CS2050 Technical Documentation

Table of Contents/Navigation:

> CS2050 HAND OF GOD <

1. This is my tech doc. There are many like it, but this one is mine.

2. My tech doc is my best friend. It is my grade. I must master it as I must master my grade.

3. My tech doc, without me, is useless. Without my tech doc, I am useless. I must program my tech doc true. I must program straighter than my compiler who is trying to bug me. I must debug him before he errors me. I will...

4. My tech doc and myself know that what counts in this project is not the files we import, the lines of our code, nor the functions we call. We know that it is the runs that count. We will run...

5. My tech doc is human, even as I, because it is my grade. Thus, I will learn it as a brother. I will learn its weaknesses, its strength, its parts, its accessories, its links and its slides. I will ever guard it against the ravages of quizzes and tests as I will ever guard my legs, my arms, my eyes and my heart against damage. I will keep my tech doc clean and ready. We will become part of each other. We will...

6. Before God, I swear this creed. My tech doc and myself are the defenders of my class. We are the masters of our compiler. We are the saviors of my grade.

7. So be it, until victory is CS2050s and there is no compiler, but output!!

Da Rulez

Your first source should always be:

- Your class lecture slides, documents and the resources linked in the canvas materials, instructor (Deb), learning assistant (LA)

DOs:

- Use clear headings, table of contents, and consistent formatting.
- Headings, Subheadings, and Table of Contents (Google Docs)
- Headings, Subheadings, and Table of Contents (Microsoft Word)
- Use lists and tables to organize information
- Provide explanations in your own words — no copy/paste from slides.
- Include code examples that you explored and edited, screenshots, or diagrams
- Include links to resources from the lecture or find your own external resources (websites, AI, videos) with a brief summary

DON'Ts:

- Copy explanations from the slides, AI, Websites, or others as your work
- Use AI to create your technical documentation

General Resources

- CS2050 Resources - Slides, documents, and other course info
- CS1050 Resources - Slides, documents, and other previous course info
- CSCI200 Schedule - Mines CS2 course (namely resources)
- CSCI128 Channel - Mines CS1 Youtube channel with basic lectures/guides
- Runestone Academy Library of Books - library of STEM-related open-source textbooks. No need to register
- CSHWStuff (.zips) - Reference code for examples

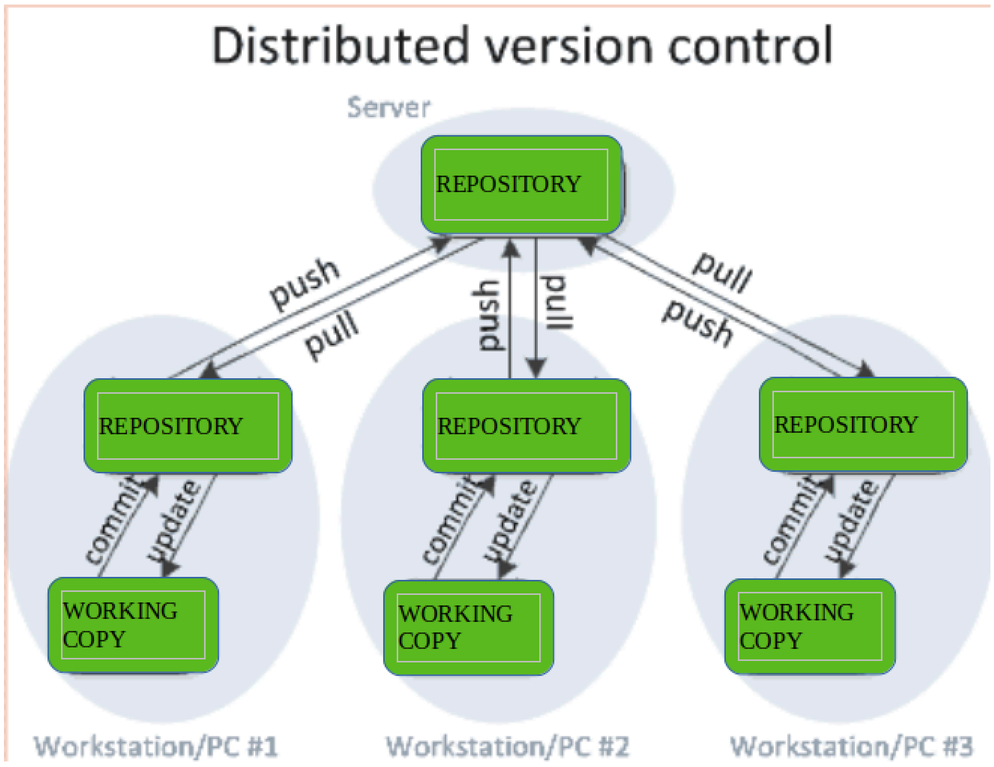
Tools and Quicklinks

- Github Profile - Quicklink
- Visualize Java Code - Used to step through code and edit code
-

Development Environment

IDE (Integrated Development Environment): A program that allows for the development, test, and implementation of code within itself (involves some sort of input/coding window and an output/debugging window). (Reference)

Version control: The means to determine what alternative iterations of a single program or code are implemented or changed at any given moment. Usually involves clear documentation of a build number (ex. 4.2.1.14) and has some sort of hierarchical structure given the context of the people working with it, in which repositories (repos) act as as backup and shared stable copies of the program or code, giving each active person to work on it and decide on any subsequent changes. Repos may or may not have multiple layers given project scope of the affiliated working party and/or means to control hierarchical changes, up to the master file. Github is an example of a program designed for version control.



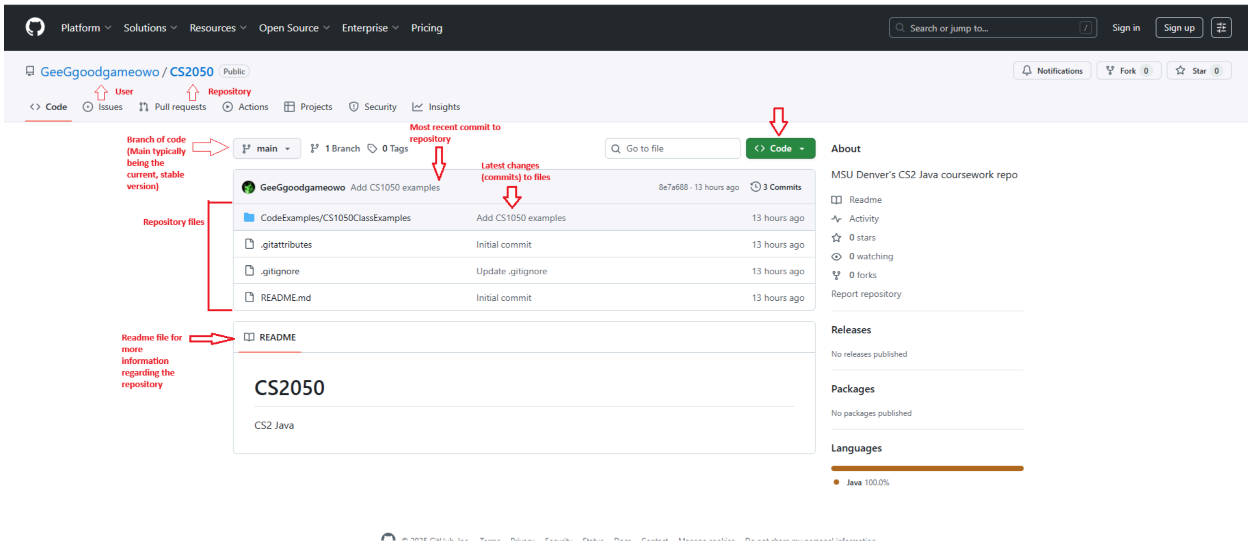
(Source)

Git and GitHub

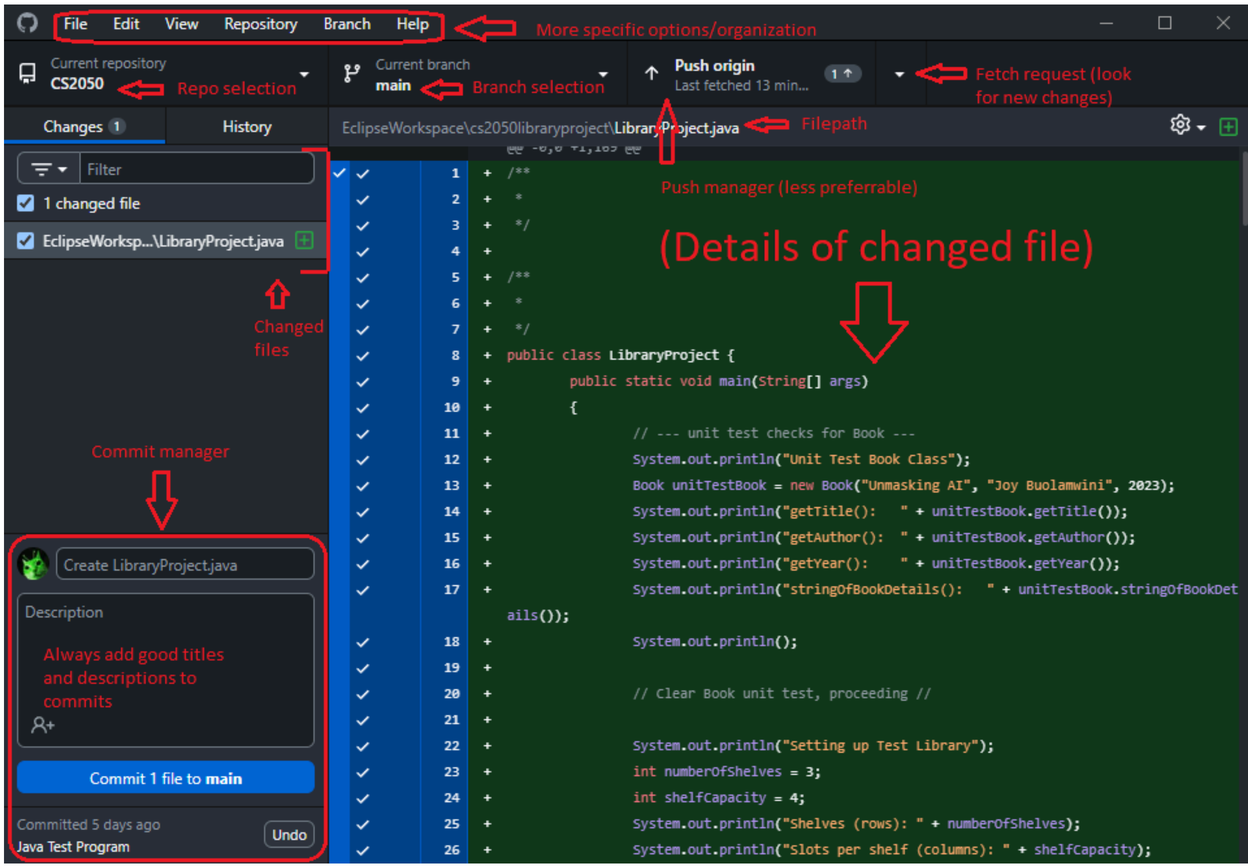
Resources:
Coder Coder: Git, GitHub, & GitHub Desktop for beginners - YT video summarizing Git, Github, and basic GitHub functionality

[Environment Setup Eclipse, Git and Github]() - CS1050 Guide on Eclipse, Git, and Github setup

Git: A software language used for version control of code and moderation of changes to the code amongst separate computers and programmers
GitHub: Microsoft's platform for Git usage with a more friendly UI and social network environment

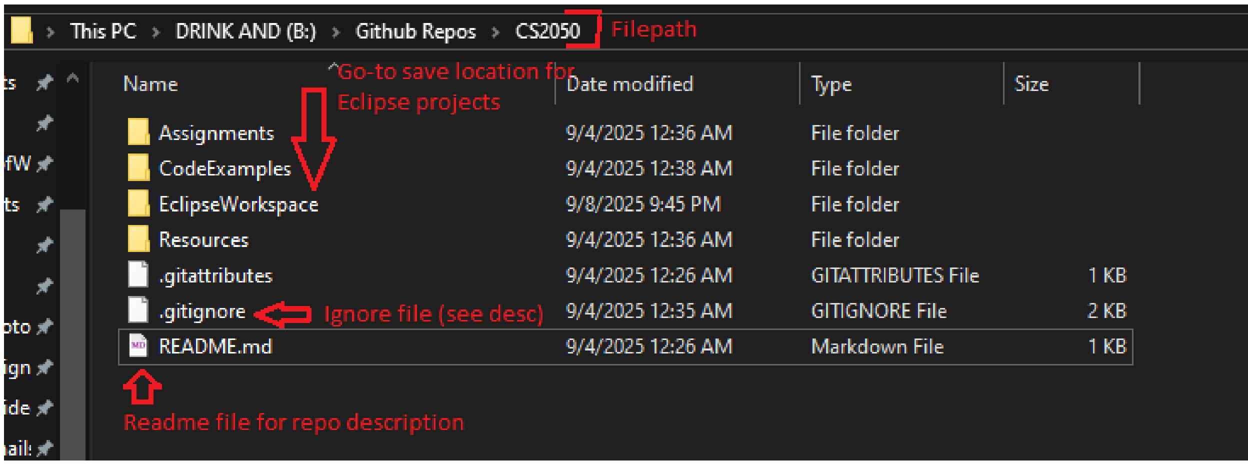


(Github web repo example)


File
Edit
View
Repository
Branch
Help
More specific options/organization
Current repository
CS2050
Repo selection
Current branch
main
Branch selection
Push origin
Last fetched 13 min...
Fetch request (look for new changes)
Changes 1
History
EclipseWorkspace\cs2050libraryproject\LibraryProject.java
Filepath
Push manager (less preferrable)
(Details of changed file)
Filter
1 changed file
EclipseWorksp...\LibraryProject.java
Changed files
Commit manager
Create LibraryProject.java
Description
Always add good titles and descriptions to commits
Commit 1 file to main
Committed 5 days ago
Java Test Program
Undo

```
@@ -0,0 +1,169 @@
+ /**
+  *
+  */
+
+ /**
+  *
+  */
+ public class LibraryProject {
+         public static void main(String[] args)
+         {
+                 // --- unit test checks for Book ---
+                 System.out.println("Unit Test Book Class");
+                 Book unitTestBook = new Book("Unmasking AI", "Joy Buolamwini", 2023);
+                 System.out.println("getTitle():   " + unitTestBook.getTitle());
+                 System.out.println("getAuthor():  " + unitTestBook.getAuthor());
+                 System.out.println("getYear():    " + unitTestBook.getYear());
+                 System.out.println("stringOfBookDetails():   " + unitTestBook.stringOfBookDetails());
+                 System.out.println();
+
+                 // Clear Book unit test, proceeding //
+
+                 System.out.println("Setting up Test Library");
+                 int numberOfShelves = 3;
+                 int shelfCapacity = 4;
+                 System.out.println("Shelves (rows): " + numberOfShelves);
+                 System.out.println("Slots per shelf (columns): " + shelfCapacity);
```

(Github Desktop example)



This PC > DRINK AND (B:) > Github Repos > CS2050

Filepath

^Go-to save location for Eclipse projects

Name	Date modified	Type	Size
Assignments	9/4/2025 12:36 AM	File folder	
CodeExamples	9/4/2025 12:38 AM	File folder	
EclipseWorkspace	9/8/2025 9:45 PM	File folder	
Resources	9/4/2025 12:36 AM	File folder	
.gitattributes	9/4/2025 12:26 AM	GITATTRIBUTES File	1 KB
.gitignore	9/4/2025 12:35 AM	GITIGNORE File	2 KB
README.md	9/4/2025 12:26 AM	Markdown File	1 KB

Ignore file (see desc)

Readme file for repo description

(Github local file organization example)

Gitignore file: Attached file to the repo that Github ignores for repository version control. "This is useful for keeping log files, temporary files, build artifacts, or personal files out of your repository." (Source)

-Examples of files to ignore: log files, temporary files, hidden files, personal files, OS/editor files, etc.

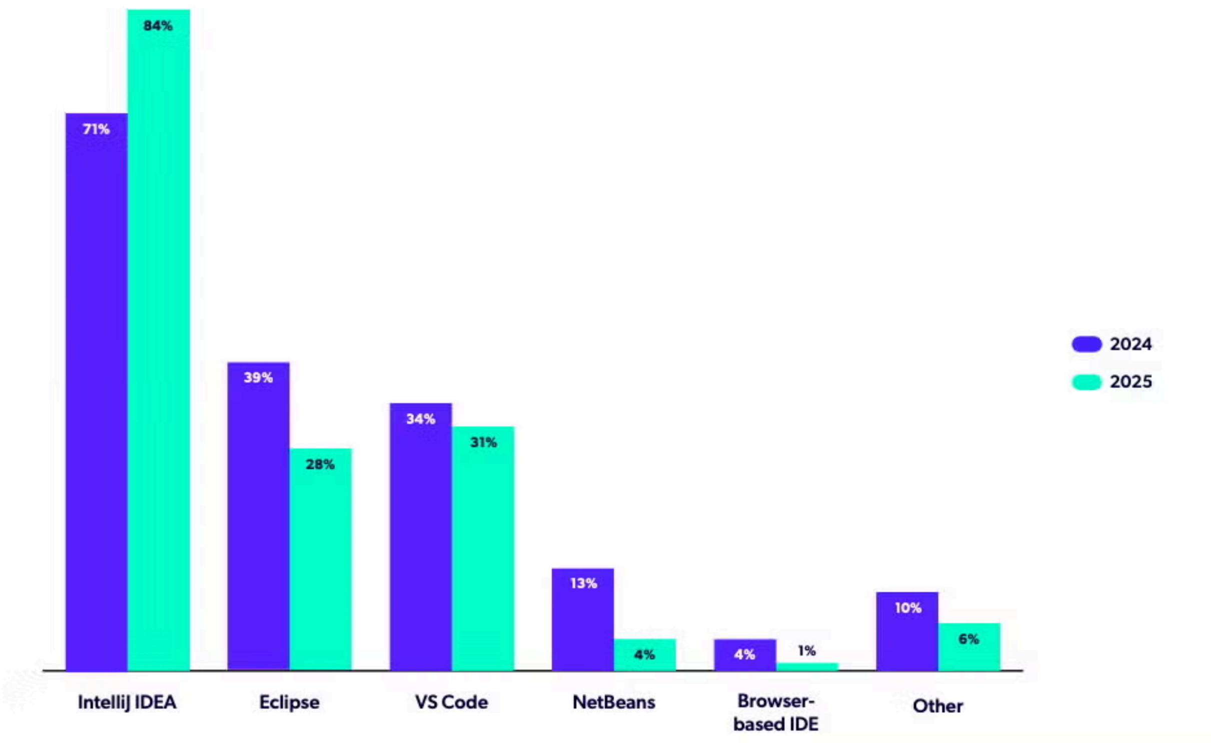
>Put GitHub instructions setup on GitHub

>Put eclipse setup on github

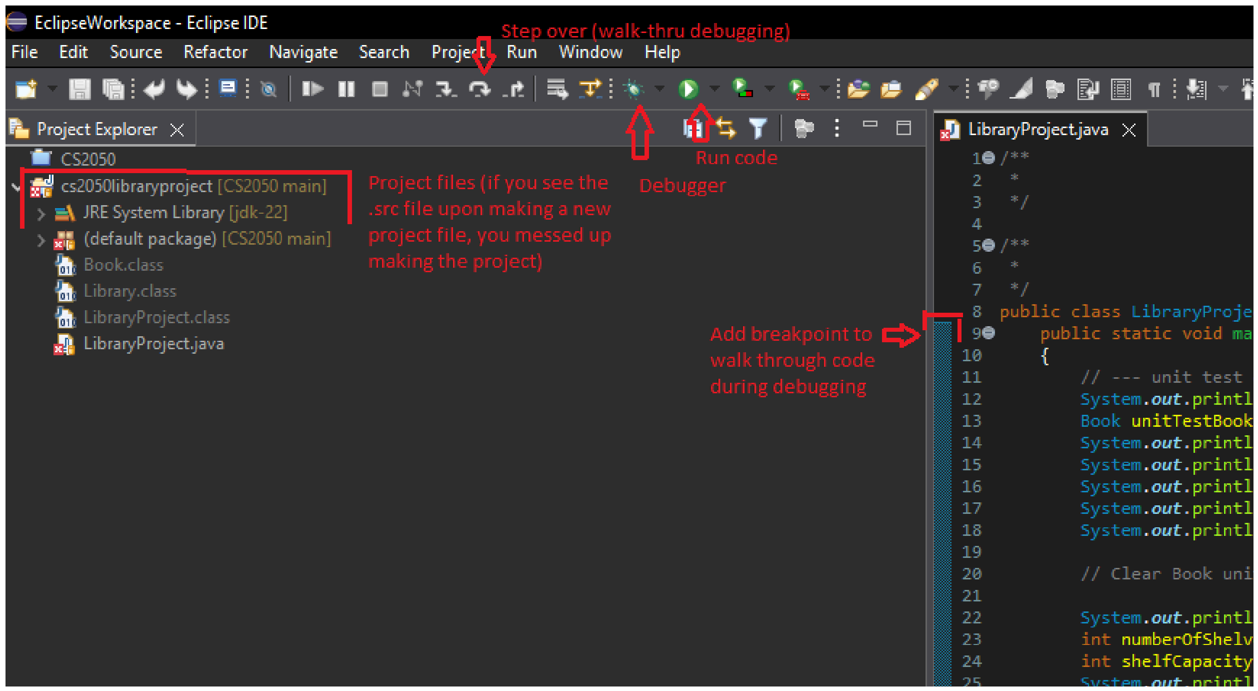
>Put tech doc on github

Eclipse IDE

Resources:
Eclipse: Changing the Java Code Style Formatting - Basic visual guide for changing Eclipse formatting
Eclipse Tips - CS1050 Debugger tips
Eclipse Cheat Sheet - Visual slideshow on basic Eclipse functionality



[(Eclipse vs Industry, Source)]()


EclipseWorkspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
LibraryProject.java
CS2050
cs2050libraryproject [CS2050 main]
JRE System Library [jdk-22]
(default package) [CS2050 main]
Book.class
Library.class
LibraryProject.class
LibraryProject.java

Step over (walk-thru debugging)

Project files (if you see the .src file upon making a new project file, you messed up making the project)

Debugger

Run code

Add breakpoint to walk through code during debugging

```
/**
 *
 */

/**
 *
 */
public class LibraryProje
    public static void ma
    {
        // --- unit test
        System.out.printl
        Book unitTestBook
        System.out.printl
        System.out.printl
        System.out.printl
        System.out.printl
        System.out.printl

        // Clear Book uni

        System.out.printl
        int numberOfShelv
        int shelfCapacity
        System.out.printl
```

(Eclipse IDE essentials)

Eclipse IDE is an open-source, industry-standard, free coding environment for (mainly) Java. Main concepts involve in-program development framework and debugging, and clear formatting for syntax and coding preferences. The necessities are listed in the picture above, more notes will be listed as different utilities are used across the length of the course.

Use the Eclipse Cheat Sheet to ensure the proper creation of a new project.

CODING QUALITY AND CONSISTENCY:

https://vitalflux.com/checklist-code-quality-matches-iso-quality-standards-square/

https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html

SUMMARIZE THESE STILL VVVVVVVVVVV
Single Responsibility Principle (SRP)
Each task (function/method) has a single responsibility. When you can't find a simple descriptive name (action verb) for a function/method it may have too many tasks

Software Design Principles Don't Repeat Yourself (DRY) and Keep It Simple Silly (KISS)
When you find yourself writing the same algorithm over and over again create a method that can be used over and over.

https://www.agilealliance.org/glossary/tdd/

https://www.agilealliance.org/glossary/unit-test/

https://mitcommlab.mit.edu/broad/commkit/coding-mindset/

https://www.javatpoint.com/java-naming-conventions

HARD CODING IS FOR LOSERS

https://www.agilealliance.org/glossary/heartbeatretro/#q=~

ENGRAIN THIS FOR ANY TECH INTERVIEW EVER
https://docs.google.com/presentation/d/1GNVTjZu4YdGPs6kaXF_DOECYMwzhDGxYufdeTUN9XYk/edit?slide=id.p1#slide=id.p1

One Return Value: When creating methods only one return value should be used

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/

CODE ORGANIZATION:
https://docs.google.com/presentation/d/1gp0D0pcgXBeg_HC7L1uHjliECrFVWzHS/edit?slide=id.g34b987c011b_2_14#slide=id.g34b987c011b_2_14

CLASSES GUIDE
https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/

JAVA CONCEPTS TO MEMORIZE
http://www.btechsmartclass.com/java/java-oop-concepts.html

MINDSET
https://mitcommlab.mit.edu/broad/commkit/coding-mindset/

BREAK DOWN SOURCES INTO SUMMARIZATION LIKE THIS VV
You have been introduced to a mindset that mirrors what professionals use in industry—Agile Software Development. This mindset includes:
Breaking problems into smaller, testable tasks (SRP - Single Responsibility Principle)
Writing modular, reusable methods (DRY - Don't Repeat Yourself)
Thinking iteratively by starting small and improving based on testing
Creating quality, readable code—which isn't just about the computer reading it, but documenting it for other developers to understand.
Industry values developers who can:
Collaborate to solve problems and design solutions
Communicate clearly through code and documentation
Debug using IDE tools
Refactor based on feedback
Think critically about modularity, memory, and design patterns

Refactoring: Improving code quality without adding new functionality (refinement)

Module 1

Objectives:
- Analyze and Apply principles of computing to identify solutions
    - Review and apply prerequisite knowledge from CS1050:
        - Appropriate data types, expressions, assignments, and variables.
        - Logical control structures (if/else, switch, loops).
        - Compare pass by value and pass by reference
        - Tracing program execution to understand scope and memory.
    - Apply 1d and 2D arrays and Array of objects

        - Explain how information is stored in memory (stack vs heap, references, arrays, objects).
        - Define and implement classes, objects, instance/static variables, and methods.
        - Apply encapsulation using access modifiers, constructors, and getters/setters.
    - Apply Software Development Life Cycle to Design, Implement and Evaluate Solutions
        - Design classes using UML and use Test Driven Development to implement and unit test solution.
        - Write readable, maintainable code following industry coding standards (naming, indentation, comments).
        - Create a program using an IDE with Debugger, version control (Git/GitHub), and Javadoc documentation.
    - Communicate and Collaborate
        - Create clear, concise technical documentation that explains code functionality using correct terminology, visuals, and examples.
        - Collaborate effectively in teams by contributing to discussions, decision-making, and shared problem-solving.

CS1050 Review

CS1050 - Module 1: Fundamental Concepts

Identifiers: Naming convention for certain aspects of the code. Cannot be functional words or modifiers of the code and should follow best coding practice for industry standard.

Classes: Begin with an uppercase (“TechDocClassExample”, “NumberTwo”)
Variables: Follow ‘camelCase’ conventions (“techDoc”, “sittingDuration”)
Methods: Verb followed by the noun, with similar ‘camelCase’ conventions (“writingTechDoc”, “isWiping”)

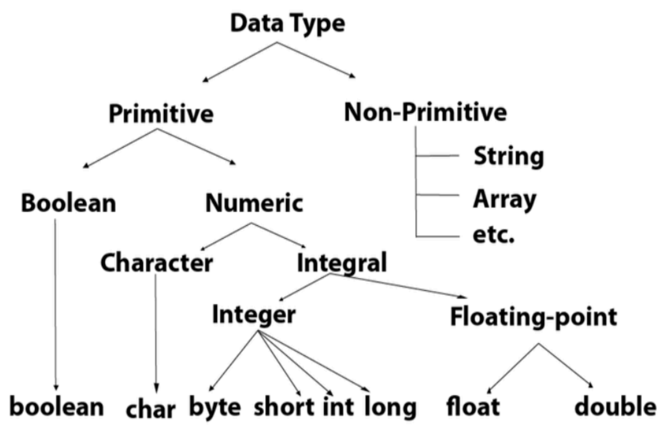
See: [Oracle’s Java Naming Conventions]

Variables:
Stores a single data type with an address to a location in memory. Depending on the type of variable, it can be stored on either the stack or the heap. Variables are, realistically, a more applicable form of an address, pointing to an allocation of the value of the variable in memory. The variable name becomes synonymous with an address (albeit the variable moreso acts as a secondary pointer to the address), which points to the assigned value, therefore returning the value on memory when called.
Local variables are specifically declared in a method vs instance and static variables which are declared in classes, and always take precedent over other variables of the same name until they are out of scope.

```
Ex: int happyCoders = 13;
    return happyCoders;
(Variable in code)   (Address on memory)    (Value on memory)
happyCoders      —>       009AFD30       —>        13
```

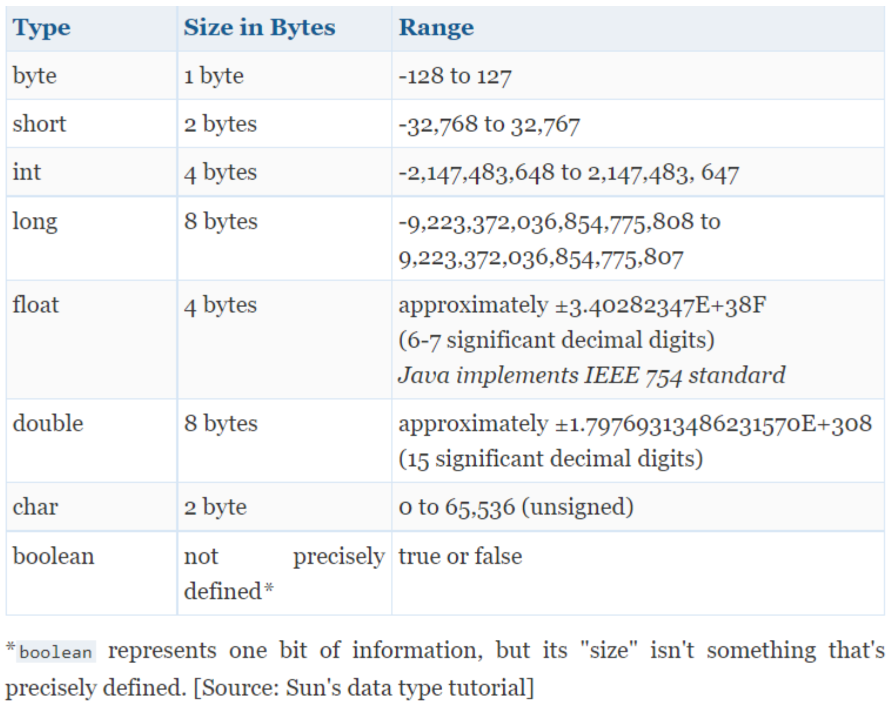
int happyCoders declares the happyCoders variable as an int data type. = 13 assigns the value to it, initializing the variable.

Data Types:
Describes the nature of the information being stored. Consists of primitive and non-primitive types, storing different things for different purposes:



(Source)

REMEMBER THESE FOR TECH INTERVIEWS



Type	Size in Bytes	Range
byte	1 byte	-128 to 127
short	2 bytes	-32,768 to 32,767
int	4 bytes	-2,147,483,648 to 2,147,483, 647
long	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	approximately ±3.40282347E+38F (6-7 significant decimal digits) *Java implements IEEE 754 standard*
double	8 bytes	approximately ±1.79769313486231570E+308 (15 significant decimal digits)
char	2 byte	0 to 65,536 (unsigned)
boolean	not precisely defined*	true or false

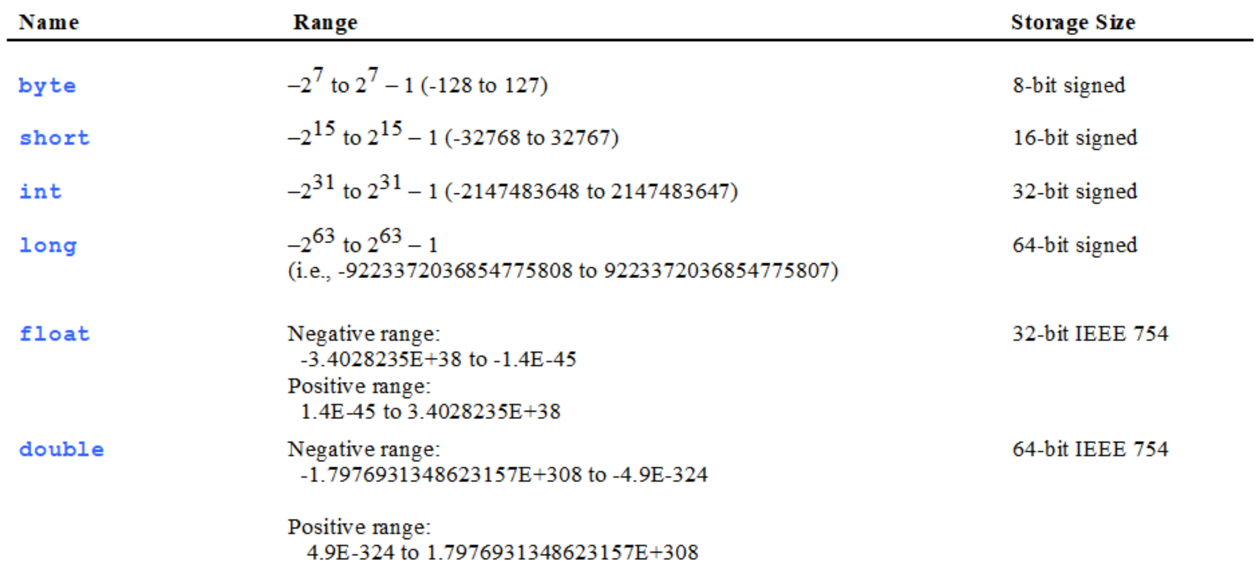
*`boolean` represents one bit of information, but its "size" isn't something that's precisely defined. [Source: Sun's data type tutorial]

(Source)

byte	short	int	long
-128 to 127	-32,768 to 32,767	-2,147,483,648 to 2,147,483,647	- huge to huge
8 bits	16 bits	32 bits	64 bits

	float	double
Range	~ 1.4E-045 to 3.4E+038	~ 4.9E-324 to 1.8E+308
Precision	single precision	double-precision
Storage	4 bytes (32 bits)	8 bytes (64 bits)

(Source)



Name	Range	Storage Size
`byte`	-2^7 to $2^7 - 1$ (-128 to 127)	8-bit signed
`short`	-2^{15} to $2^{15} - 1$ (-32768 to 32767)	16-bit signed
`int`	-2^{31} to $2^{31} - 1$ (-2147483648 to 2147483647)	32-bit signed
`long`	-2^{63} to $2^{63} - 1$ (i.e., -9223372036854775808 to 9223372036854775807)	64-bit signed
`float`	Negative range: -3.4028235E+38 to -1.4E-45 Positive range: 1.4E-45 to 3.4028235E+38	32-bit IEEE 754
`double`	Negative range: -1.7976931348623157E+308 to -4.9E-324 Positive range: 4.9E-324 to 1.7976931348623157E+308	64-bit IEEE 754

(Source)

Notes:
-Longs are specifically declared with an L at the end of the literal.
Ex (Source):
long bigNumber = 2147483647; (is actually an int, lol)
long bigNumber = 2147483648; (is an int that's out of bounds, causing error)
long bigNumber = 2147483648**L**; (a properly declared long)

-Floats and doubles can be declared with suffixes as well (though most decimal-based math defaults to doubles), and are generally used for the same practice: large or small values with extended amounts of digit places. Doubles hold more data, but in larger volumes of numbers to fit in memory or when programming in performance-driven applications like games, floats might be preferable.
(Source)
Ex (Source):
System.out.println("1.0 / 3.0 is " + 1.0 / 3.0);

displays $\texttt{1.0 / 3.0 is 0.}\underbrace{\texttt{3333333333333333}}_{\text{16 digits}}$

System.out.println("1.0F / 3.0F is " + 1.0F / 3.0F);

displays $\texttt{1.0F / 3.0F is 0.}\underbrace{\texttt{3333333}}_{\text{7 digits}}\texttt{4}$

Constants: Variables assigned with hard values to not change throughout the length of the code. Are typically all capitalized with words separated by an underscore ("STOPGAP", "GIVEUP_COUNTER")
Literals: The values assigned during variable initialization.

(Literal)
Ex: int happyCoders = **13**;

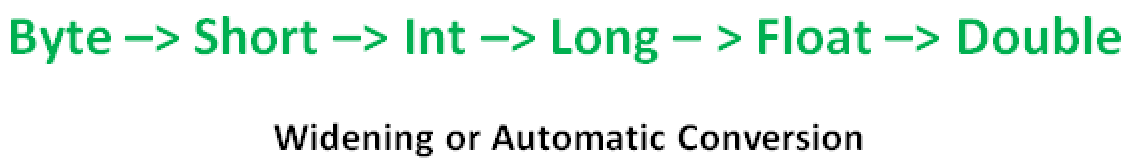
Types of errors (Source):

- Compile Time/SyntaxError: Prevent the code from executing which throw during the compile. Almost usually a syntax error, which gives it the synonymous name. Are dependent on the language programming in and the rules of its syntax. Examples in Java include lack of semi-colons at the end of lines or missing/improper brackets
- RuntimeError: Are detected during the execution of the code and are usually pertaining to invalid requests in code. Examples include improper assigned types, invalid code (dividing by zero), or indexing beyond the range of an array.
- LogicError (SemanticError): Errors that arise from your bad (incorrect, improper, or nonexistent output) when the program compiles and executes, but your output is not what was intended. These are user-based errors and will not throw an error statement.

Type Casting:
Casting converts one data type to another for variable type switching. Involves two types of casting depending on if the amount of data space is being gained

(widening/implicit) or being lost (narrowing/explicit). Implicit is normally safe to do, explicit casting must be stated.

Example:
Implicit - double d = 3; <- 3 is an int, but conversion to a double just adds enough room for the rest of the digits on the end
Explicit - int number = (int)3.0; <- explicitly states the data type, shaving off digits from the end


Byte –> Short –> Int –> Long – > Float –> Double

Widening or Automatic Conversion

(Source)

^ Notes:
-Hierarchy for automatic conversion is listed from last to first reading from left to right. If of the two types, the largest of one type is a double, the other will convert to a double. If of the two types, the largest of one type is a float, the other will convert to a float, and so on and so forth.
-Explicit conversion follows the reverse order
-char and boolean are not compatible with each other
-If the output using conversions and math functions results in overly funky numbers, it's likely caused by an overflow (too much data/digits/significant figures for the assigned data type)

CS1050 - Module 2: Predefined Classes, Methods and Decisions Structures

Methods: Functions from a computer program that call an action (verb) to a parameter. Method has parameter(s) in parenthesis. Changes or modifies values.
Formal parameters: The "placeholder" parameters listed in a generic (blank) method.
Actual parameters: The legitimate parameters/values that complete the method function
Ex.
Generic method: println(string x);
^ Method print with string x, the formal parameter of the method

Execution:
System.out.println("This is a method");
^ Method print with "This is a method" string, the actual value occupying string x

Quick output formatting:

%b: boolean
%c: char
%d: decimal in
%f: floating-point number
%e: number in standard scientific notation
%s: string
^Ex:
boolean pants = true
string sadge = “This is so sad lmao”
System.out.println(“It is %b that I ripped my pants. %s”, pants, sadge);
[Output]: “It is true that I ripped my pants. This is so sad lmao”

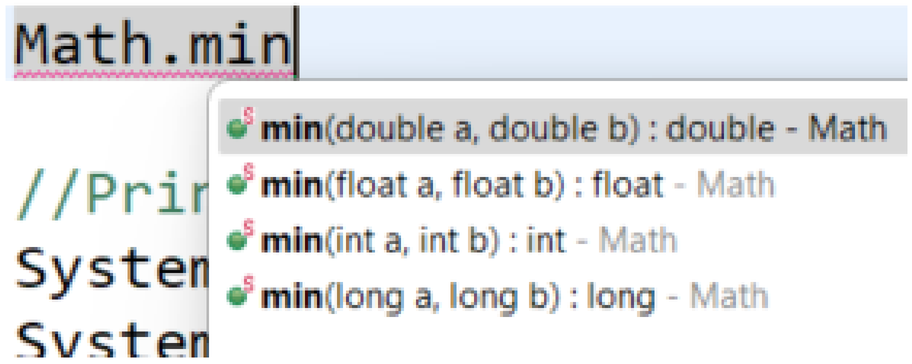
Escaping Characters (Source):
\t - tab.
\b - backspace (a step backward in the text or deletion of a single character).
\n - new line.
\r - carriage return. ()
\f - form feed.
\' - single quote.
\" - double quote.
\\ - backslash.
u+<hexadecimal digit> - Unicode character

ANALOGY OF THE CENTURY:

Class: A container of methods for objects (the blueprints to creating an instance of something), and stores variables that act like the “characteristics” to the object of the blueprint
Method: A series of modifiers within a class (describe what the object can do)
Object: An instance of or from a class, given its construction on what values are passed to it
^Ex:
```
public class Car {
        private int avgMPG = 21.2
        private string makeModel = “Ford Exploder”
        // Variables

        private void isRevd {
                System.out.println(“Vroom vroom”);
        }
        // Method
}
```

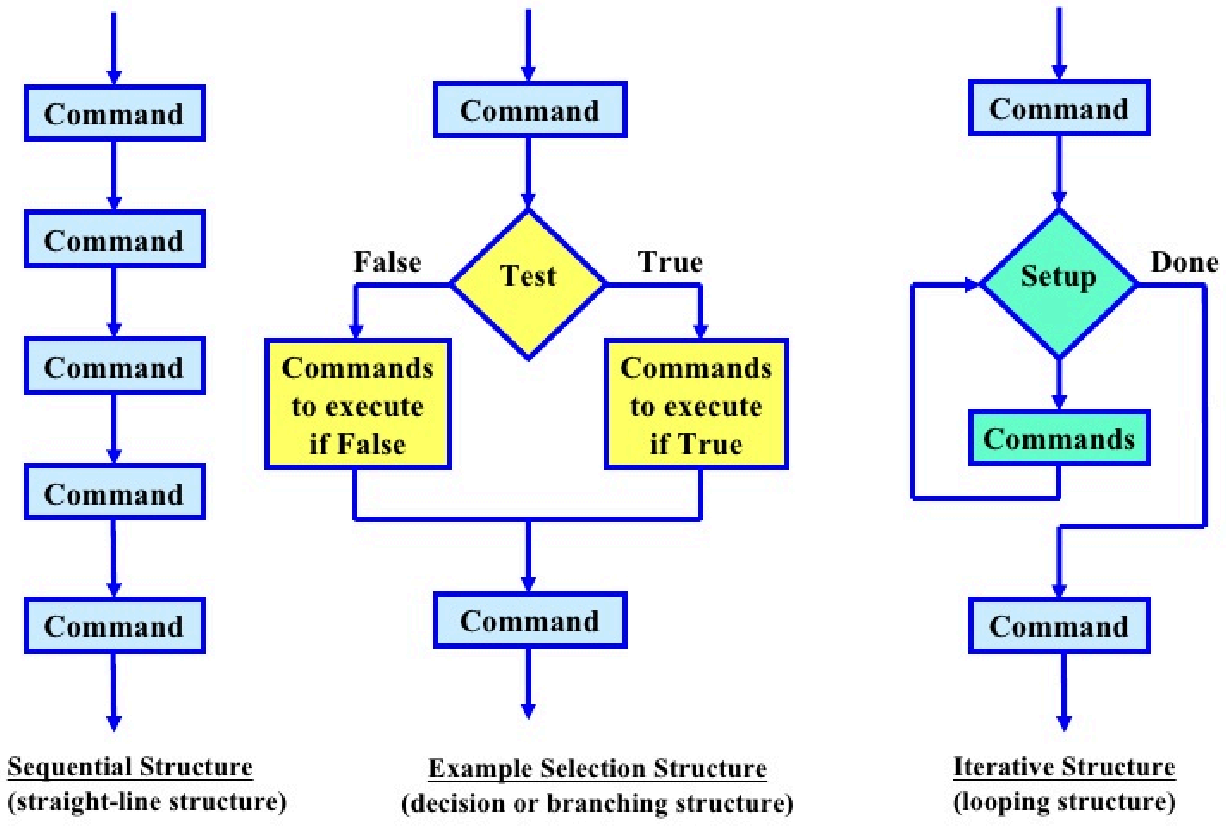
Method overloading: When a class has multiple methods of the same name with different parameters
Ex (Source):



```
Math.min
//Pri
Systen
Systen
```

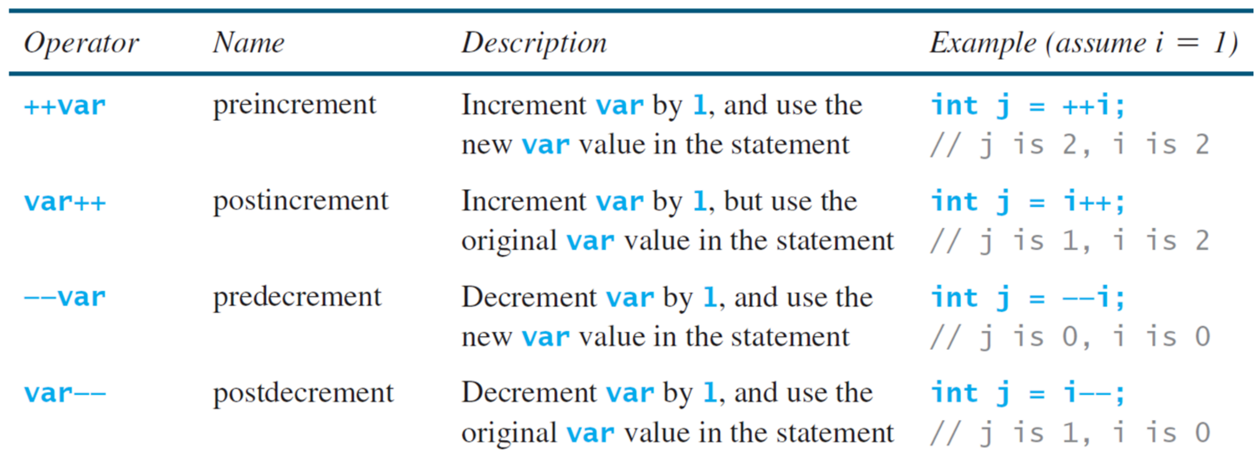
```
min(double a, double b) : double - Math
min(float a, float b) : float - Math
min(int a, int b) : int - Math
min(long a, long b) : long - Math
```

DIFFERENT TYPES OF CONTROL FLOWS (Source):



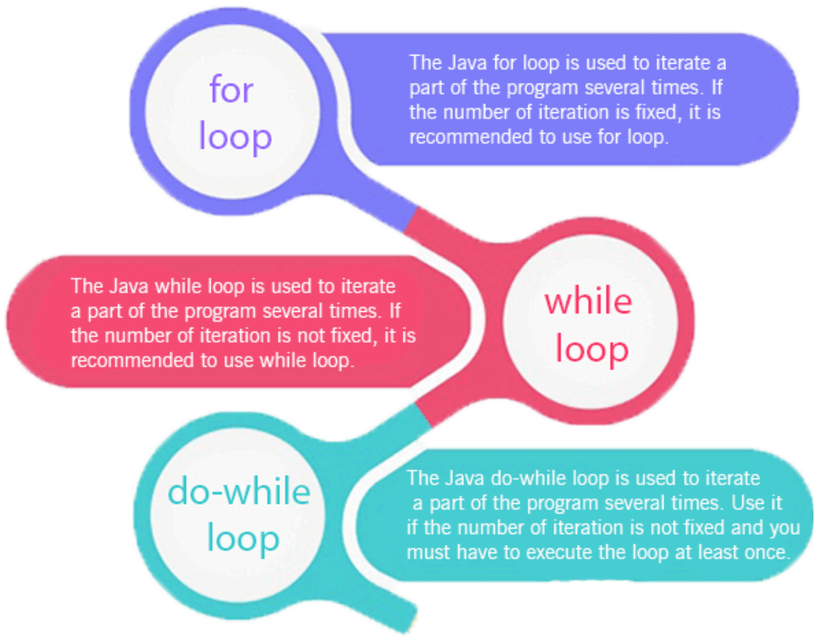
CS1050 - Module 3: Loops, Methods and Software Development

Pre and Post-Increment Operators (Source):



Operator	*Name*	*Description*	*Example (assume i = 1)*
`++var`	preincrement	Increment `var` by `1`, and use the new `var` value in the statement	`int j = ++i;` `// j is 2, i is 2`
`var++`	postincrement	Increment `var` by `1`, but use the original `var` value in the statement	`int j = i++;` `// j is 1, i is 2`
`--var`	predecrement	Decrement `var` by `1`, and use the new `var` value in the statement	`int j = --i;` `// j is 0, i is 0`
`var--`	postdecrement	Decrement `var` by `1`, and use the original `var` value in the statement	`int j = i--;` `// j is 1, i is 0`

Loop Guide (Source):



The Java for loop is used to iterate a part of the program several times. If the number of iteration is fixed, it is recommended to use for loop.

The Java while loop is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop.

The Java do-while loop is used to iterate a part of the program several times. Use it if the number of iteration is not fixed and you must have to execute the loop at least once.

Scope: The area of a program that is in focus/reference (usually for variables)
^ Ex:

```
Public class Example {
        private int number = 1;
}
```

^(Variable number is only in the scope of the Example class)

THE CALL STACK:

The source of memory that keeps track of active methods, operating first in last out (push onto stack, pop off of stack). Stack memory includes (Source):

- Values of parameters and local variables (automatic storage duration)
- Current line of code executing
- Return address to return to calling function

The method on the top of the stack stays on top until its ending brace is reached. When it is, the information for the method is "popped" off the stack and memory local to the stack is removed (deallocated).

Pass By Value: Passing primitive data types from one method to another (i.e the value from one memory location is copied to another).
^ Ex.
gaming(kd1, kd2);
public static void gaming (int ratio1, int ratio2)
-(Gaming copies values from kd1 and kd2 to ratio1 and ratio2, passing by value)

CS1050 - Module 4: Arrays and Write to Files

Arrays: A group of memory locations with all the same data type. When initializing, the declaration of the array holds null (for non-primitive types) and only stores the reference number to the array until the size is specified, which creates enough room on memory on the heap.

- For primitive types (Source):
zero for numeric
\u0000 - for char type - this is Unicode for "null"
false for Boolean type
-Rules for Arrays:
    1. Once array is created, the size is fixed
    2. All elements are the same type
    3. Declaration does NOT allocate memory for the array, just the location of it on the heap
    4. The array must be constructed with the new operator for anything to be done to it
    5. After the array is created (memory allocated), the value stored in the array is the address (which means indexes of the array can be passed by reference using the name of the array)
    6. Attempting to access an element with an index outside the range of the array results in out of bounds runtime error

Pass By Reference: Passing a value by passing a memory address which points to where the value is on the heap.

Algorithm Approaches:

1. Rule-based: A systematic list of instructions to follow

2. Machine-based: Given data, a goal, and feedback, let the object figure out the best course of action

SEARCH ALGORITHMS:
-Note (Source): It is generally not good practice use a return in a for loop. One exception is searching, which helps keep the algorithm from searching in a large list even after the item is found.

- Linear Search: Starts at the beginning of the array (of any order) and searches each index one at a time, returning either the index or a –1. Execution time grows linearly with array size.
- Binary Search: **Given the array is ordered first**, goes to middle of array and compares value, comparing over/under value. If over, takes the middlepoint of the top half of the array and repeats; if under, takes the middlepoint of the bottom half of the array and repeats. Returns index of item or a -1.

EXCEPTION HANDLING:
throws (type of error): A command to list after declaration in order to "account" for the possibility of the error being thrown and to write a case scenario (try/catch) for what to do if it does within the method.

^Ex (Source): public static void main (String[] argos) throws IOException {
}
- If the exception goes up the chain (moves inner to outer) (Source):

1. Exception is passed to calling code
2. If the caller did not embed the method call in a try-catch, the above step is repeated until:
   - Exception is caught or exception is passed to main
3. If the exception reaches main and the call is not embedded in a try-catch block:
   - Program halts (crashes)
   - Trace of the method calls, the exception type, and its error message is printed

(NOTE:
-Exceptions are objects and the root class = java.lang.Throwable
-Any unchecked exceptions (RuntimeExceptions, Error, subsequent subclasses) generally cannot be caught)

Try is the code that's trying to be run that might throw the exception, and catch (exception) is the code executed given the exception does throw.
^Ex (Source):

```
try {
   Scanner fileScanner = new Scanner(new File("grades.txt"));
   System.out.println("File opened successfully.");
   fileScanner.close();
} catch (FileNotFoundException e) {
   System.out.println("Error: Could not find the file.");
}
```

- ORDER OF OPERATIONS FOR TRY/CATCH: The program starts at current method and moves backward from prior method calls looking for a matching case to the exception thrown. Whichever catch is found first will be the code executed, and the program restarts after try-catch block. If no exceptions occur, the code in the catch is skipped. A "finally" clause can be added after all catches in order to run code regardless of if there is an exception.

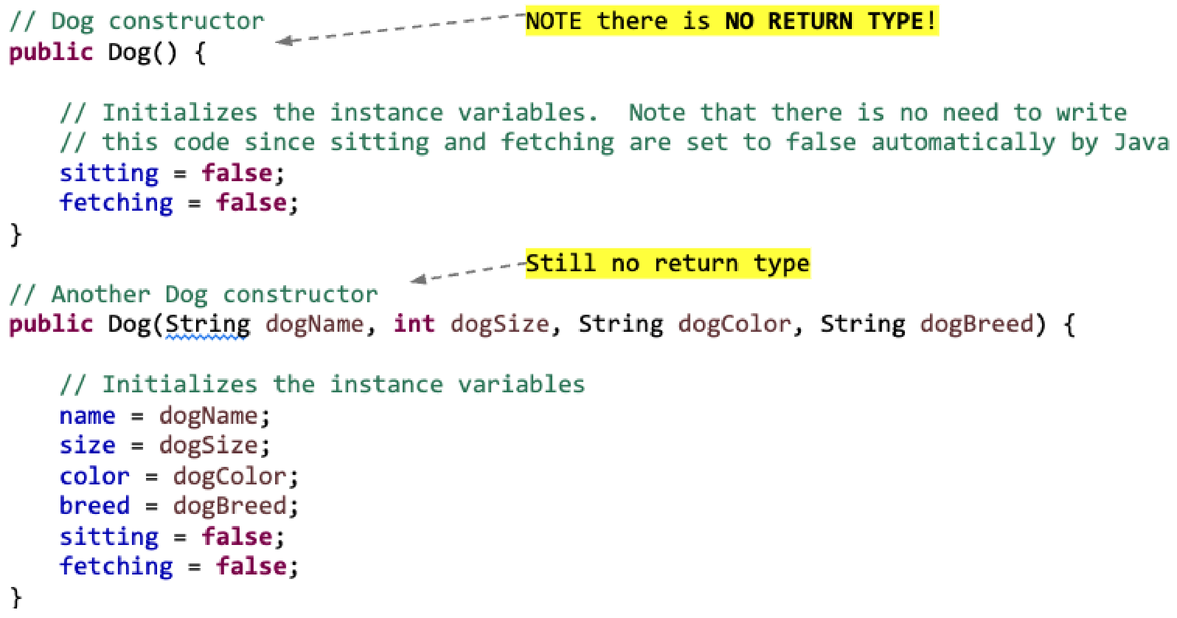
CS1050 - Module 5: Objects and Classes

Objects hold two important details:

1. State (Instance variables)
2. Behavior (Methods)

Constructors: A method that shares the same name as the class with no return type specified and is used to build an outline of the data fields used for the class. Similarly to methods, can be overridden with different parameters under the same name.

^Ex (Source):



```
// Dog constructor
public Dog() {

    // Initializes the instance variables.  Note that there is no need to write
    // this code since sitting and fetching are set to false automatically by Java
    sitting = false;
    fetching = false;
}

// Another Dog constructor
public Dog(String dogName, int dogSize, String dogColor, String dogBreed) {

    // Initializes the instance variables
    name = dogName;
    size = dogSize;
    color = dogColor;
    breed = dogBreed;
    sitting = false;
    fetching = false;
}
```

Instance variable: Variables initialized during the instance of an object

Instance method: Methods initialized during the instance of an object

-Can only be executed through an instance of the class first

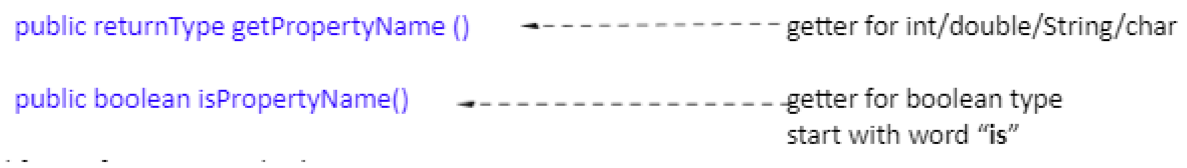
USE WHEN:

- Each object of the class needs an independent copy of the variable
- When instance variables are accessed in a method

Encapsulation: The process of ensuring proper scope for variables/hard values for each class for security, functionality, and maintainability function. Is done mainly in two ways:

1. Add the private modifier to all instance variables, making them hidden outside the class
    - Public: Can be accessed by any other class. Can apply to class or anything of it (methods, variables)
    - Private: Can apply only to anything of the class it belongs to
    - Protected: Ties to packages
    - Default (no specification): Also tied to packages, can be accessed by any class within the same package
2. Create getters and setters to retrieve and set private instance variables (not necessary if variables are only used inside class) (are public methods):
    - Getter (Accessor): method that returns the value of an instance variable
    - Setter (Mutator): method that sets the value of an instance variable

^General Forms Example (Source):

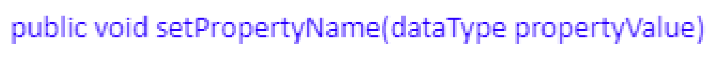


```
public returnType getPropertyName ()

public boolean isPropertyName()
```

getter for int/double/String/char
getter for boolean type start with word "is"

^ Getters general form



```
public void setPropertyName(dataType propertyValue)
```

^ Setters general form

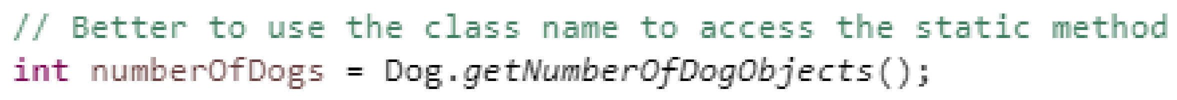
Static (Class) Variables: Variables that exist for every instance of a class. Any modifications done to the value are copies among all instances of the static variable. It is not stored on the heap with the instance variable.

Static (Class) Methods: A method shared by all objects of a class. Can be executed without passing by reference to an instance of the class, or really any instance to even exist.

(NOTE:

- Constants, by default, are static
- Cannot access instance variables or methods
- Use class name over object name when referencing to make it obvious the method is static)

^ Ex (Source)

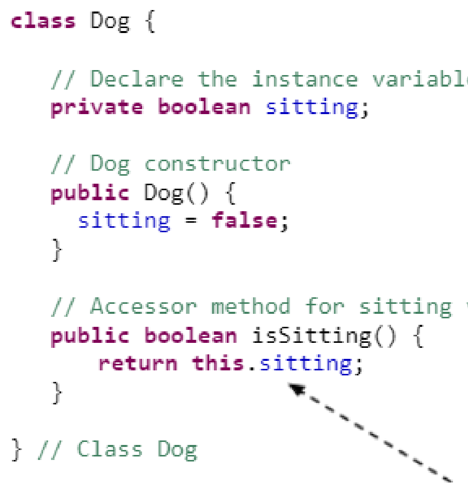


```
// Better to use the class name to access the static method
int numberOfDogs = Dog.getNumberOfDogObjects();
```

USE WHEN:

- Only one copy of the variable is needed and used by all objects
- All objects need to share a variable
- A method is not dependent on a specific instance

this reference: A keyword to refer to the object it is being used in
^ Ex (Source):



```
class Dog {

   // Declare the instance variables for a dog
   private boolean sitting;

   // Dog constructor
   public Dog() {
     sitting = false;
   }

   // Accessor method for sitting variable
   public boolean isSitting() {
       return this.sitting;
   }

} // Class Dog
```

^ (or remove the “this.”, same thing)

Associations (Connections between classes):
- Aggregation: Classes have a loose connection and imply ownership/belonged hierarchy, but the parts are independent in nature
  - ^ Ex: Kitchen and House
- Composition: Classes have a strong connection and hierarchy is dependent in nature between classes
  - ^ Ex: Human and Heart

Inheritance:
- Functionality (state and behavior) is put in a superclass (parent) to which subgenres of the class (children) inherit the functionality of said superclass
^ Ex:
Skyscraper (Superclass)
-storyCount
-residentCount
-meltBeams()
VVVVVVVVV
Skyscraper 1
-storyCount
-residentCount
-meltBeams()
-underRemodeling()

**Note: Method overriding is when subclasses override methods inherited from the superclass. Method signature (name and return type) must match, and the

method overridden must be accessible. Static methods are inherited but can't be overridden. If calling a method in the superclass, super keyword must be used. Use @Override above overridden method to make it more readable.
^ Ex (Source):

```
@Override
public String toString() {
	System.out.println("In CircleFromSimpleGeometricShape toString method
");
	return super.toString() + " radius: " + radius;
}
```

-Contructors **ARE NOT** inherited from superclasses
-Encapsulation rules still apply in scope between super and subclasses
-Must include a default constructor (no-args) in superclass (parent) otherwise you will get an error like this (Source):

```
Exception in thread "main" java.lang.Error: Unresolved compilation
problems:
Implicit super constructor Animal() is undefined. Must explicitly invoke
another constructor
```

Module 1: Object Oriented Fundamentals and Environment

STACK AND HEAP:https://www.geeksforgeeks.org/java/java-stack-vs-heap-memory-allocation/

Heap operator: "."

Add heading for different sections

Add heading for different sections

FIX THIS FIRST

```
public class MainClass {
  public static void main(String[] args) {

  myBook = new Book("Jimmy", "Jimmy's Book", 2025);


  }
}

class Book {
 private string author;
 private string title;
 private int year;

 public Book(String author, String title, int year){
   this.author = author;
   this.title = title;
   this.year = year;


   public string getTitle(){
     return title;
     }

   public string getAuthor(){
     return author;
     }

   public int getYear(){
     return year;
     }
   }
 }
```

Module 2: Algorithm Analysis and Object Oriented Relationships

Big-O (order of): A rating system for algorithms in terms of speed

-A theoretical approach independent of computers and specific input, which estimates time complexity

-Classifies the upper