ENSF 409
Lesson 2 – Notes

Summary

Learning objectives
- Knowledge
    - Basics of classes
    - Distinguish between primitive and non-primitive data types
    - Standard capitalization of classes, methods, and variables
    - Available access modifiers and their visibility
    - Standard naming convention for getter and setter methods
    - Distinguish between class variables and methods, and instance variables and methods
    - Basics of UML class diagrams
    - Constructors
    - Overloading
- Skills
    - Write a simple class, including getter and setter methods
    - Handle command-line arguments
    - Throw an exception
    - Instantiate an object
    - Depict a single class in a UML class diagram
    - Pass multiple variables to a method
    - Use `this`

Related textbook reading
- Chapter 4: Objects and Classes

Important Reminders and Links

Course code repository

https://github.com/abarcomb-work/ENSF409.git

A git tutorial will be provided soon as part of the supplemental course material on tools and practices.

Formatting

Class names begin with an upper-case letter, while method and variable names begin with a lower-case letter. Subsequent words which form the name are written using CamelCase, e.g., `LibraryBook` (a class), `getAuthorName` (a method). Constants should use all capital letters, and underscores between words, e.g., MINIMUM_TEST_SCORE.

You are expected to adhere to Java standards of capitalization in all submitted work.

In Java, getter and setter methods are used to respectively retrieve and assign values to fields in the class. The naming convention for these methods is 'get' or 'set' followed by the variable name, which is then adjusted to CamelCase. For example, `getAuthorName`.

You are expected to adhere to Java naming convention for getter and setter methods.

Quick Guide

Access Modifiers

	Different class, same package	Different package, subclass	Unrelated class, same module	Different module and p1 not exported
`package p1;` `class A {`	`package p1;` `class B {`	`package p2;` `class C extends A {`	`package p2;` `class D {`	`package x;` `class E {`
`private int one;`	🟥	🟥	🟥	🟥
`int two;`	🟩	🟥	🟥	🟥
`protected int three;`	🟩	🟩	🟥	🟥
`public int four;`	🟩	🟩	🟩	🟥
`}`	`}`	`}`	`}`	`}`