UNIVERSITY AT ALBANY
State University of New York

COLLEGE OF ENGINEERING AND APPLIED SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

ICSI311 Principles of Programming Languages

Assignment 02 Created by Qi Wang

Click here for the recording

Table of Contents

Part I: General Information

- All assignments are individual assignments unless it is notified otherwise.
- All assignments must be submitted via Blackboard. No late submissions or e-mail submissions or hard copies will be accepted.
- Two submission attempts will be allowed on Blackboard. **Only the last attempt will be graded.**
- Work will be rejected with no credit if
    - The work is late.
    - The work is not submitted properly (Blurry, wrong files, crashed files, etc.).
    - The work is a copy or partial copy of others' work (such as work from another person or the Internet).
- Students must turn in their original work. Any cheating violation will be reported to the college. Students can help others by sharing ideas and should not allow others to copy their work.
- Documents to be submitted:
    - **UML class diagram(s)** – created with Violet UML or StarUML
    - **C++ source file(s) with standard comments**
    - **Supporting files if any** (For example, files containing all testing data.)

  Note: Only the above-mentioned files are needed. Copy them into a folder, zip the folder, and submit the **zipped** file. We don't need other files from the project. **Submissions that don't meet this requirement may be rejected with no credit.**
- Students are required to submit a design, all the error-free source files with comments, and supporting files. Lack of any of the required items or programs with errors will result in a really low credit or no credit.
- **Grades and feedback**: TAs will grade. Feedback and grades for properly submitted work will be posted on Blackboard. Students have limited time/days from when a grade is posted to dispute the grade. Check email daily for the grade review notifications sent from the TAs. If students have any questions regarding the feedback or the grade, they should reach out to their TAs first.

Part II: Grading Rubric

Performance Indicators		Max points	Earned points
#1: UML design	• Efficient and meet all requirements • Visibility, name, and type/parameter type/return type present for all classes with no issues • Class relationships are correct	5	
#2: Node	• Comments and naming conventions • A generic non-inner class and all methods present with no issues • Meet all requirements	5	
#3: Binary Search Tree	• Comments and naming conventions • Specifications and implementation are separate • A generic class and all methods present without issues • Meet all requirements	15	
#4: ADT – people database	• Comments and naming conventions • Specifications and implementation are separate • All methods present without issues • Meet all requirements	15	
#5: Test	• Comments and naming conventions • Enough testing data from a file is used • Decompose method *main* • ADT-people database is completely tested • Meet all requirements	10	
Total Points Awarded			/ 50
Feedback to the student			

Part III: Examples

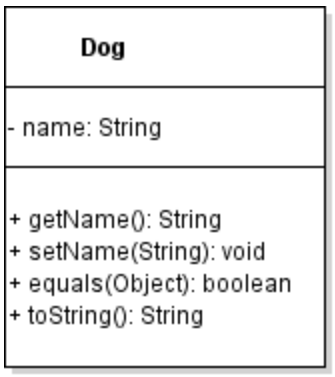
To complete a project, the following steps of a software development cycle should be followed. These steps are not pure linear but overlapped.

Analysis-design-code-test/debug-documentation.

1) Read project description to understand all specifications(**Analysis**)
2) Create a design (an algorithm or a UML class diagram) (**Design**)
3) Create programs that are translations of the design (**Code/Implementation**)
4) Test and debug(**test/debug)**
5) Complete all required documentation. (**Documentation**)

The following shows a sample design. By convention.

- *Constructors* and *constants* should not be included in a class diagram.
- For each *field* (instance variable), include *visibility*, *name*, and *type* in the design.
- For each *method*, include *visibility*, *name*, *parameter type(s)* and *return type* in the design.
    - DON'T include *parameter names*, only *parameter types* are needed.
- Show class relationships such as *dependency*, inheritance, *aggregation*, etc. in the design. Don't include the *driver* program since it is for testing purpose only.



Part IV: Description

Assignment 2 An abstract data type – a people database

Goals:

- Review and develop a deep and comprehensive understanding of the object-oriented paradigm
- Review and develop a deep and comprehensive understanding of abstract data types
- Review data structures and searching algorithms such as binary search trees, binary search, etc.

Design an abstract data type that maintains a database containing data, such as name and birthday, about your friends and relatives. You should be able to enter, remove, modify, search this data, list everyone who satisfies a given criterion (for example, people born in a given month), or list everyone in the database. You can assume that the names are unique. Use a binary search tree as the data structure when implementing the ADT people database. **It is required to use C++ for this assignment.**

You must design and implement a generic node (as a non-inner class) and a generic binary search tree (BST) and use the BST as the data structure of the ADT people database. Assignments using BST or Node from C++ library or any other libraries will be rejected. The following are the main components for this assignment:

- A generic node (Can't be an inner class.)
- A generic BST (The specifications and implementation must be separate.)
- ADT people database ((The specifications and implementation must be separate.)
- Test the ADT
    - You must save your own testing data in a text file. The file must be in the project default folder and must be submitted as part of the assignment.
    - Read the data from the file for testing.
    - Method *main* must be decomposed.

☺ **Be Happy and Efficient!**