Object-Oriented Programming (OOP) Basics

CSCI 161 – Introduction to Programming I

Professor Thomas Rogers

Overview

- Chapter 8 in the textbook *"Building Java Programs"*, by Reges & Stepp.
- Review of OOP History and Terms
- Discussion of Object State and Behavior

OOP History

- Object-Oriented Programming dates back to 1967 and the ***Simula*** computer programming language
  - Used for simulation programs
  - Its elements for holding data were known to be called "objects"
- Today there are many programming languages that are object-oriented, including Java, C++, Python,...

OOP Terms

- **Object-Oriented Programming** – A ***reasoning*** about a program as a set of objects rather than a set of actions.
  - **“Reasoning”** in this context refers to many aspect of programming: design, construction, the view of the overall program and its component solution, …
- **Procedural Programming** – Is the oldest style of programming. Based on functions, procedures and methods, and is what this class has been focused on until now…

OOP Terms (continued)

- ***Procedural vs. OOP*** – ***verb*** vs ***noun***
- **Verb** – Procedural or functional in nature.
  - Action in nature
  - Hinted by method names: *displayPrompt, getNextInteger, readFile.*
- ***Noun*** – *Objects are the building blocks of the program*
  - *As such, they are things, nouns, things actually found in the world and the space of the problem at hand.*
  - *Object names read like nouns: file, person, reader, scanner, etc.*

OOP Terms (continued)

- Java is a class-based OOP language:
  - Class-based in that ***objects are instances of classes***
  - It is through that relationship (instance of a class) that an object gets its type
  - Remember, primitive data types (PDTs) like int, double, char are not instances of classes, thus are not objects once declared:
    - Example: `int num = 5;`
  - However, the String class is just that, a class, so a variable declared from the String class is an object, with methods, and of the type String class:
    - Example: `String str = new String(5);`

OOP Terms (continued)

- **Object** – A programming entity that contains state (data) and behavior (methods).
- **State** – A set of values (internal data) stored in an object.
- **Behavior** – A set of actions an object can perform, often reporting or modifying its internal state.
- **Client** – (or “Client Code”) – Code that interacts with the objects of a class (aka calling code, or caller).

Classes, Objects, State

- Classes, objects and state are not completely new:
  - The variables within a class are what hold its data and define its ***state***.
  - You already develop each program and include one class, the **program class**, and it contains the required **main** method.
  - You have already worked with several classes, including *Scanner, File, String, Integer, Double*…
  - You can also add and define your own classes, as separate .java files, and these act as ***blueprints*** for useful objects within your program.

Classes, Objects, State
(continued)

- **Class vs Object:** A class in your program that provides a blueprint for how objects created from that class will act, behave, etc., since each class defines the following about its objects:
  - The **state** in each object, stored internally in private variables called “fields”.
  - The **behavior** each object can performed is defined by the methods within the class.
  - How to **construct** an object of that class type.

Object Behavior

- There are several; types of methods that define object behavior, and they have names:
  - **Instance Method** – A method inside an object that operates on that object. Examples: String has .length(), Scanner has .hasNext().
  - **Implicit Parameter** – The object itself is an implicit parameter of all Instance methods, meaning the state (data within) is always available within instance methods.
  - **Mutator** – An instance method that modifies the object's internal state.
  - **Accessor** – An instance method that provides information about the state of an object without modifying it.

Object Initialization: Constructors

- **Constructor** – The method of the same name as the class, and it is used to "Construct" the object from the class:
  - A class can have many constructors, all having the same name as the class, but with different parameters.
  - A common use of a constructor is to allow the caller to specify initializing values for state (field) variables of the class (as in supplying X and Y for the Point class)

Constructors – With & Without

```
// Without a constructor
Point p1 = new Point();
p1.x = 7;
p1.y = 2;

// With a constructor
Point p1 = new Point(7, 2);
```

Constructors (continued)

- The **default** constructor is inserted by the JVM if no other constructors are defined in your classes code.
- It is a good practice to always include a constructor, even if it accepts no parameters and is used only to initialize state (field) variables to specific, known values.
- **Warning** – Be careful NOT to include void as the return data type of a constructor. Constructors have no return type and including void out of habit will indicate to Java that the method is “just another method” and not a constructor.
- **This** – The keyword **this** refers to the object derived from the class and is a convenient way within constructors and methods to reference object fields.

More to come...

- More advanced OOP topics in the next lecture...