CSCE 240: Advanced Programming Techniques
Lecture 11: Exceptions, Error Handling

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

13TH 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.

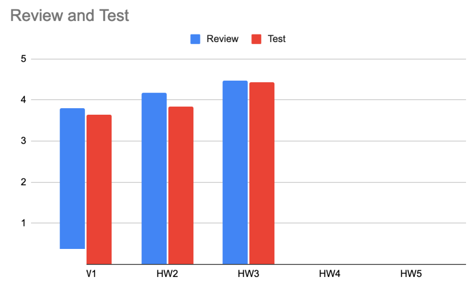
Organization of Lecture 11

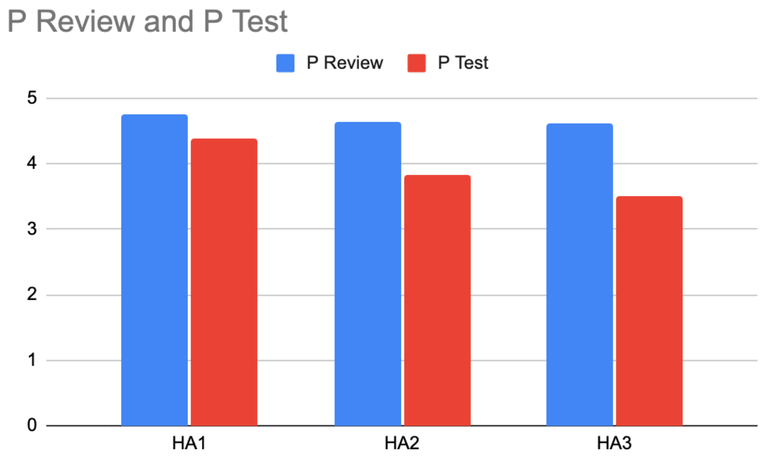
- Introduction Section
  - Recap of Lecture 10
  - Announcements
- Main Section
  - Concept: Errors
  - Concept: Exceptions, for error handling
  - Discussion: Project
- Concluding Section
  - About next lecture – Lecture 12
  - Ask me anything

Introduction Section

Recap of Lecture 12

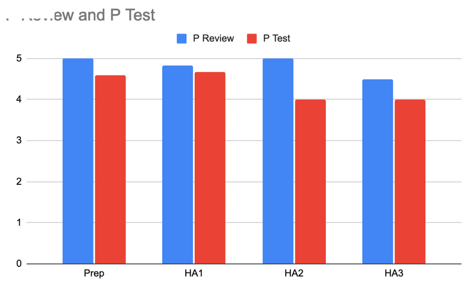
- Peer review of HW3
  - Slight fall in quality of code OR better peer testing
  - Caveat: Sample size is small and varies every class
- Review of Inheritance
  - Concept: Inheritance Type
- Review of Polymorphism





Reference:
Spring 2022
(45+ students)

Reference:
Spring 2023
(6 students)



Spring 2023

Announcements

- Blackboard tour: where to see course recordings
- Programming Assignment #1: marks will be posted soon

Programming Assignment # 1

- **Goal**: extract data from the companies of choice
  - Language of choice: Any from the three (C++, Java, Python)
- Program should do the following:
  - Take company / 10-k page (URL) as input
  - Read content about the 10-k page
    - a local text version of the report page // Store it as file with names <companyname>-<quarter-year>.txt
      // Optional: get reports for multiple quarters (say 3). Keep them as separate files with names <companyname>-<quarter-year>.txt
  - **Identify how may parts are there in the report** //Hint: You can search for a hardcoded string/ pattern
  - Report statistics of content: lines, words, chars, and parts.
  - Write content out in an output file formatted with indentation
- Code organization
  - Create a folder in your GitHub called "prog1-extractor"
  - 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 update Google sheet

PA: Code Reviewing Rubric Used

- Look out for
  - Can one understand what the code is doing ?
  - Can one explain the code to someone else (non-coder) ?
  - Can one 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 (out of 100 -/+)
- -100: code not available
- -80: code with major issues
- -60: code with minor issues
- -20:
- 0: (full marks): no issues
- +20: special features

PA: Code Testing Rubric Used

- 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 ?
  - Is there a hardcoding of directory ? Paths should be relative to code base directory.
  - Any special feature?
- What not to judge
  - Length of documentation. It can just be short and accurate.
  - Person writing the code

Assign rating (out of 100 -/+)
- -100: code not available
- -80: code with major issues (e.g., abnormal termination, incomplete features)
- -60: code with minor issues
- -20:
- (full marks): no issues
- +20: special features

Main Section

Concept: Errors

What is an Error ?

- **Error**: Anything that is not as-expected
- Errors at different levels

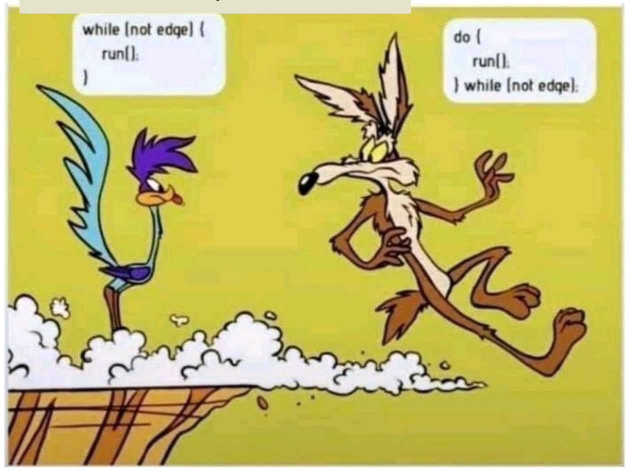
	Types*
• **Conceptual**: at the problem and solution approach level	[interface error, logic error]
• **Implementation**: in the program	[syntax error, compilation error, arithmetic error]
• **Ongoing / runtime**: while running	[resource error, runtime error]

*Credit: https://textexpander.com/blog/the-7-most-common-types-of-errors-in-programming-and-how-to-avoid-them

Why There are Errors ?

- Conceptual: at the problem and solution approach level
  - Customer did not make the requirement clear (requirement)
  - Developer did not understand the problem clearly (specification)
- Implementation: in the program
  - Poor coding
  - Programming concepts were used wrongly
  - Test cases were exhaustive
- Ongoing / runtime: while running
  - World changed, and so did problem, solution
  - Runtime environment – resources or data, changed

Credit: Anonymous Creator



```
while (not edge) {
   run();
}
```

```
do {
   run();
} while (not edge);
```

Difference between “while” and “do-while”

*Credit: https://textexpander.com/blog/the-7-most-common-types-of-errors-in-programming-and-how-to-avoid-them

Error Handling

- Objective
  - Program has predictable behavior
    - Usually, terminate with a message
    - Optional: tries to recover
  - Developer gets hints to improve the code

- Example of error handling by a developer

```
check_condition

if (abnormal) {
    // print message
    // terminate
}
```

Error Handling via Exception Mechanism

- Most languages have an exception mechanism to ***anticipate*** abnormal situations and do something about those ***rare*** cases

- Typical pattern of using exceptions in programming language

```
try {
            // developer anticipates

}  catch {
           // do something about abnormal situation

          // print message
          // terminate
}
```

Exception in C++

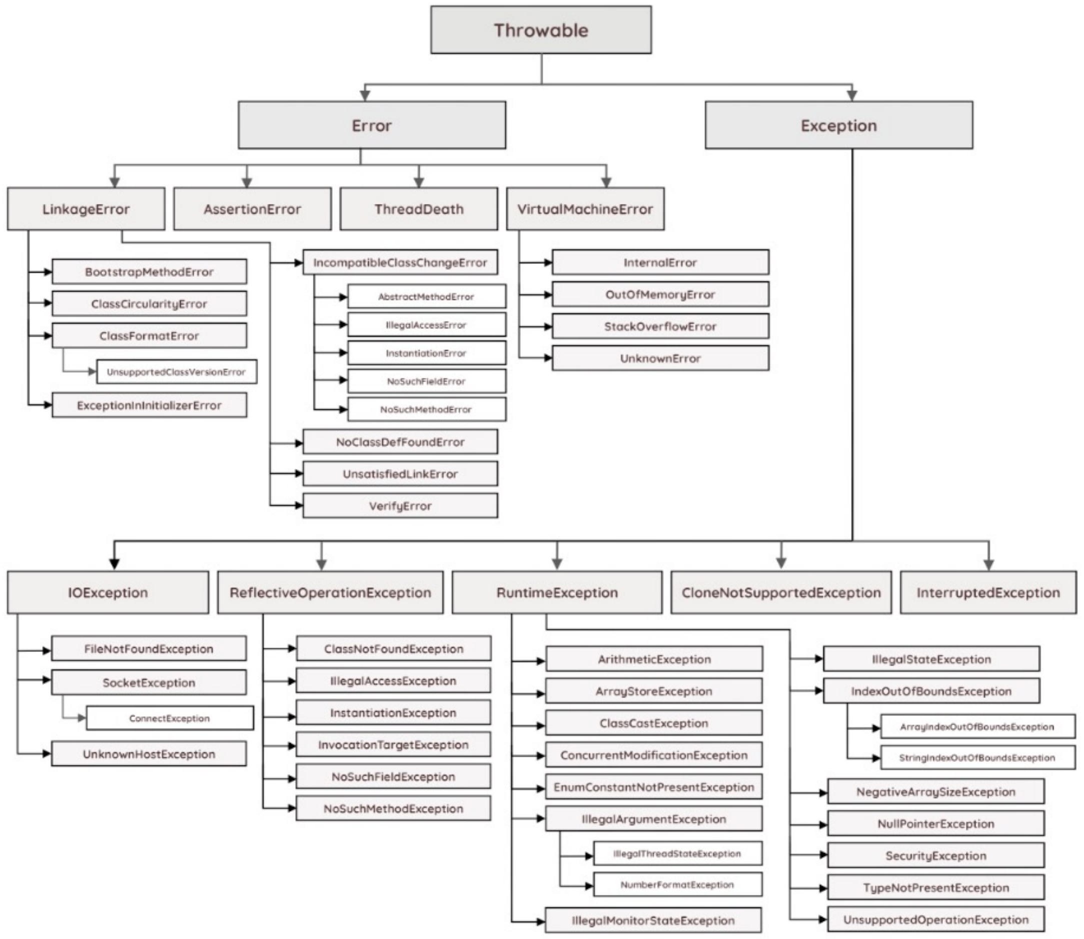
- Demonstration
  - Using exception for string out-of-range
  - Custom exception handling
- Discussion
  - Possible to have multiple handlers
  - Can throw exception too

Code Example: https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class13and14_C%2B%2B_ExceptionConsDestructor/src/Class13and14_C%2B%2B_ExceptionConsDestructor.cpp

Exception Handling in Java

- Demonstration
  - Using exception for string out-of-range

- Discussion
  - All exceptions have a super-class, Exception



Credit: https://rollbar.com/blog/java-exceptions-hierarchy-explained/#

Exception Handling in Python

- Demonstration
  - Using exception for string out-of-range

- Discussion
  - Multiple exception handlers
  - Specialized handler called if specified

Common Use-Cases for Exception Handling

- Input/ Output
  - Files, Streams not found
  - Runtime errors
- String manipulation
- Arithmetic errors – e.g., divide by zero

In-Class Exercise

Design of Exceptions

- Example setting: Calculator
  - Numbers and their operations
  - Operations: Addition, subtraction, division multiplication
  - Number types: natural numbers, whole numbers, rational numbers (fractions), irrational numbers, decimal numbers, binary, complex numbers, octal, hexadecimal, …
- Errors considerations
- Exceptions and reuse considerations

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

Discussion

- Types of errors
  - Expected
- Ways to handle
  - Reporting: Printing
  - Overcoming: Self-correcting, confirming with user, ignoring, …
- Handling programmatically - Exception handling

Reminder: Student Assessment

A	=	[900-1000]
B+	=	[850-899]
B	=	[800-849]
C+	=	[750-799]
C	=	[700-749]
D+	=	[650-699]
D	=	[600-649]
F	=	[0-599]

Tests	**1000 points**
• Course Project: programming assign.(5) and report, in-class presentation	600 points
• Class Participation and Home Work	200 points
• Quizzes and Exams	200 points
Total	1000 points

Assignments: Late Submission Policy and Extra Marks

- There is no provision for late submission for programming assignments
  - Except when prior approval has been taken from instructor due to health reasons

- One can possibly make more marks when doing final project assembly
  - **Remember**: PA1, PA2, PA3, PA4, PA5 will be the 5 programs from assignments. [100 points for each assignment]
  - **Remember**: Assembling code from one's on assignments gets the standard [100 points].
  - Extra points will be given if you make your code (for PA1 – PA5) available to others (make repository public) AND someone uses your code (any of PA1-PA5). Both will have to be reported in project report.
    - **40 points will be given per assignment to student whose assignment is reused**, and
    - **20 points will be given to person who reuses code**
  - Extra points will not exceed 100 points for any student. That is, one cannot make more than 700 points.

Concluding Section

Lecture 11: Concluding Comments

- We looked at the concept of exception
  - Errors are inevitable, handling has to be in place
  - Exception provides developer a way control behavior when rare situations occur; usually runtime
- Programming Assignment #2 is due by 10pm

About Next Lecture – Lecture 12

Lecture 12: Constructors / Destructors

- We will discuss constructors and destructors in detail
- PA #2 will 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
Feb 20 (Tu)	Review: inheritance, Polymorphism	Quiz 1 – In class
Feb 22 (Th)	In class test	Prog 3 - start
Feb 27 (Tu)	In class Project Review: PA1 and PA2	
Feb 29 (Th)	OO – operators, access control	Prog 3 - end Semester - Midpoint