CSCE 240: Advanced Programming Techniques

Lecture 10: Object Oriented Concepts – Polymorphism, Regular Expressions, HW 3 (Review)

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

8TH FEBRUARY 2024

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 10

- Introduction Section
  - Recap of Lecture 9
- Main Section
  - Concept: Regular expressions
  - Concept: Polymorphism
  - Discussion: Project discussion
- Concluding Section
  - About next lecture – Lecture 11
  - Ask me anything

Introduction Section

Recap of Lecture 9

- We relooked at the concept of inheritance
- We discussed home work assignment #3 – due today - Thursday (Feb 8)
- We discussed programming assignment (PA) #2 – due next Thursday (Feb 15)

About Programming Languages in Course

- C++ is the main language for the course.
  - Used to demonstrate concepts and expect everyone to know it at the level that they can do peer evaluation and testing of each other's code in **home assignments**.
  - **For projects and programming assignments**, students have option to code in Java or Python as well.
- Cross-language understanding of concepts
  - Code in multiple languages is sometimes shown to demonstrate generality of concepts and specific peculiarities in implementation
  - UML diagrams will used to conveyed cross-language concepts as well

In quizzes,

- **Questions will be about concepts, pseudo-code and UML diagram**.
- C++ code fragments may be shown or asked to be written, but they do not have to be running code. The quizzes will be in class and can be done on paper or a text editor like Google doc.

Main Section

Peer Review of Home Work 3

Programming Home Work (#3) – C++

Home Work #2

- 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"

Home Work #3

- Build a program called **OOGeometricPropertyCalculator**
  - Your new code will do the same as Home Work#2 but with OO design
  - It will have **4 classes: Shape – the parent, and its three children - Rectangle, Circle and Triangle**
  - Shape will have three members: **area**, **perimeter** and **errorMessage**; and at least **three functions getArea(), getPerimeter() and getErrorMessage().**
  - In your code, there will be a utility file (OOGeometricPropertyCalculator.cpp) with main() and will call the classes and functions. You can choose to have one or more files for the classes.
    (E.g, For the 4 classes, 4 headers + 4 .cpp files).
  - **You will also draw UML class diagrams for it**
- Functionality Reminder
  - 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.

Programming Home Work (#3) – C++

- Code guidelines for the OO code you will write
  - 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.

- In documentation
  - **Have a UML class diagram for the classes**
  - Observe how long was the code earlier and now. If you have to add a new functionality (like getVertices() to get all the vertices in a shape), how easy or hard will it be in HW2 code or HW3 code?

Peer Review: Homework Assignment #3

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; documentation (UML not available)
  - 3: code with minor issues ; documentation not explainable (UML)
  - 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; documentation (UML) not consistent with behavior
  - 4:
  - 5: No issues

Discussion

- Peer Code Reviewing
- Peer Testing

Concept: Regular Expressions

What are Regular Expressions (RegExs)

- A mechanism for searching in springs
  - Programming-language independent
  - Declarative – specifies what to do but not how
- How it works
  - Step 1; Define a regex pattern
  - Step 2: Apply it on a string
  - Step 3: Get result
- Historical details: https://en.wikipedia.org/wiki/Regular_expression

Example #1

- **String**: "Advanced Programming Techniques”
  - Is ”**Prog**” in this string ? => string match
  - What are the characters after “Prog”? => string match followed by sub-string extraction

- **Output** - Yes
  - string – ’**Advanced Programming Techniques**' matches pattern - **(.*)(Prog)(.*)**

- Nature of problem => Solving the problem
  - Is ”Prog” in this string ? => string match
  - What are the characters after “Prog”? => string match followed by sub-string extraction

Example #2

- String 1: "**AdvProg.pdf**"
  - Is this a valid file name?
- String 2: " **Adv Prog.pdf** "
  - Is this a valid file name?

  - Pattern = **"[a-zA-Z_][a-zA-Z_0-9]*\\.[a-zA-Z0-9]+"** ;

- Output – Yes (String 1) and No (String 2)

Review: Regular Expression

Metacharacter	Explanation
^	Matches the starting position within the string
.	Matches any single character
[ ]	Matches a single character that is contained within the brackets
[^]	Matches a single character that is not contained within the brackets.
$	Matches the ending position of the string
*	Matches the preceding element zero or more times
+	Matches the preceding element one or more times
|	Separates choices

Regex	Matches any string that
hello	contains {hello}
gray|grey	contains {gray, grey}
gr(a|e)y	contains {gray, grey}
gr[ae]y	contains {gray, grey}
b[aeiou]bble	contains {babble, bebble, bibble, bobble, bubble}
[b-chm-pP]at|ot	contains {bat, cat, hat, mat, nat, oat, pat, Pat, ot}
colou?r	contains {color, colour}
rege(x(es)?|xps?)	contains {regex, regexes, regexp, regexps}
go*gle	contains {ggle, gogle, google, gooogle, goooogle, ...}
go+gle	contains {gogle, google, gooogle, goooogle, ...}
g(oog)+le	contains {google, googoogle, googoogoogle, googoogoogoogle, ...}
z{3}	contains {zzz}
z{3,6}	contains {zzz, zzzz, zzzzz, zzzzzz}
z{3,}	contains {zzz, zzzz, zzzzz, ...}

Example Source: https://cs.lmu.edu/~ray/notes/regex/

Regular Expression in C++

- Part of stdlib
- Code example: https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class9and10_C%2B%2B_OOAdv/src/Class9and10_C%2B%2B_OOAdv.cpp
  - Function: demoRegex()

References:
https://www.softwaretestinghelp.com/regex-in-cpp/
https://www.cplusplus.com/reference/regex/

Implementation: Finding Words in Python

- Python has extended Regex specifications for convenience
- Useful for
  - Matching patterns
  - Information extraction
  - Content manipulation (e.g., substitution)
  - Error (e.g., spelling) correction

```
data = "The CSCE 771 course is taught at
          University this Fall!”
pattern = "[tT]+\w"
m = re.findall(pattern, data)
print(m)
```

```
['Th', 'ta', 'ty', 'th']
```

Details: https://docs.python.org/3/library/re.html

Regex Python Code Examples

- More regular expression examples
  - https://github.com/biplav-s/course-d2d-ai/blob/main/sample-code/l20-text-overview/WordLesson-Examples.ipynb

Regular Expression in Java

- Part of java.util.regex library
- (Matcher, Pattern) are the most important classes

- **Note**: same regex expression (pattern) works across languages

Concept: Polymorphism

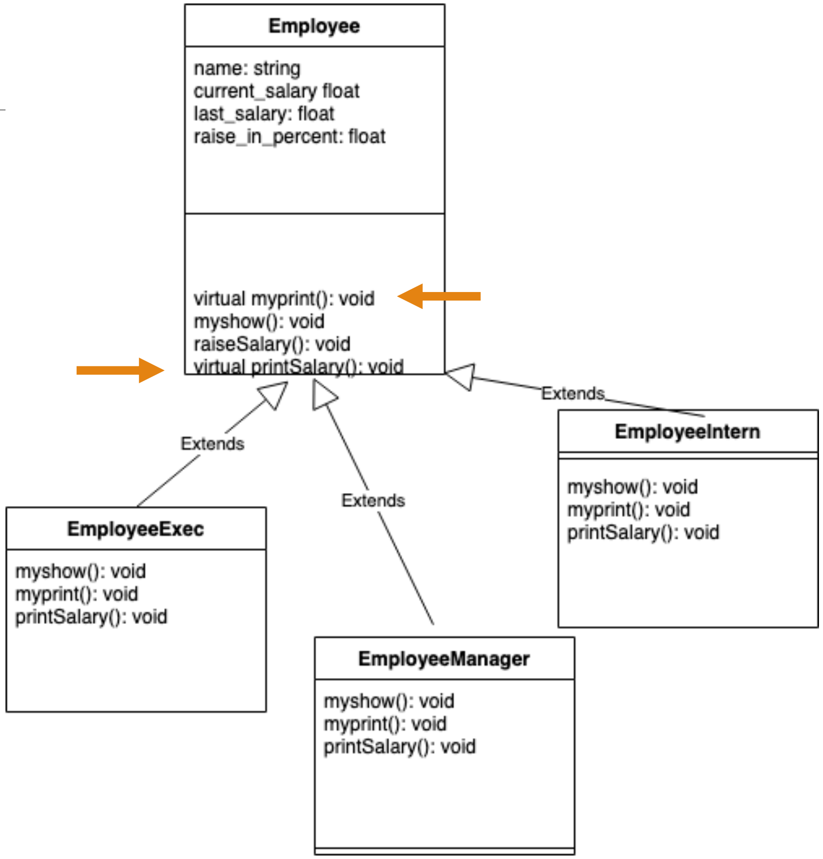
“Multiple shapes”

What is Polymorphism ?

- A class “inheriting” or reusing **characteristics** from another, existing class, dynamically depending on how the method is declared !
- In contrast, inheritance discussed until now was static

Why Use Polymorphism ?

- Promote reuse
- Make code understandable, improve maintainability
- Promote security and data integrity
- Improve testing
- Improve code development productivity
- **Context-dependent customization of inheritance**



Credits: Based on code at
- https://www.geeksforgeeks.org/polymorphism-in-c/
- https://www.geeksforgeeks.org/virtual-functions-and-runtime-polymorphism-in-c-set-1-introduction/

How to Use Polymorphism ?

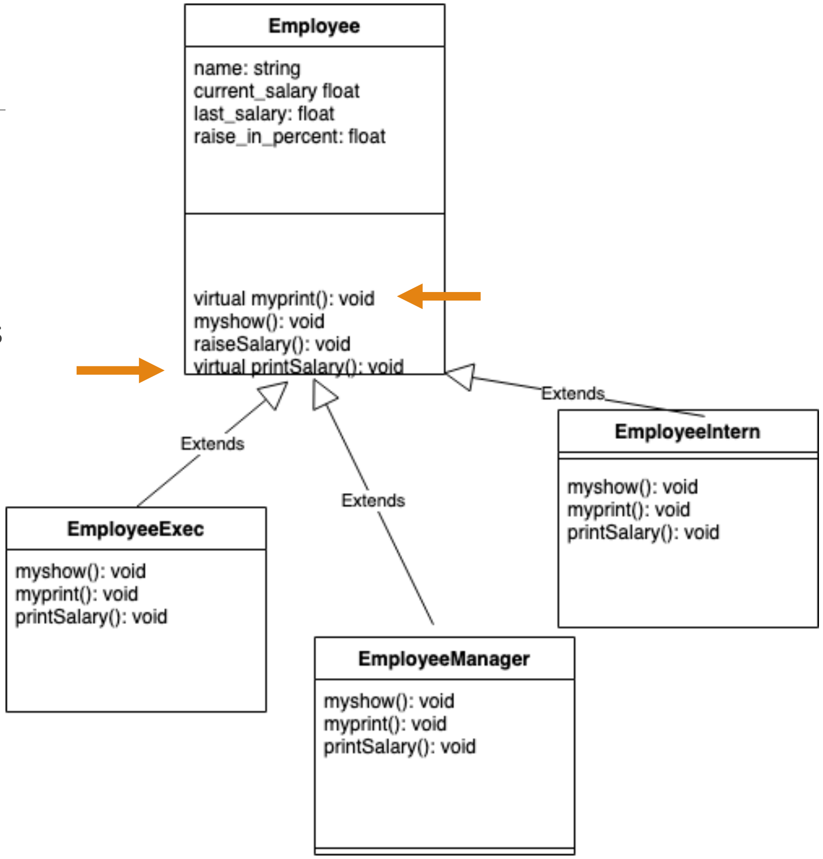
- Language independent syntax
- Illustration
  - 4 classes; 1 base, 3 derived
  - Basic: no data members; myshow() and myprint() functions
  - Advanced: 3 data members, printSalary() function

```
Employee: myprint base class
Employee: myshow base class

EmployeeManager: myprint derived class
Employee: myshow base class

EmployeeIntern: myprint derived class
Employee: myshow base class

EmployeeExec: myprint derived class
Employee: myshow base class
```



Employee

name: string
current_salary float
last_salary: float
raise_in_percent: float

virtual myprint(): void
myshow(): void
raiseSalary(): void
virtual printSalary(): void

EmployeeExec

myshow(): void
myprint(): void
printSalary(): void

EmployeeManager

myshow(): void
myprint(): void
printSalary(): void

EmployeeIntern

myshow(): void
myprint(): void
printSalary(): void

Key Points - Polymorphism

1. The method must appear in a class that is part of an inheritance hierarchy
2. The method must declared virtual in the base class at the top of the hierarchy
3. Derived classes override the behavior of the inherited virtual methods as needed.
4. Clients must invoke the method via a pointer (or reference) to an object, not directly through the object itself

Code:
https://github.com/biplav-s/course-adv-proglang/tree/main/sample-code/CandC%2B%2B/Class9and10_C%2B%2B_OOAdv/src

Credit: Fundamentals of Programming C++, Richard L. Halterman

Notes on Polymorphism

- Support for Polymorphism is not uniform across languages
- C++ is most expressive; controlled by virtual; allows dynamic binding (change of behavior)
- Java and Python have limited support; does static binding

Discussion: Course Project

Course Project – Knowing About Companies

- **Project**: Develop collaborative assistants (chatbots) that offer useful information about companies
- Specifically, use the EDGAR dataset on companies at: https://www.sec.gov/edgar/searchedgar/companysearch.
  - For Apple, it is: https://www.sec.gov/edgar/browse/?CIK=320193&owner=exclude
- **Each student will choose two companies (from thousand available).**
- Programming assignment programs will: (1) extract data about two companies from 10-k, (2) process it, (3) make content available in a command-line interface, (4) handle any user query and (5) report on interaction statistics.

Core Programs Needed for Project

- Prog 1: extract data from the district
- **Prog 2: process it (extracted data) based on questions**
- Prog 3: make content available in a command-line interface
- Prog 4: handle any user query and
- Prog 5: report statistics on interaction of a session, across session

Content Reference: Queries for (Answers) Data We Have

- What does the (company) do? // Answers in Part 1
  - What is the (company's) business?
  - What are (company's) risk factors?
  - What does (company) own?
  - …
- Where does (company) operate? // Answers in Part 2
  - What has (company) disclosed?
- How is (company) structured? // Answers in Part 3
  - Who is (company's) CEO?
  - How much does (person) earn?
  - …
- What was in (company) statements? // Answers in Part 4
  - …

Concepts: 10-K, Parts, Items

Parts

- Part 1: Business Background and Risks
  - Item 1: Business
  - Item 2: Risk factors
  - Item 3: Properties
  - Item 4: Legal Proceedings
- Part 2: Operations and Disclosures
  - .. Market
  - .. Disclosures
- Part 3: Company Structure
  - Directors
  - Compensation
- Part 4: Financial Statements
  - Statements

Programming Assignment # 2

- Goal: **process extracted text based on questions**
  - Language of choice: Any from the three (C++, Java, Python)

- Program should do the following:
  - Take input from a local file which has content obtained from Prog#1 (when **company** name given as input)
  - Given an information type as input, the program will return its content
    - Examples: what is company's risk factors? What does company's CEO earn?
    - Input type can be given as command line argument.
      Examples:
      - prog2processor –t "what are **IBM's** risk factors?" // Tell about company
      - prog2processor –t "all information" // Get all info for a company
  - For demonstrating that your program works, have a file called "test_output.txt" showing the set of supported commandline options and output in the doc folder.

Concepts: 10-K, Parts, Items

Parts

- Part 1: Business Background and Risks
  - Item 1: Business
  - Item 2: Risk factors
  - Item 3: Properties
  - Item 4: Legal Proceedings
- Part 2: Operations and Disclosures
  - .. Market
  - .. Disclosures
- Part 3: Company Structure
  - Directors
  - Compensation
- Part 4: Financial Statements
  - Statements

- Code organization
  - Create a folder in your GitHub called "prog2-processor"
  - Have sub-folders: src (or code), data, doc, test
  - Write a 1-page report in ./doc sub-folder
  - Send a confirmation that code is done to instructor and TA, and update Google sheet

Hint: Use Regex for Information Extraction

- Use regex to find information types and sub-types
- Make a set of patterns (regex) for information of interest
  - Think of patterns as “rules” .e.g., phone number comes after “Phone”
  - Patterns for information types and sub-types
  - Ordering of rules may matter if multiple rules match
- If pattern is found, extract content of interest near it using string operation

Concluding Section

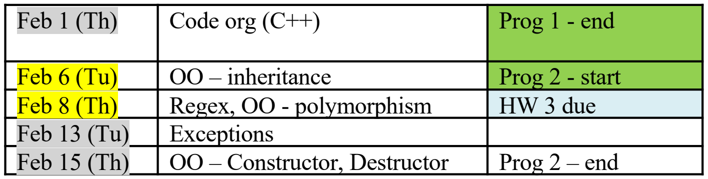
Lecture 10: Concluding Comments

- Understood concept of regular expressions
- Saw how regex could be helpful for Prog. Assignment #2 (due Feb 15)
- Looked at the concept of Polymorphism

About Next Lecture – Lecture 11

Lecture 11: Exceptions

- Concept of errors, exceptions
- Controlling program behavior when it may give errors/ exceptions



Feb 1 (Th)	Code org (C++)	Prog 1 - end
Feb 6 (Tu)	OO – inheritance	Prog 2 - start
Feb 8 (Th)	Regex, OO - polymorphism	HW 3 due
Feb 13 (Tu)	Exceptions	
Feb 15 (Th)	OO – Constructor, Destructor	Prog 2 – end