in a method:

```
0:   iconst_2
1:   istore_1
2:   iload_1
3:   sipush  1000
6:   if_icmpge       44
9:   iconst_2
10:  istore_2
11:  iload_2
12:  iload_1
13:  if_icmpge       31
16:  iload_1
17:  iload_2
18:  irem
19:  ifne    25
22:  goto    38
25:  iinc    2, 1
28:  goto    11
31:  getstatic       #84; // Field java/lang/System.out:Ljava/io/PrintStream;
34:  iload_1
35:  invokevirtual   #85; // Method java/io/PrintStream.println:(I)V
38:  iinc    1, 1
41:  goto    2
44:  return
```

This is what the java translates it to

Source: https://en.wikipedia.org/wiki/Java_bytecode

Java Program Structure

- The main unit of organization in Java is a class. Here is a simple example:

  This file must be called MyClass.java. Its contents are as follows:

```
public class MyClass {

        public static void main(String[] args) {

            int i = 1;
        }
}
```

This must match the file name. It is the name of the class.

This a method called main… main is a special method that determines what gets run when the program executes

This is a variable called i, which currently stores a value of 1.

All java statements end with a semicolon.

Hello World Program

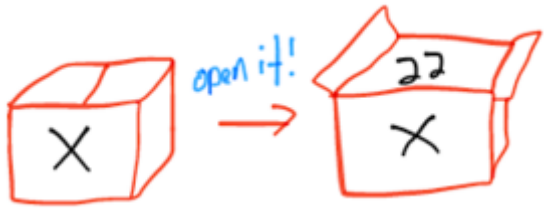
```
package com.techelevator;

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

Java Variables

- A variable is a representation for something that might change.

  Consider the math formula: $c^2 = a^2 + b^2$

  We don’t know for a fact what a, b, or c are, they can take different values depending on the situation. Therefore, a, b, and c are like containers that could take on different values at different times. These containers are called **variables**.



Data Types

- In order for computer to store data, must know what type it is holding
- All data is represented by bits
  - Bit is a switch
  - 2 states – on or off
- Byte
  - Amount of computer storage needed to store 1 character
  - 8 bits
- ASCII
  - American Standard Code for Information Interchange
  - 1 Byte – 256 different combinations
- Unicode
  - Universal Character encoding
  - 2 Bytes – 65,536 different combinations (16 bit)
  - 32 bits also available

Java Variables: Declaring and Assigning Values.

- This is what a java variable declaration looks like:

  int i = 0;

  Here, we have declared a variable called i of type integer, we have also given it an initial value of zero. Assigning values is accomplished using equals (=), the assignment operator.

- Consider this:

```
int i;
i = 1;
```

  Here, we have declared a variable called i of type integer, but then gave it the value 1.

Java Variables: Data Types.

Data Type	Description	Example
int	Integers (whole numbers)	int i = 1;
double	Decimals	double x = 3.14;
float	Also decimals, but older format. Avoid, use doubles instead.	float x = 3.14f; // Note that to declare a float you must explicitly state that the number is a float by appending the f.
char	One character. **Note the single quotes**.	char myChar = ‘a’;
boolean	True or false	boolean isItTurnedOn = true; boolean paidBills = false;
String	Several characters. **Note the double quotes**.	String name = “Minnie”; String fullName = name + “ Mouse”;

Java Variables: Rules & Conventions

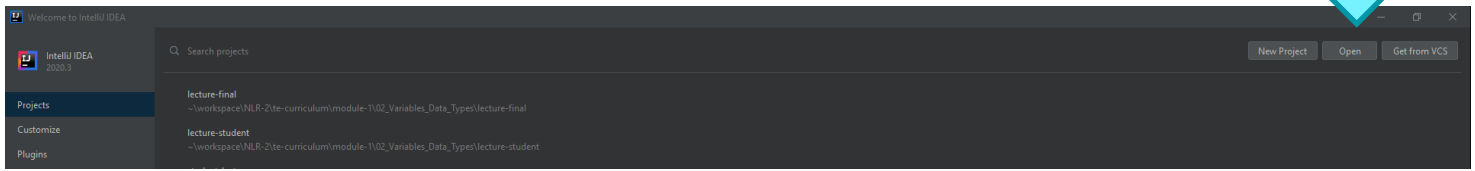
Conventions:

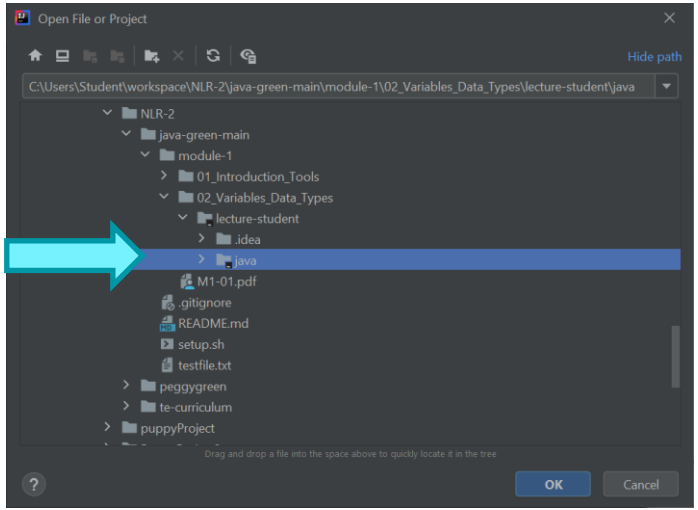
- Ideally, variables should be named using “Camel Case.” Examples of variable names: *playerOneScore*, *cityTemperature*, *shirtSize,* etc. (See the pattern?)
- Ideally, variables never start with an upper case.
- Variable names should be descriptive and of reasonable length.

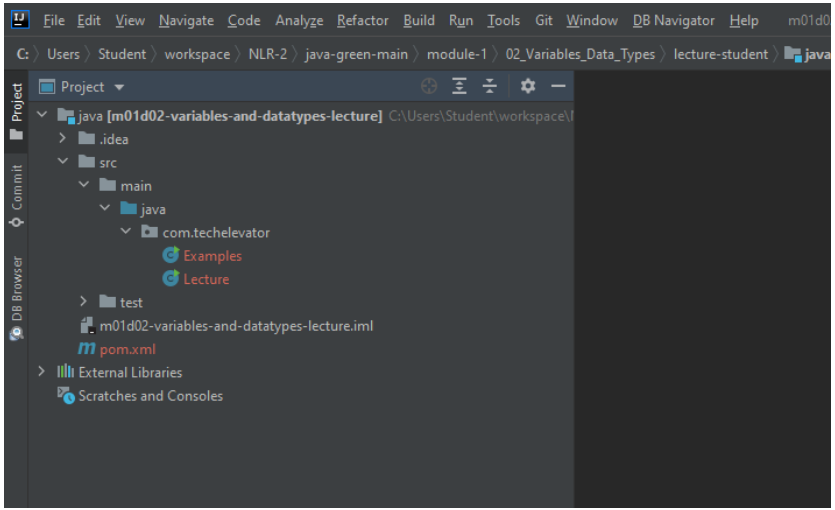
Rules:

- Variables can begin with an underscore (_), a dollar sign ($), or a letter.
- Subsequent characters can be letters, numbers, or underscores.
- Variable names cannot be the same as java keywords.

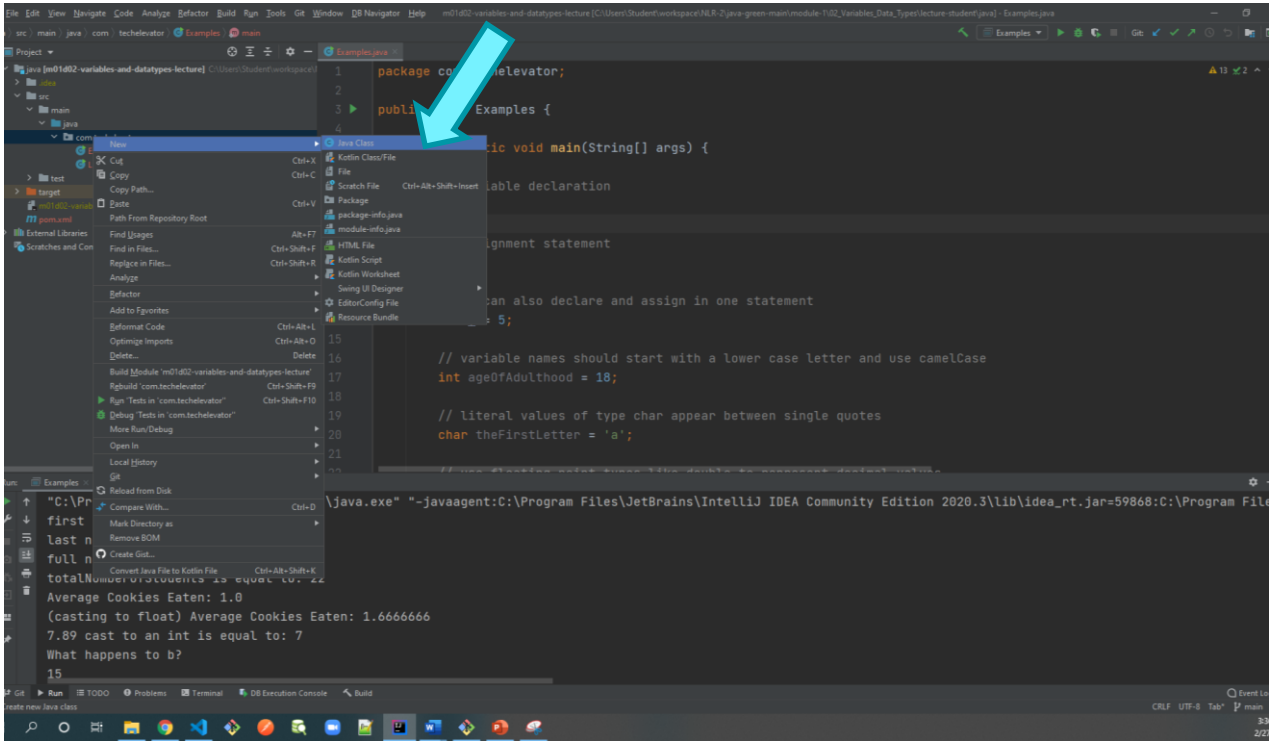
IntelliJ – our IDE (open existing project)

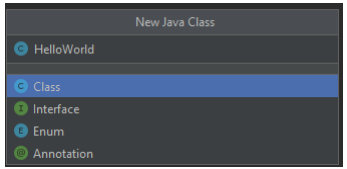






IntelliJ – our IDE!





Let’s code!

Working with Numbers: Basic Operators

- Math operators can be used to perform basic arithmetic between two numeric variables (From the previous slides, variables of type int, float, and double are examples).
- These are the basic operators: **+** (addition), **-** (subtraction), ***** (multiplication) **/** (division), **%** (modulus, aka remainder).
- The basic operators can be combined with the assignment operator to store result calculations, for example: **int i = 4 + 6**;

Working with Numbers: Order of Operations

For now, order of operations is almost the same as normal arithmetic: **P**lease **E**xcuse **M**y **D**ear **A**unt **S**ally (Mnemonic strategy to help students remember computational order. It stands for **Parentheses, Exponents, Multiplication, Division, Addition, and Subtraction**.)

Category	**Operators**
parenthesis	do first
multiplicative	* or / or %
additive	+ or -
assignment	=

Working with Numbers: Example 1

- Express the following English statement in Java: I paid for an item that cost $8.50 with a $10.00 bill, how much change would I get in return?

```
public class MyClass {

        public static void main(String[] args) {

                double price = 8.50;
                double payment = 10.00;
                double change = payment - price;
                System.out.println(change);
        }
}
```

Working with Numbers: Example 2

- Let’s try something a little bit more complex: We can convert degrees in Fahrenheit to Celsius using the following formula: $(T_F - 32) \times (5/9) = T_C$. How much is 98.6 degrees Fahrenheit in Celsius?

```
public class MyClass {

        public static void main(String[] args) {

                double tempInF = 98.6;
                double tempInC = (tempInF -32.0) * (5.0/9.0);
                System.out.println(tempInC);
        }
}
```

Working with Numbers: Type Conversion

ints, doubles and floats can be used together in the same statement, but Java will apply certain rules:

- Mixed mode expressions are automatically promoted to the higher data type (in this case, a double):

```
public class MyClass {

   public static void main(Strin

       int myInt = 4;
       double myDouble = 2.14;

       int firstAttempt = myInt - myDouble; // Won't work, Java will complain!
   }
}
```

The result of myInt and myDouble is promoted to a double, it will no longer fit in firstAttempt, which is a int.

Working with Numbers: Type Conversion

(continued from previous page)...

- We can overcome this problem by doing a cast:

  int secondAttempt = (int) (myInt - myDouble);

- We can also overcome this problem by making the variable secondAttempt a double:

  double secondAttempt = myInt - myDouble;

Working with Numbers: Type Conversion

Remember this problem?

```
public class MyClass {

        public static void main(String[] args) {

                double tempInF = 98.6;
                double tempInC = (tempInF -32.0) * (5/9);
                System.out.println(tempInC);
        }
}
```

Note that instead of 5.0/9.0, it is now 5/9, and when run, the result is 0.0. Using the rules we just discussed, can you figure out why?

Combining Strings

The plus sign can also be used with Strings:

```
public class MyClass {

        public static void main(String[] args) {

                String firstName = “Carl”;
                String lastName = “Jung”;

                String combinedName = lastName + “, ” + firstName;
                System.out.println(combinedName);
        }
}
```

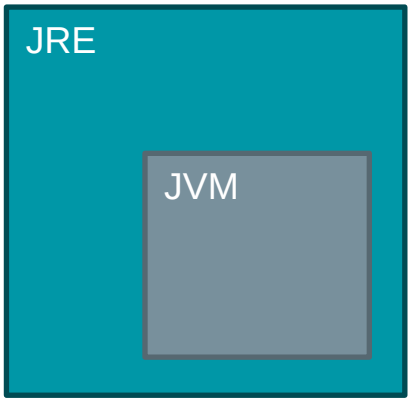
The following code will print ***Jung, Carl***. This process is known as **concatenation**.

Objectives - RECAP

- identify and explain fundamental concepts and components of Java Framework
- read and write code that uses variables
- declare a variable and assign a value to a variable
- know and use industry acceptable naming conventions for variables
- choose the appropriate primitive data type to represent different kinds of data in a program
- utilize arithmetic operators to form mathematical expressions
- explain the concept of data conversion (or casting), when it occurs, why it's used
- explain the purpose and use of literal suffixes and why they should be used
- code and test a simple Java using an Integrated Development Environment (IDE).
- perform simple tasks in an IDE such as:
    - Organizing code into projects
    - Understand feedback on syntax errors
    - Utilize the "intellisense" feature of an IDE to assist in code development

Objectives

- identify and explain fundamental concepts and components of Java Framework



JRE – Java Runtime Environment
JVM – Java Virtual Machine

.java files are source code files
.class files are byte code files

Objectives

- identify and explain fundamental concepts and components of Java Framework
- read and write code that uses variables
- declare a variable and assign a value to a variable

```
int i = 0;
```

```
int age;
age = 25;
```

Objectives

- identify and explain fundamental concepts and components of Java Framework
- read and write code that uses variables
- declare a variable and assign a value to a variable
- know and use industry acceptable naming conventions for variables

```
int numberOfKittens = 5;
```

```
String myFirstCat = “KC”;
```

Objectives

- identify and explain fundamental concepts and components of Java Framework
- read and write code that uses variables
- declare a variable and assign a value to a variable
- know and use industry acceptable naming conventions for variables
- choose the appropriate primitive data type to represent different kinds of data in a program

int numberOfKittens = 5;

Objectives

- identify and explain fundamental concepts and components of Java Framework
- read and write code that uses variables
- declare a variable and assign a value to a variable
- know and use industry acceptable naming conventions for variables
- choose the appropriate primitive data type to represent different kinds of data in a program
- utilize arithmetic operators to form mathematical expressions

```
int i = 7 / 2;
```

```
int i = 7 % 2;
```

```
int i = 7 % 2 * 3 + 5 / 2 - 4;
```

Objectives

- identify and explain fundamental concepts and components of Java Framework
- read and write code that uses variables
- declare a variable and assign a value to a variable
- know and use industry acceptable naming conventions for variables
- choose the appropriate primitive data type to represent different kinds of data in a program
- utilize arithmetic operators to form mathematical expressions
- explain the concept of data conversion (or casting), when it occurs, why it's used

```
public class Main {
  public static void main(String[] args) {
    double money = 9.78;
    int myDollars = (int) money; // Explicit casting: double to int

    System.out.println(money);
    System.out.println(myDollars);
  }
}
```

Objectives

- identify and explain fundamental concepts and components of Java Framework
- read and write code that uses variables
- declare a variable and assign a value to a variable
- know and use industry acceptable naming conventions for variables
- choose the appropriate primitive data type to represent different kinds of data in a program
- utilize arithmetic operators to form mathematical expressions
- explain the concept of data conversion (or casting), when it occurs, why it's used

Objectives

- identify and explain fundamental concepts and components of Java Framework
- read and write code that uses variables
- declare a variable and assign a value to a variable
- know and use industry acceptable naming conventions for variables
- choose the appropriate primitive data type to represent different kinds of data in a program
- utilize arithmetic operators to form mathematical expressions
- explain the concept of data conversion (or casting), when it occurs, why it's used
- code and test a simple Java program using an Integrated Development Environment (IDE).

Version: 2020.3.2
Build: 203.7148.57
25 January 2021