CSCE 240: Advanced Programming Techniques
Lecture 6: Object Oriented Concepts, HW 2 (Review)

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

27TH JANUARY 2022

Carolinian Creed: "I will practice personal and academic integrity."

Credits: Some material reused with permission of Dr. Jeremy Lewis. Others used as cited with thanks.

Organization of Lecture 6

- Introduction Section
  - Recap of Lecture 5
  - TA and SI Updates

- Main Section
  - Home work 2 – peer review
  - Concept: OO Thinking
  - Concept: Classes, Objects
  - Background for project: Chatbots

- Concluding Section
  - About next lecture – Lecture 7
  - Ask me anything

Introduction Section

Recap of Lecture 5

- We discussed
  - the concept of user-defined types
  - the concepts of static and dynamic memory allocation

- Discussed Home Work 2 (due Thursday, Jan 27)
  - Peer evaluation in class

- Discussed Programming Assignment #1 (due Thursday, Feb 3)

Updates from TA, SU

- TA update: Yuxiang Sun (Cherry)

- SI update: Blake Seekings

Main Section

Peer Review of Home Work 2

Feedback from Peer Reviewing of HW 1

- Home works marks will be best of 4, out of 6

- The same person should not be the coder, reviewer and tester (**conflict of interest**)

- For a coder, the same person should not be the reviewer/ tester for ALL the home works
  - Change seats sometimes
  - Don't miss the opportunity to get more feedbacks

- Do not try to just give marks. Give feedback
  - Coders may be asked to show code to class to illustrate good practices or critical issues
  - Be both reviewer and tester. If you are just one, you will get marks for just one.

- Weightage give as follows
  - 60%: coding
  - 20%: how well reviewing of others' code is done
  - 20%: how well testing of others' code is done

Programming Home Work (#2) – C++

- Write a program called GeometricPropertyCalculator.
  - The program reads an input file (called input.txt). Each line in the file contains dimensions of a geometric shape – rectangle, shape and triangle. Specifically:
    - For rectangle, it contains – RECTANGLE <length-in-cm> <breadth-in-cm>
    - For circle, it contains – CIRCLE <radius-in-cm>
    - For triangle, it contains – TRIANGLE <side-1-in-cm> <side-2-in-cm> <side-3-in-cm>
  - The user specifies the property to calculate as argument to the program: 1 for AREA and 2 for PERIMETER
  - The program writes output lines to an output file (called output.txt) for each shape that it reads and the property – AREA or PERIMETER.
    - For example, for RECTANGLE and property as AREA, the program should write – RECTANGLE AREA <calculated value>

  - Write GeometricPropertyCalculator in C++
    - It should support RECTANGLE, CIRCLE and TRIANGLE
    - It should support properties AREA and PERIMETER
    - If there is insufficient information, the program should give an error. E.g. TRIANGLE AREA "Not enough information to calculate"

Programming Home Work (#2) – C++

- Code guidelines
  - Have sub-directories in your folder
    - src sub-folder, (or code) for code
    - data sub-folder, for input.txt and output.txt
    - doc sub-folder, for documentation on what the code does or sample output

- Hint
  - Area
    - Rectangle: length x breadth
    - Circle: pi * r^2
    - Triangle: -
  - Perimeter
    - Rectangle: 2 * (length + breadth)
    - Circle: 2 * pi * r
    - Triangle: sum of sides

Peer Review: Homework Assignment #2

1. Go to spread sheet and on "Homework Assignments - Peer Review" tab. Go for today's date

2. Go to the row with your name

3. Peer review (10 mins)
   1. Enter roll number of person on your **LEFT** under "ID of code reviewer"
   2. Share code for the reviewer to see
   3. Reviewer: enter review (1-5)
   4. **Note**: negotiate – review code of neighbor or get own's code reviewed

4. Peer test (10 mins)
   1. Enter roll number of person on your **RIGHT** under "ID of code tester"
   2. Share command line for the tester to see
   3. Tester: enter review (1-5)
   4. **Note**: negotiate – test code of neighbor or get own's code tested

Peer Reviewing Guideline (10 mins)

- Look out for
  - Can you understand what the code is doing ?
  - Can you explain the code to someone else (non-coder) ?
  - Can you spot possible issues without running it?
    - Are the variables initialized ?
    - Are files closed?
    - Is their unnecessary code bloat ?

- What not to judge
  - Usage of language features, unless they are inappropriate

- Assign rating
  - 1: code not available
  - 2: code with major issues
  - 3: code with minor issues
  - 4:
  - 5: no issues

Peer Testing Guideline (10 mins)

- Look out for
  - Does the program run as the coder wanted it to be (specification) ?
  - Does the program run as the instructor wanted it to be (requirement - customer) ?
  - Does the program terminate abruptly ?
  - Any special feature?

- What not to judge
  - Person writing the code

- Assign rating
  - 1: code not available
  - 2: code runs with major issues (abnormal termination, incomplete features)
  - 3: code runs with minor issues
  - 4:
  - 5: No issues

Discussion

- Peer Code Reviewing

- Peer Testing

Concept: Object Oriented Thinking

World as Procedures or Objects ?

- View 1: World as procedures
  - Data as inputs and outputs
  - Functions with arguments that manipulate data
  - Control-based view of the world

- View 2: World as objects
  - Objects, representing concepts in the world, with properties and methods
  - Data (object) passed to (their or others) objects' methods to operate on them or get modified
  - Data-based view of the world

Relation to User Defined Types

- When we start solving real world problems, we often use a set of information together
  - Examples:
    - Name = {title, first-name, middle-name, last-name, suffix}
    - Address = {Street name, Number, City, State, Zip code}
  - Need not be of the same type as a language's pre-determined / basic data types

- May be of the same or different basic data types

- Note:
  - In UDTs, there is no provision to define function specifically for a UDT
  - All data members are public (i.e., anyone can edit)

Concept: Classes and Objects

Terminology

- Class: a well-defined concept
  - Could be related to a physical or abstract idea
  - Something which can be identified in a domain that is of interest
  - Examples: People, Students, Lectures, Dreams, Exception, Programming Languages

- Object: an instance of a class

  - Examples:
    - People: US-President, Dalai Lama, R2D2
    - Exceptions: Known, Unknown, IO-Exception, Memory-Exception

Why Objects / Classes

- Ease of writing and maintaining code
  - Abstraction helps in communication
  - Code is easier to understand
  - Code becomes easier to test

- No impact on code's executional performance

- C++ concepts
  - Struct
  - Class (Object Oriented Programming)

Goals similar to user defined types and more!

- Encapsulation
  - Easier to understand

- Abstraction
  - Easier to understand

- Separation of concerns
  - Write code with a focus on an issue (concern) at a time; do not overload issues
  - Makes code reuse easier

- Drive code reuse
  - Code improves with reuse; drive quality
  - Brings productivity

Programming Languages: Classes and Objects

- Class: general concept that has properties that all instances share
  - Data members
  - Functions

- Object: specific instances
  - Reuses the definitions of data members (variables) and methods of the class
  - Data types and values of the member variables can be cast/ updated

Code Demo – C++

- **Function:** demoPersonClass()

- Note
  - Separate file for class specification
  - Separate file for class implementation
  - Explicit destructor

Code Demo – Java

- **File / Class :** L6ClassDemo.java / L6ClassDemo

  - File name has to be the same as class name
  - Three different constructors with varying arguments (i.e., constructors can be overloaded)

Code Demo – Python

- **File / Class:** L6_ClassDemo / PersonInfo

- Note
  - One constructor but with default values for parameters

Discussion: Course Project

Course Project – Assembling of Prog. Assignments

- **Project**: Develop collaborative assistants (chatbots) that offer innovative and ethical solutions to real-world problems ! *(Based on competition - https://sites.google.com/view/casy-2-0-track1/contest)*

- Specifically, **the project will be building a chatbot that can answer questions about a South Carolina member of state legislature from**:
https://www.scstatehouse.gov/member.php?chamber=H

  - Each student will choose a district (from 122 available).
  - Programming assignment programs will: (1) extract data from the district, (2) process it, (3) make content available in a command-line interface, (4) handle any user query and (5) report on interaction statistics.

Discussion: Nature and Simplifications

- Once you select a district, the elected legislator is fixed.

- Some simplifications
  - **Download local copy** v/s web query
  - **Read static content first**
  - **Handle a subset of content**
  - **Have default handling for questions** the chatbot does not understand

- Do project in a language you are most comfortable with

- Use all advanced programming concepts to simplify coding

Discussion: Chatbot Loop

- Input: from user (called utterance)
  - Problem specific query (i.e., about district chosen)
  - Chitchat
  - Unrelated

  Handling different data types

- Output: from system (response)
  - Handle unrelated
  - Handle chitchat
  - Answer to query

  Show formatted content!

- **Do it until user say over!**

Concluding Section

Lecture 6: Concluding Comments

- We experienced peer review on home works

- Discussed the concepts of mixed types

- Discussed formatted printing

About Next Lecture – Lecture 7

Lecture 7: Object Oriented Continued, UML Notations

- OO support in C++
  - Methods and encapsulation
  - Different access restrictions

- Unified Modeling Language Notations

- Chatbot basics