CSCI 200: Foundational Programming Concepts & Design

Final Exam Review

1. Struct

Create a **`struct`** called **`Point`** that has two data members that are **`double`**s called **`x`** and **`y`**.

Additionally, write a function called **`distance`** that accepts two **`Point`** variables as input and returns a **`double`**. The function needs to return the Euclidean distance between the two points.

Write a full program that will read in four floating point values from the user separated by spaces. The first pair of values will correspond to the **`x`** and **`y`** values for **`Point a`**. The second pair of values will correspond to the **`x`** and **`y`** values for **`Point b`**.

Finally, print the distance between **`a`** and **`b`** with four decimal places.

Include all necessary headers.

Note: While it is possible to solve this problem without writing a **`struct`** or a function, on the exam to receive full credit you must write a **`struct`** and function.

2. Lists

- What are the pros/cons of an Array?
- What are the pros/cons of a LinkedList?
- What is the same about an Array & a LinkedList?
- What is different about an Array & a LinkedList?
- When should one be used over the other?
- *Consider memory, run time complexity, and other considerations*

3. Sorting

- Given a list (Array or LinkedList), implement one of Selection/Insertion/Bubble Sort to sort the list in ascending order.

4. Exceptions

- What is the structure sequence necessary when implementing exception handling?
  - What is the role of each component?

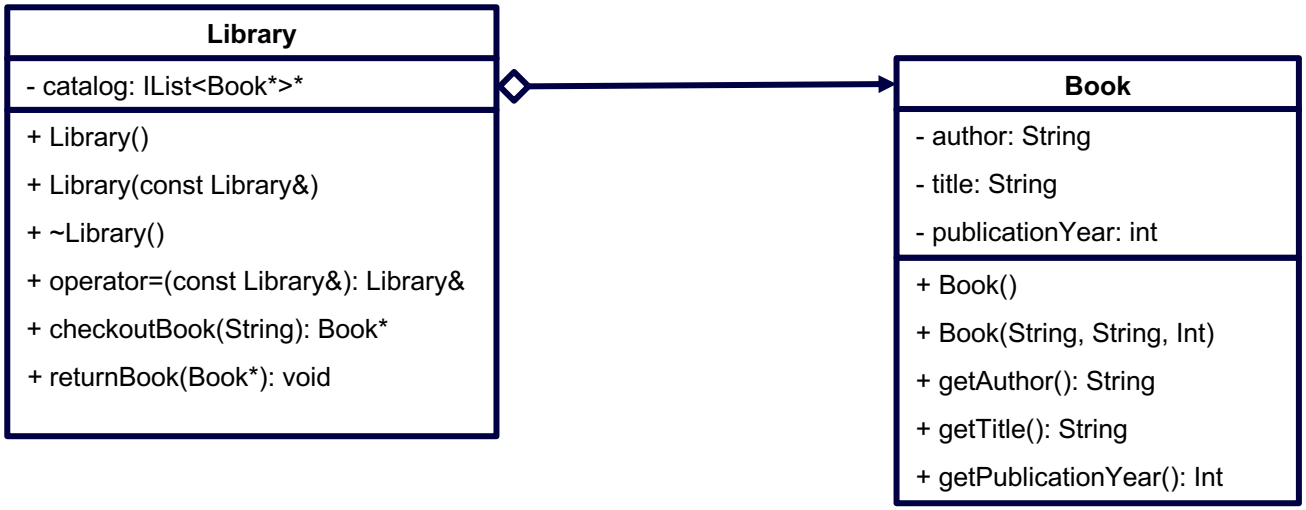
- Why should we perform exception handling in our code?

5. Fill in the missing pieces of code

```
// TODO 2: complete the following function named populate_array
/**
 * @brief populates an array by setting each element equal to its index i % MOD_VALUE
 * @param arr integer array
 * @param SIZE size of array
 * @param MOD_VALUE modulus value to apply to each element
 * @return int sum of all element values
 */

// TODO 1: create a static integer array with 100 elements

// TODO 3: call populate_array with a modulus value of 21 and store result

// TODO 4: print result of TODO 3
```

6. From UML to Class

- Write the C++ Classes corresponding to the following UML diagrams.



7. Create the Book

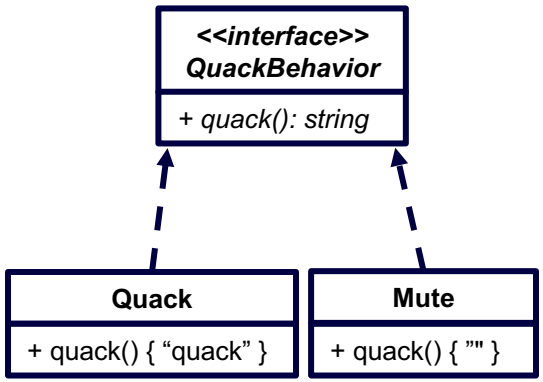
- Create the following Book functions:
  - Default constructor → Creates book with following members:
    - The C++ Programming Language
    - Bjarne Stroustrup
    - 1986
  - Parameterized constructor → sets each data member
  - Getters → returns each data member

8. Create the Library

- Create the following Library functions:
  - Default constructor → Initializes an empty list
  - Copy constructor → Performs a Deep Copy Making New Books
  - checkoutBook → If book with title is in list, removes from list and returns. If title is not in list, returns null pointer
  - returnBook → puts book in list

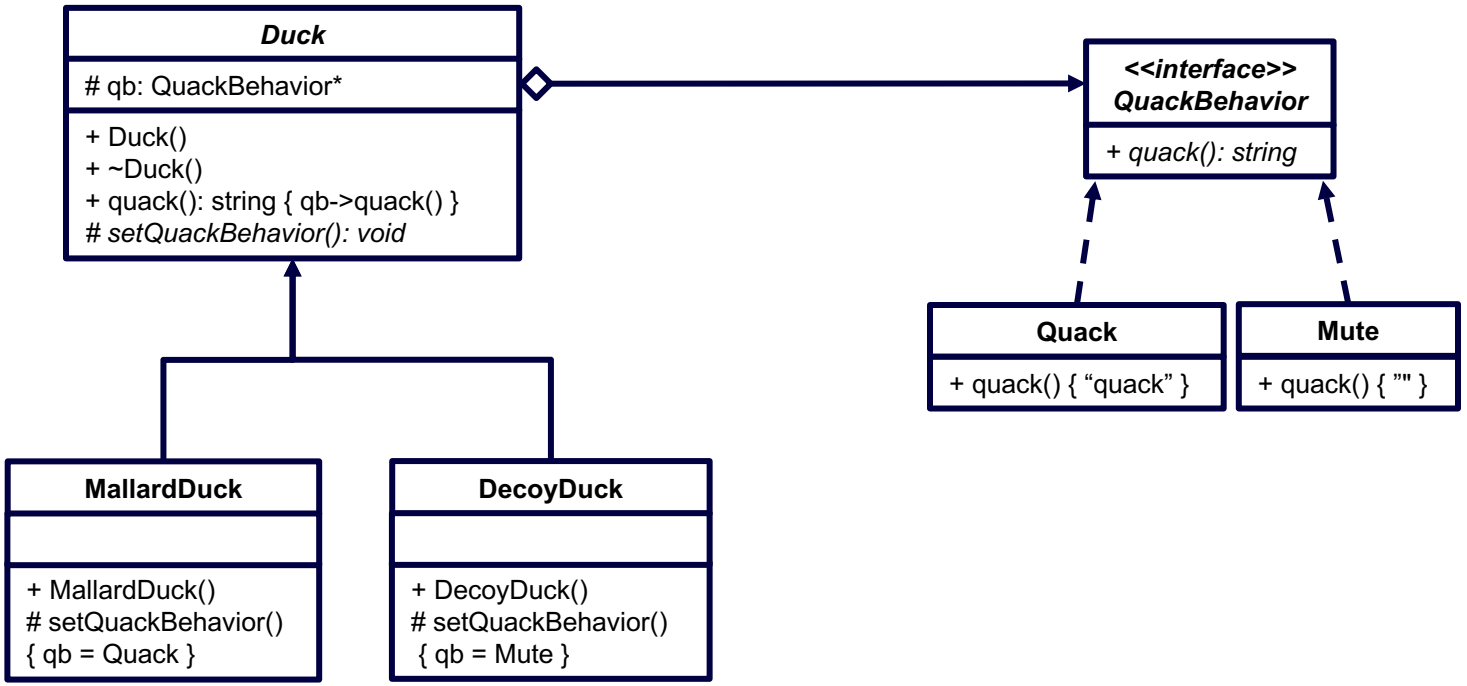
9. From UML to Class II

- Write the C++ Classes corresponding to the following UML diagrams.



10. From UML to Class III

- Write the C++ Classes corresponding to the following Duck UML diagrams. Have the Duck constructor call the setQuackBehavior method.



11. The Duck Pond

- Create a list of Ducks with 10 ducks. The odd numbered ducks are decoys, the even numbered ducks are mallards.
- Then iterate through the list and make each quack.
- Cleanup all memory.

- Use runtime polymorphism!

12. 2D Arrays

- Create a static 2D Character Array with 5 rows and 6 columns. Initialize each element to a space.

13. 2D Arrays II

- Prompt the user to enter the number of rows *r* and columns *c*.
- Create a dynamic 2D Array of Books (see #6) called bookCase that has *r* shelves and *c* books per shelf.
- Make every book B. Stroustrup's book.

14. 2D Arrays III

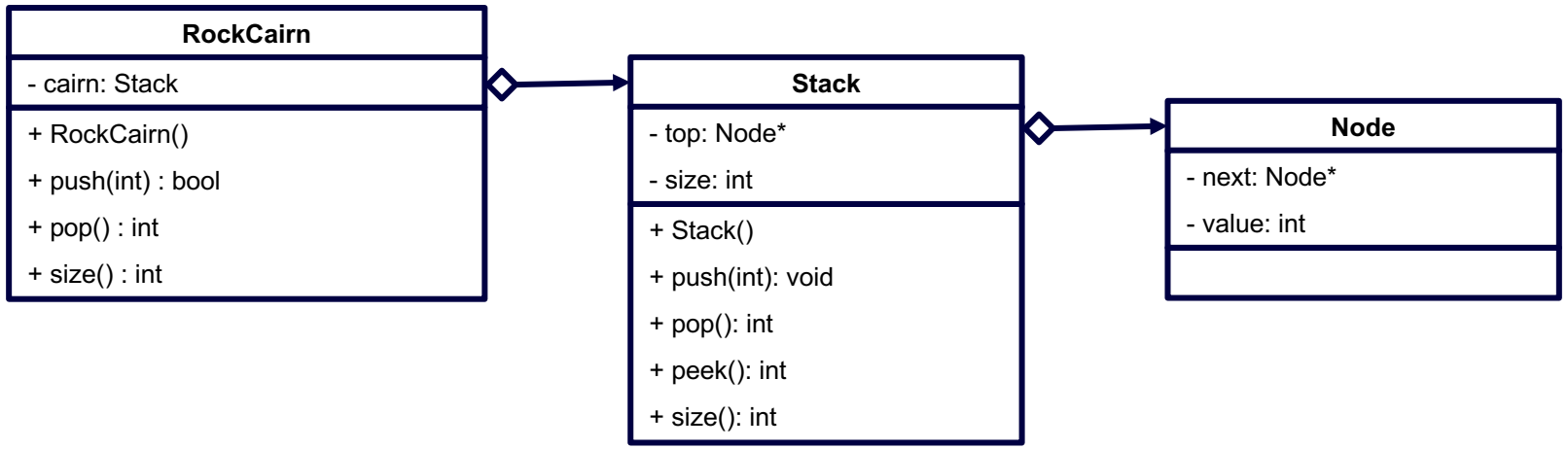
- Prompt the user to enter the number of shelves *s* and books *b*.
- Create a dynamic 2D Array of Books (see #6) called bookCase that has *s* shelves and *b* books per shelf.
- When creating each book, prompt the user for the book information.
- Then sort the books on each shelf alphabetically.

15. 2D Arrays IV

- After the books are sorted, ask the user for a book name.
  - If it exists, tell the user which shelf the book is on and what position on the shelf it is in.
  - Otherwise, tell the user the book cannot be found.
- What is a good search strategy?
- What is the run time to search the bookcase?
  - Express it in terms of *s* and *b*

16. Stack Class

- Create a class called RockCairn that contains a stack of integers. RockCairn::push() will only push to its stack if the value to be pushed is smaller than the current top value of the stack. It returns true if the value was pushed, false if not. You can assume Stack & Node are implemented.



17. SOLID

- What are the SOLID Principles?
- What is a scenario that each would be applied to?