Variables and Data Types

Module 1 - 02

Day 1 Review

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

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
rmdir <dir-name> rm –r <dir-name>	Removes directory only if it is empty Removes folder with -r (recursively remove all contents). -f (force) - make it happen no matter what (no prompt) - **dangerous**.

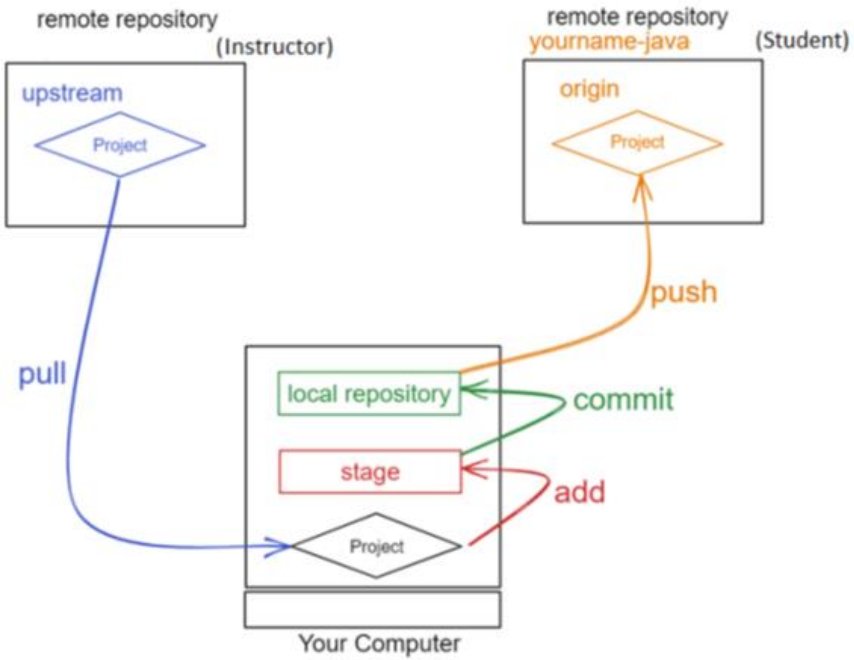
Remote Repositories

upstream
java-yellow-main

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

origin
yourname-java-yellow

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



```
git pull upstream main

git add –A
git commit –m “notes or comments”
git push origin main
```

Remember the Workflow!

1. Get new materials – (in your local repository's root directory.)

   ```
   git pull upstream main
   ```

2. complete the exercises
3. Add your changes to the stage

   ```
   git add -A
   ```

4. Commit the changes to your local repository

   ```
   git commit -m “comment”
   ```

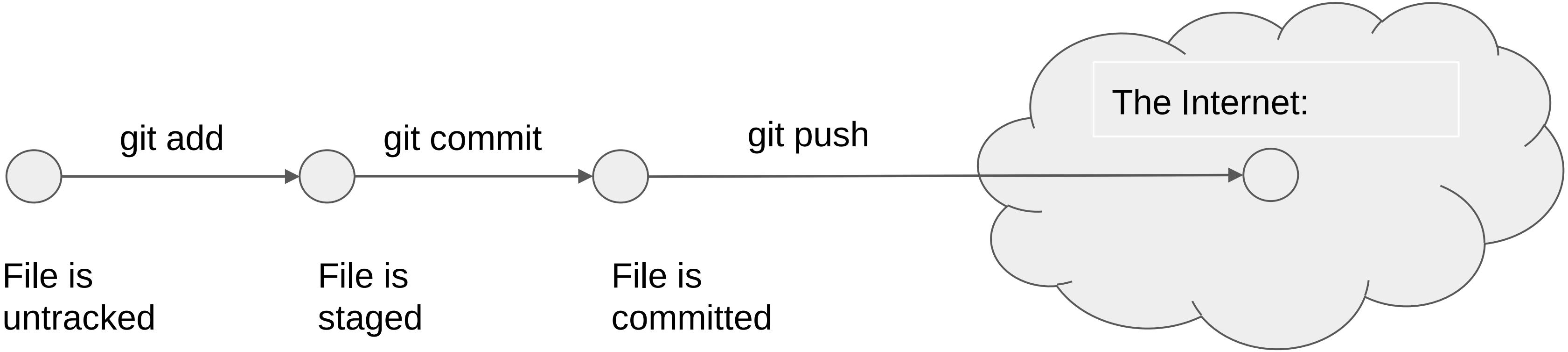
5. Push your changes to GitLab – (the remote repo).

   ```
   git push origin main
   ```

6. Submit your exercises in the LMS.

Source Control : Git Flow ()

- **git status**: See the current status of your files.
- **git add -A:** Stage any files you have changed.
- **git commit -m “Commit message”**: Commit files to your local repository
- **git push origin main**: Push committed changes to network repository.



git add

git commit

git push

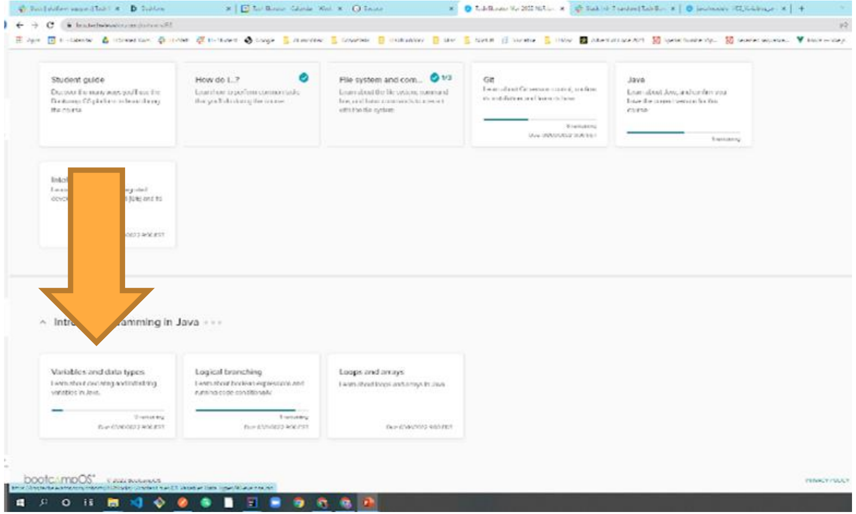
The Internet:

File is untracked

File is staged

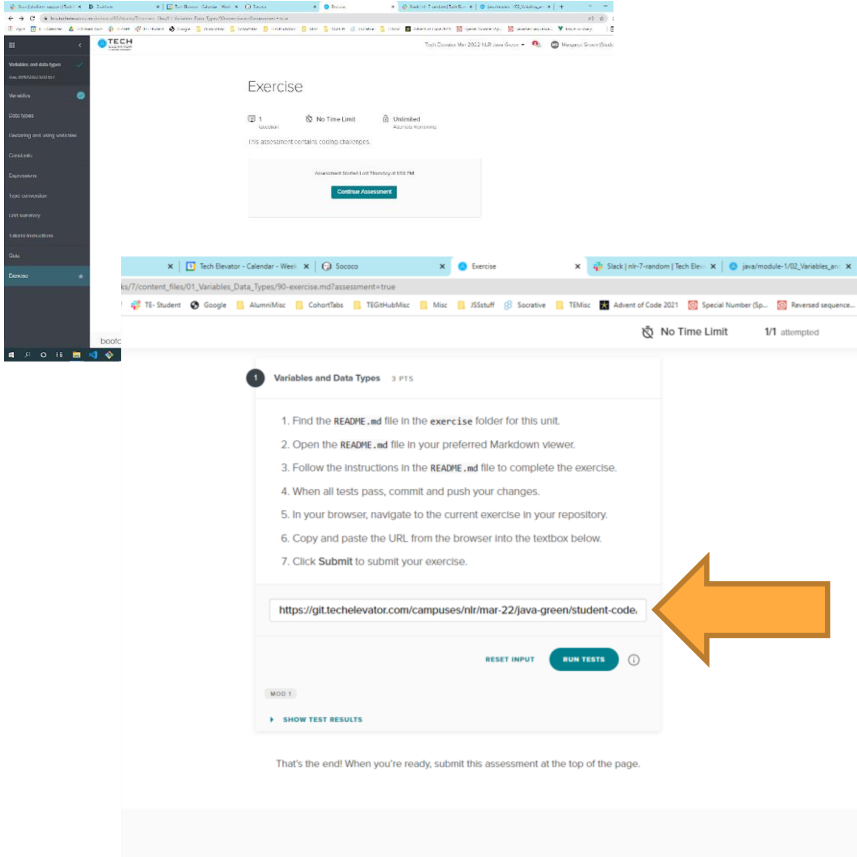
File is committed

How to submit your exercises in LMS



- Open a new tab in Chrome and navigate to lms.techelevator.com
- Find the assignment

How to submit your exercises



Exercise

This assessment contains coding challenges.

Assessment Started Last Thursday at 1:58 PM

1 Variables and Data Types 3 PTS

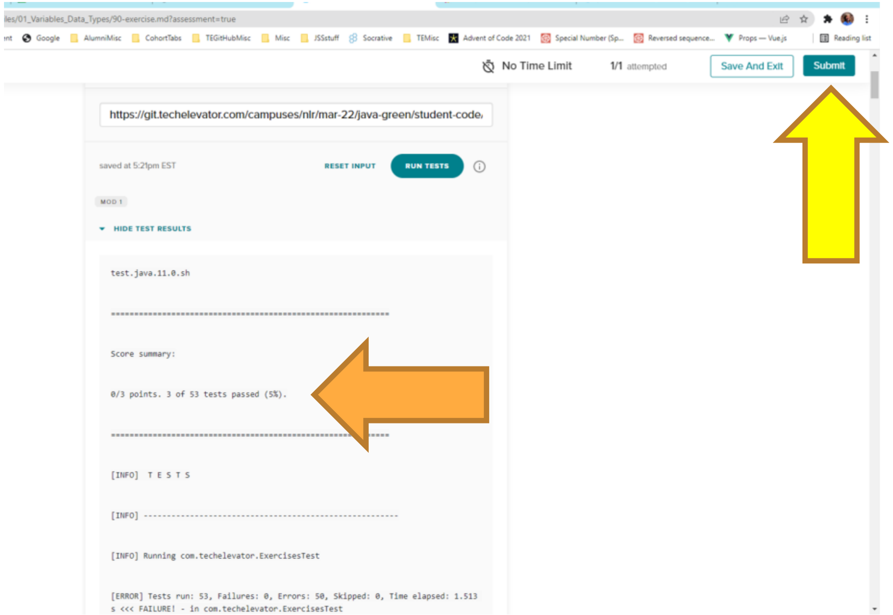
1. Find the `README.md` file in the `exercise` folder for this unit.
2. Open the `README.md` file in your preferred Markdown viewer.
3. Follow the instructions in the `README.md` file to complete the exercise.
4. When all tests pass, commit and push your changes.
5. In your browser, navigate to the current exercise in your repository.
6. Copy and paste the URL from the browser into the textbox below.
7. Click **Submit** to submit your exercise.

https://git.techelevator.com/campuses/nlr/mar-22/java-green/student-code,

That's the end! When you're ready, submit this assessment at the top of the page.

- Click Start Assessment/Run Tests

How to submit your exercises



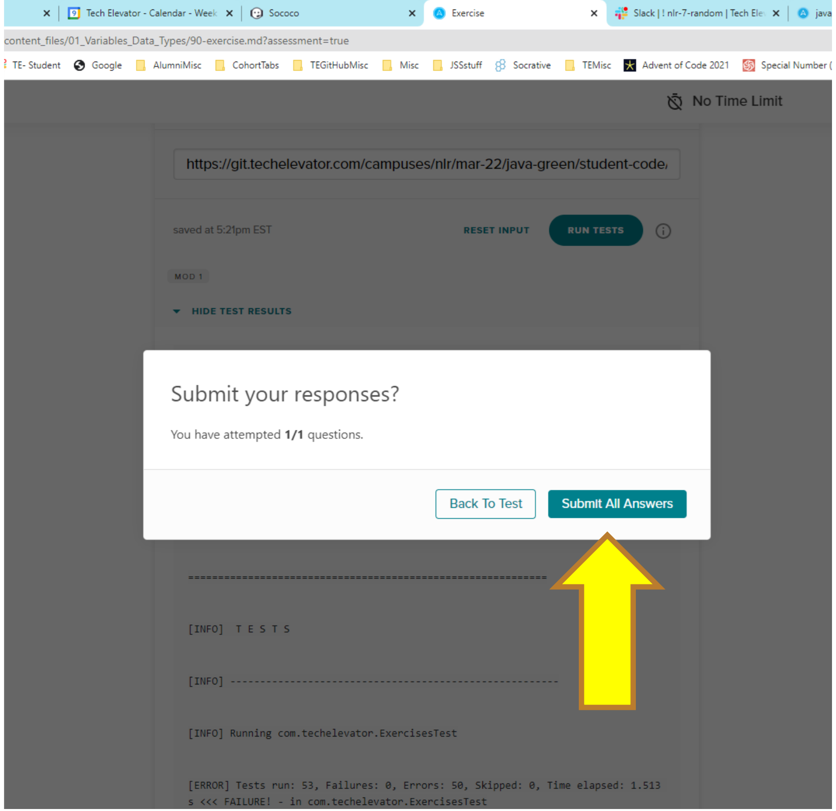
https://git.techelevator.com/campuses/nlr/mar-22/java-green/student-code/

saved at 5:21pm EST

```
test.java.11.0.sh

============================================================

Score summary:

0/3 points. 3 of 53 tests passed (5%).

============================================================

[INFO]  T E S T S

[INFO] -------------------------------------------------------

[INFO] Running com.techelevator.ExercisesTest

[ERROR] Tests run: 53, Failures: 0, Errors: 50, Skipped: 0, Time elapsed: 1.513
s <<< FAILURE! - in com.techelevator.ExercisesTest
```

- Check the screen for your score summary
- Don’t forget to hit the submit button!

How to submit your exercises



https://git.techelevator.com/campuses/nlr/mar-22/java-green/student-code/

saved at 5:21pm EST

Submit your responses?

You have attempted **1/1** questions.

```
============================================================

[INFO]  T E S T S

[INFO] -------------------------------------------------------

[INFO] Running com.techelevator.ExercisesTest

[ERROR] Tests run: 53, Failures: 0, Errors: 50, Skipped: 0, Time elapsed: 1.513
s <<< FAILURE! - in com.techelevator.ExercisesTest
```

- Submit all answers

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

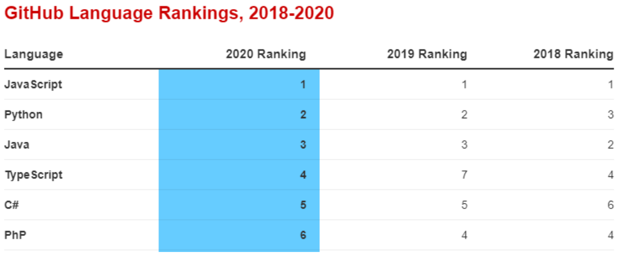
What is Java?

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

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/).



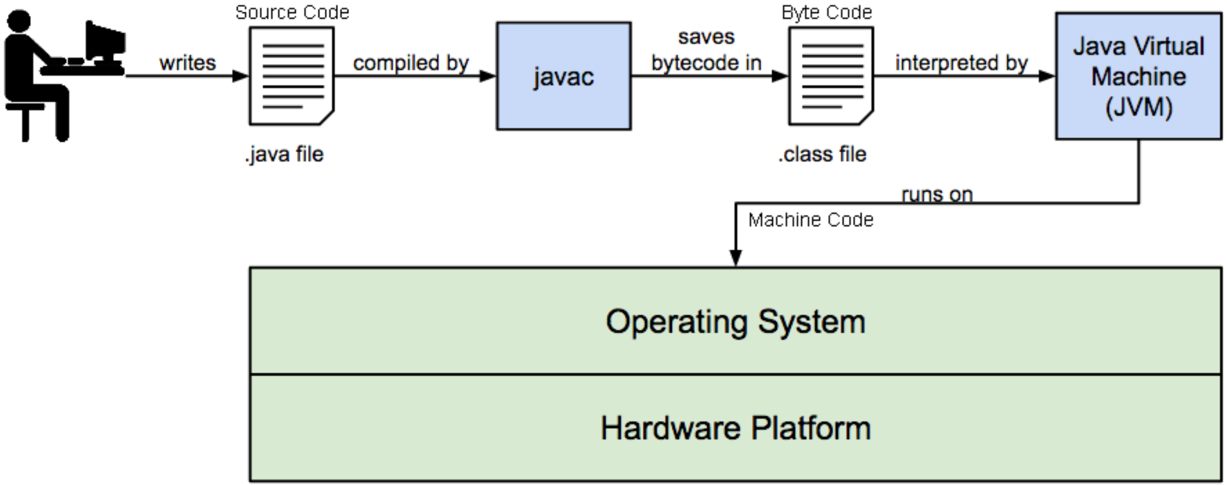
GitHub Language Rankings, 2018-2020

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 Architecture



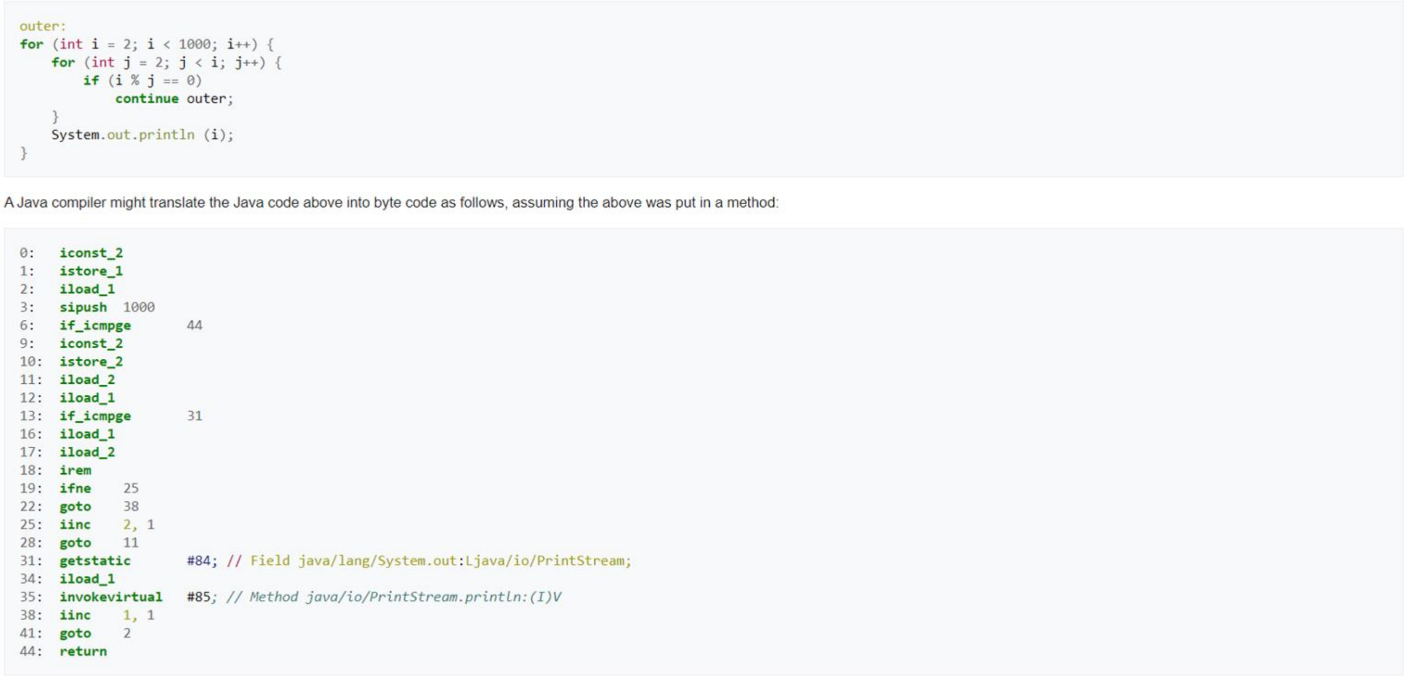
Source Code - code written by humans. *Stored in .java files.*

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.

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);
}
```

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 we will write

This is what the java translates it to

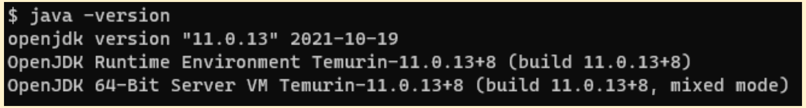
Source: https://en.wikipedia.org/wiki/Java_bytecode

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

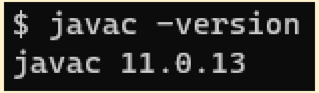
In terminal: java -version



```
$ java -version
openjdk version "11.0.13" 2021-10-19
OpenJDK Runtime Environment Temurin-11.0.13+8 (build 11.0.13+8)
OpenJDK 64-Bit Server VM Temurin-11.0.13+8 (build 11.0.13+8, mixed mode)
```

Java JDK Version Check

In terminal: javac -version



```
$ javac -version
javac 11.0.13
```

Hello World!

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

```
$ javac HelloWorld.java
```

9. Run the program using the *JVM*

```
$ java HelloWorld
```

Source Code

```
public class HelloWorld {

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

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

Parts of Hello World

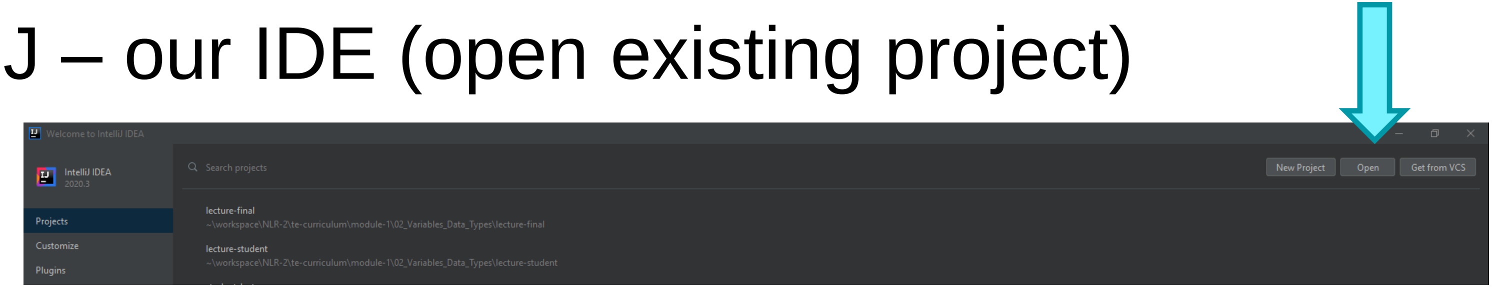
Code	Description
`public class HelloWorld`	Java programs are made up of one or more classes which hold program code. All Java code must be in a class.
`public static void main(String[] args)`	The entry point to the program. When we run the program this contains the code that is executed.
`{...}`	Encloses a set of commands. For each opening brace there is a closing brace
`// Prints out a message`	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)

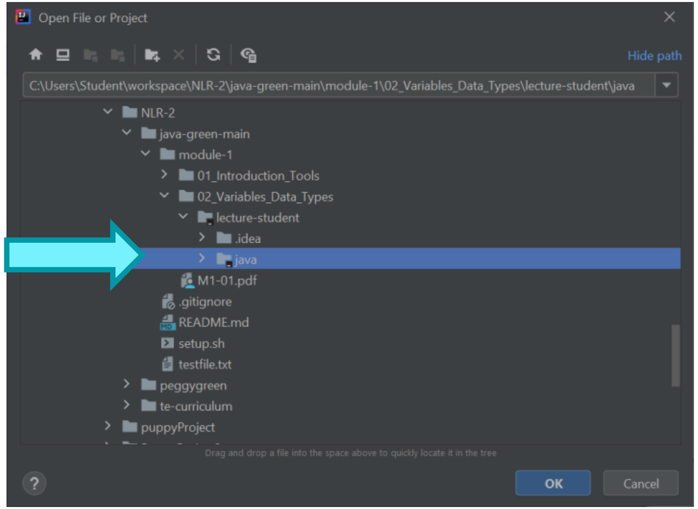
- Organizes code into projects
- Provides immediate feedback on syntax errors
- Assists in code with Intellisense
- Performs many routine tasks and supplies common code snippets
- Allows us to suspend a program and step through using a **Debugger**

IntelliJ – our IDE (open existing project)



lecture-final
~\workspace\NLR-2\te-curriculum\module-1\02_Variables_Data_Types\lecture-final

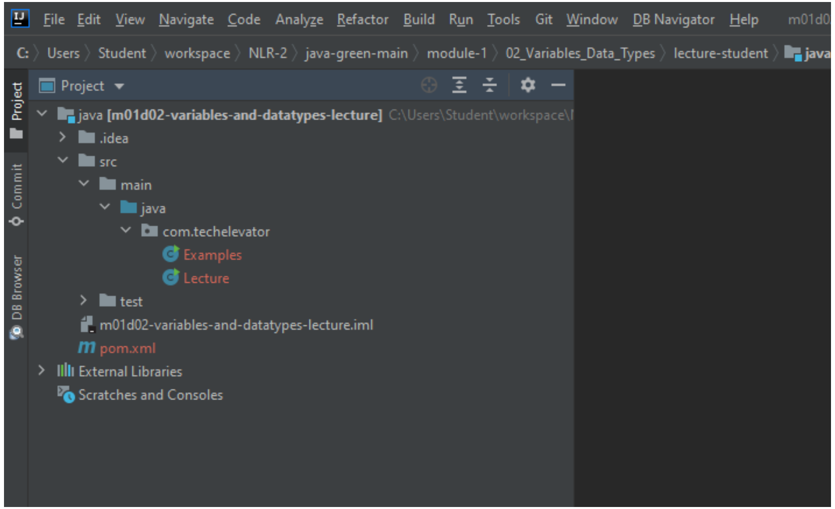
lecture-student
~\workspace\NLR-2\te-curriculum\module-1\02_Variables_Data_Types\lecture-student



C:\Users\Student\workspace\NLR-2\java-green-main\module-1\02_Variables_Data_Types\lecture-student\java

```
NLR-2
  java-green-main
    module-1
      01_Introduction_Tools
      02_Variables_Data_Types
        lecture-student
          .idea
          java
      M1-01.pdf
    .gitignore
    README.md
    setup.sh
    testfile.txt
  peggygreen
  te-curriculum
puppyProject
```

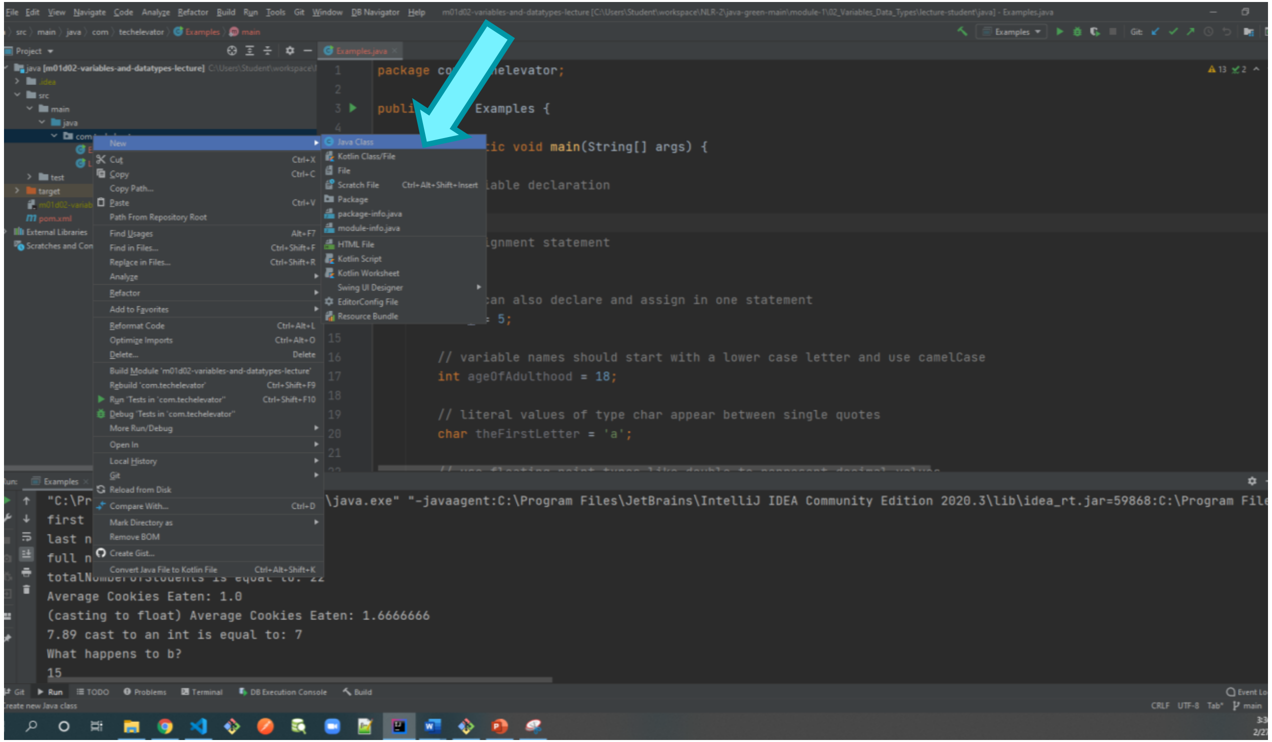
Drag and drop a file into the space above to quickly locate it in the tree



C: 〉 Users 〉 Student 〉 workspace 〉 NLR-2 〉 java-green-main 〉 module-1 〉 02_Variables_Data_Types 〉 lecture-student 〉 java

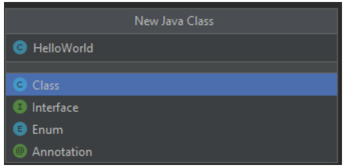
```
java [m01d02-variables-and-datatypes-lecture] C:\Users\Student\workspace\
  .idea
  src
    main
      java
        com.techelevator
          Examples
          Lecture
    test
  m01d02-variables-and-datatypes-lecture.iml
  pom.xml
External Libraries
Scratches and Consoles
```

IntelliJ – our IDE!


File Edit View Navigate Code Analyze Refactor Build Run Tools Git Window DB Navigator Help
m01d02-variables-and-datatypes-lecture [C:\Users\Student\workspace\NLR-2\java-green-main\module-1\02_Variables_Data_Types\lecture-student\java] - Examples.java
src 〉 main 〉 java 〉 com 〉 techelevator 〉 Examples 〉 main
Project
Examples.java
java [m01d02-variables-and-datatypes-lecture] C:\Users\Student\workspace\
.idea
src
main
java
com
test
target
m01d02-variab
pom.xml
External Libraries
Scratches and Con
New
Cut Ctrl+X
Copy Ctrl+C
Copy Path...
Paste Ctrl+V
Path From Repository Root
Find Usages Alt+F7
Find in Files... Ctrl+Shift+F
Replace in Files... Ctrl+Shift+R
Analyze
Refactor
Add to Favorites
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Build Module 'm01d02-variables-and-datatypes-lecture'
Rebuild 'com.techelevator' Ctrl+Shift+F9
Run 'Tests in 'com.techelevator'' Ctrl+Shift+F10
Debug 'Tests in 'com.techelevator''
More Run/Debug
Open In
Local History
Git
Reload from Disk
Compare With... Ctrl+D
Mark Directory as
Remove BOM
Create Gist...
Convert Java File to Kotlin File Ctrl+Alt+Shift+K
Java Class
Kotlin Class/File
File
Scratch File Ctrl+Alt+Shift+Insert
Package
package-info.java
module-info.java
HTML File
Kotlin Script
Kotlin Worksheet
Swing UI Designer
EditorConfig File
Resource Bundle
Examples
Git
Run
TODO
Problems
Terminal
DB Execution Console
Build
Create new Java class
CRLF UTF-8 Tab* main

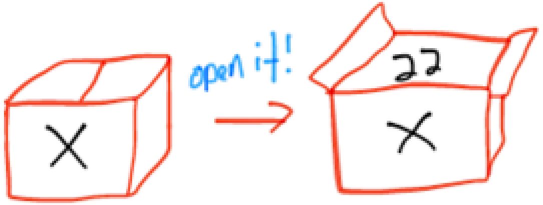
```
package co     elevator;

publi          Examples {

           tic void main(String[] args) {

           iable declaration

           ignment statement

           can also declare and assign in one statement
              = 5;

        // variable names should start with a lower case letter and use camelCase
        int ageOfAdulthood = 18;

        // literal values of type char appear between single quotes
        char theFirstLetter = 'a';
```

```
"C:\Pr   \java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.3\lib\idea_rt.jar=59868:C:\Program File
first
last n
full n
totalNu                    is equal to: 22
Average Cookies Eaten: 1.0
(casting to float) Average Cookies Eaten: 1.6666666
7.89 cast to an int is equal to: 7
What happens to b?
15
```



HelloWorld

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.

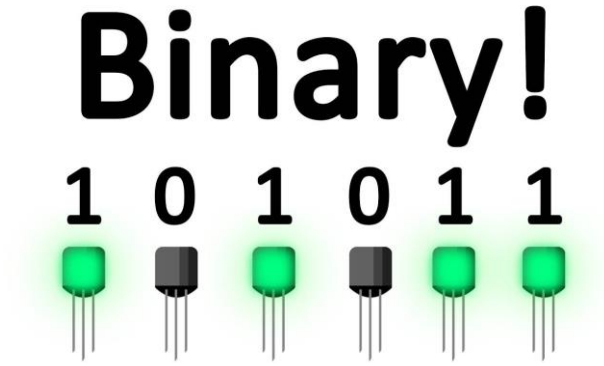
- **Parts:**
  - Data Type
  - Name
  - Value

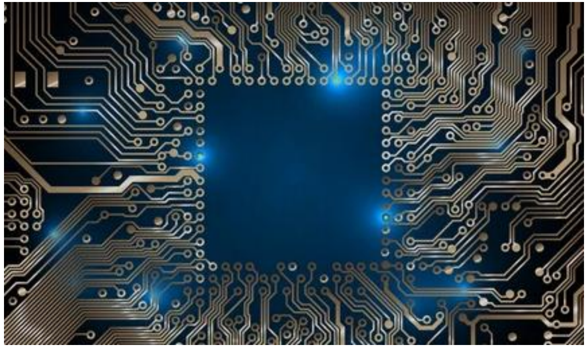
A variable allows us to set and hold a value to be used later. They are also used to make code more readable.

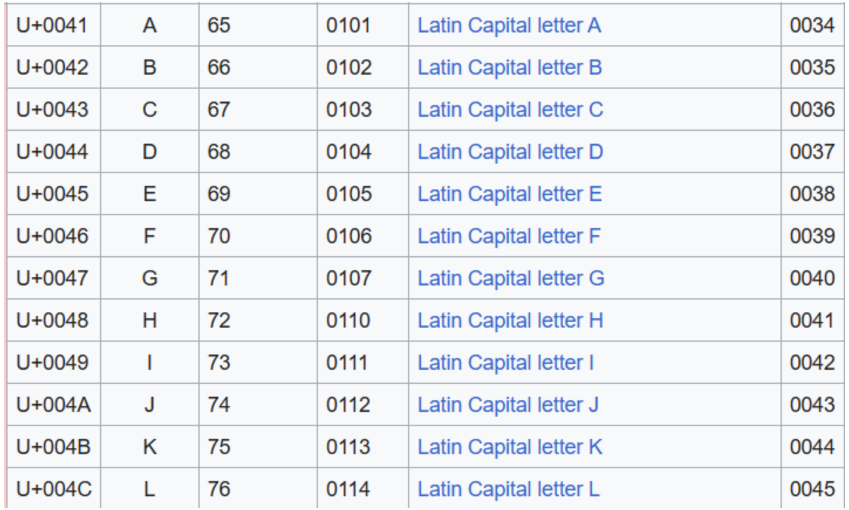
Java Primitive Data Types

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

Data Types







U+0041	A	65	0101	Latin Capital letter A	0034
U+0042	B	66	0102	Latin Capital letter B	0035
U+0043	C	67	0103	Latin Capital letter C	0036
U+0044	D	68	0104	Latin Capital letter D	0037
U+0045	E	69	0105	Latin Capital letter E	0038
U+0046	F	70	0106	Latin Capital letter F	0039
U+0047	G	71	0107	Latin Capital letter G	0040
U+0048	H	72	0110	Latin Capital letter H	0041
U+0049	I	73	0111	Latin Capital letter I	0042
U+004A	J	74	0112	Latin Capital letter J	0043
U+004B	K	75	0113	Latin Capital letter K	0044
U+004C	L	76	0114	Latin Capital letter L	0045

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`.

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;
```

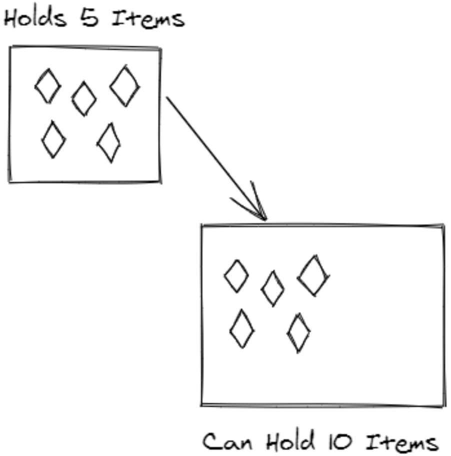
Widening

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

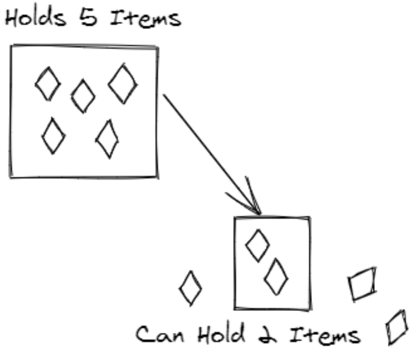
int → long
float → double

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

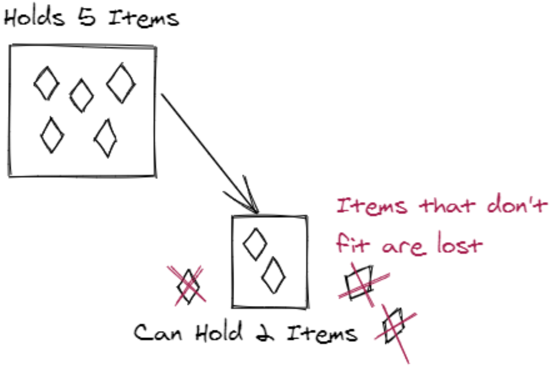
Explicitly Cast (not automatic and code must indicate that it should happen and the data type to convert (cast) it to)

```
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

Literal Suffix

In Java, a literal suffix is a character added to a literal value to indicate its data type explicitly. It helps clarify the data type of the literal when it might be ambiguous or when you want to specify the desired type explicitly. Here are some common literal suffixes used in Java:

Integer Literal Suffixes:

L or l for long data type: Example: 100L
Floating-Point Literal Suffixes:

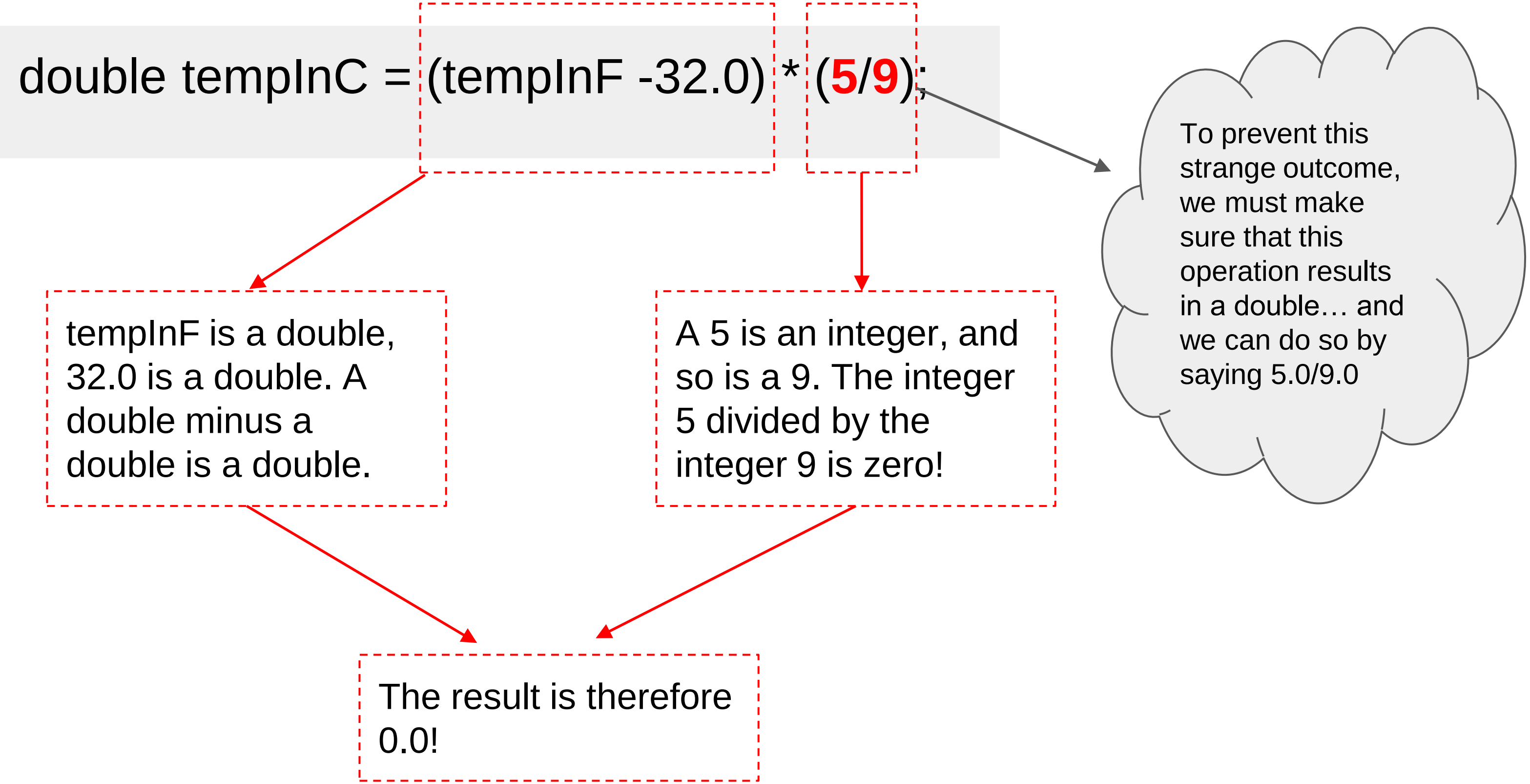
D or d for double data type (default): Example: 3.14D
F or f for float data type: Example: 2.5F

Working with Numbers: Example

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

```
public class MyClass {

        public static void main(String[] args) {




        }
}
```

Working with Numbers: Example 2 (Cautionary)



```
double tempInC = (tempInF -32.0) * (5/9);
```

tempInF is a double, 32.0 is a double. A double minus a double is a double.

A 5 is an integer, and so is a 9. The integer 5 divided by the integer 9 is zero!

The result is therefore 0.0!

To prevent this strange outcome, we must make sure that this operation results in a double… and we can do so by saying 5.0/9.0

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;
  ```

Combining Strings

The plus sign can also be used with Strings:

```
public class MyClass {

        public static void main(String[] args) {

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

The following code will print ***Simpson, Homer***. This process is known as **concatenation**.

Review Questions

1. What is the difference between an int and a double variable?

2. What is a literal?

3. What is a cast?

Review Questions - Answers

1. What is the difference between an int and a double variable?

- Int is a 32-bit signed integer data type. It can store whole numbers within a certain range.
- Range: The range of an int in Java is from -2,147,483,648 to 2,147,483,647.
- Use int when you need to work with whole numbers and do not require decimal precision.

- double is a 64-bit floating-point data type, which can store both integer and decimal (floating-point) values with higher precision.
- double has a much wider range, approximately from -1.7976931348623157E308 to 1.7976931348623157E308, and can represent a vast range of real numbers with decimal precision.
- Use double when you need to work with real numbers that include decimal places and need higher precision, such as in scientific calculations, financial applications, or when you require a wide range of values.

Review Questions - Answers

2. What is a literal?

   In Java, a literal is a source code representation of a fixed value. It's a way to directly specify a constant value within your code. Literals can be used to represent values of various data types, such as integers, floating-point numbers, characters, and strings. Here are some examples of literals in Java:

- Integer Literals:

  int x = 42; Here, 42 is an integer literal.
- Floating-Point Literals:

  double d = 2.71828; Here, 2.71828 is a double literal.
- Character Literals:

  char c = 'A'; The character 'A' enclosed in single quotes is a character literal.
- String Literals:

  String message = "Hello, World!"; The text enclosed in double quotes is a string literal.
- Boolean Literals:

  boolean isTrue = true; true and false are boolean literals.

Review Questions - Answers

3. What is a cast?

- In Java, a cast is an operation that allows you to explicitly convert or change the data type of a value from one type to another.
- Casting is sometimes necessary when you want to assign a value of one data type to a variable of another data type, but the types are not directly compatible.
- Casting can be useful when you need to avoid data loss or maintain compatibility between data types.
- Example: `double x = (double) 5 / 2;`