CSCI 136 Data Structures & Advanced Programming

Lecture 3

Spring 2020

Instructors: Bill & Dan

Administrative Details

- Lab today in TCL 216 & 217a
  - Lab is due by 8pm Monday
    - To submit: Push your repository to github (see lab handout)
- Lab design docs are usually "due" at beginning of lab
  - Written design docs will be required for most labs
  - You'll discuss with another student at start of lab
  - Several implementation options
    - Some may be better than others.... talk it out with each other and with us!
- Since we are still getting into things, we will talk about design docs and begin with a Nim example

Last Time

- Some Java Examples (Hellow.java, Sum.java)
  - Entering, editing, compiling, running programs
  - User input: Scanner, argv[]
  - Primitive and numeric types
  - System.out.prinln(…)

Today' s Outline

- Objects!
  - OOP is a powerful way to organize your code
  - What features does Java provide to support OOP?
- Design documents
  - Debug our logic before our code
    - Nouns: variables
    - Verbs: methods
- Nim
- Lab 1 Demo

Object-Oriented Programming

- Objects are building blocks of Java software
- Programs are collections of objects
  - Cooperate to complete tasks
  - Represent “state” of the program
  - Communicate by sending messages to each other
    - Through *method invocation*

Object-Oriented Programming

- Objects can model:
  - Physical items - dice, board, dictionary
  - Concepts – date, time, words, relationships
  - Processing - sort, search, simulation
- Objects contain:
  - State (instance variables)
    - Attributes, relationships to other objects, components
      - Letter value, grid of letters, number of words
  - Functionality (methods)
    - Accessor and mutator methods
      - addWord, lookupWord, removeWord

Object Support in Java

- Java supports the creation of programmer-defined types called *class types*
- A *class declaration* defines data components and functionality of a type of object
  - Data components: *instance variable (field) declarations*
  - Functionality: *method declarations*
  - *Constructor(s)*: special method(s) describing the steps needed to create an object (*instance*) of this class type

A Programming Principle

Use constructors to initialize the state of an object, nothing more.

- State: instance variables
- Frequently constructors are short simple methods
- More complex constructors will typically use helper methods.
- You constructors can call other constructors to reuse code

Access Modifiers

- `public` and `private` are called *access modifiers*
  - They control access of other classes to instance variables and methods of a given class
  - `public` : Accessible to all other classes
  - `private` : Accessible only to the class declaring it
- There are other levels of access that we'll see in more detail later (e.g., `protected`)
- Data-Hiding (encapsulation) Principle
  - Make instance variables `private/protected`
  - Use `public` methods to access/modify object data

Nim Design

- What is “the data”?
  - How should we represent a single pile?
  - How should we represent all of the piles?
- What questions will we ask of the data?
  - void makeMove(whichPile, howMany)
  - boolean isLegalMove(whichPile, howMany)
  - toString() ← We’ll talk about later
  - isGameOver()
  - whosTurnIsIt()
  - swapTurns()

LET’S IMPLEMENT NIM!

Nim Implementation

- Of Note:
  - toString() let's us separate the representation of the board from the display.
    - Do NOT keep two versions of the game state.
    - Instead, generate a String representation on demand.
    - Why?
  - Multiple constructors; some call each other
    - Reuse code with overloading
  - Replicate this design process for Lab 1 Coinstrip

At This Point, Ready for Lab 1!

- Basic Java syntax
  - Java types: primitives, arrays, classes
  - Control structures: branches, loops, functions
  - Programmer-defined types: class types
  - Essential Java classes:
    - `String`
    - `Random`
    - `Scanner` (`import java.util.Scanner;`)
  - We will learn to appreciate Unix, git, and the command line together
    - You will master them someday (it takes time!)

COINSTRIP DEMO

CoinStrip Design

- How to store game state? Think about:
  - Space needs
  - Time to find coin
- Useful methods?
  - void makeMove(whichCoin, howFar)
  - boolean legalMove(whichCoin, howFar)
  - toString() ← We'll talk about later
- What, if anything, did lab description omit?
  - Form of "game board" to show players