Module 7: Class design

Learning Objectives

1. Design a class given a specification.
2. Implement a C++ class with accessors, mutators, member variables, and member functions.

Process Skills

1. Critical thinking. Identify key aspects of a scenario to represent them as a class.
2. Teamwork. Collaborate in developing a class design.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. If there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: _____ Date: _____

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	
Presenter. Talks to the facilitator and other teams.	
Reflector. Considers how the team could work and learn more effectively.	
Recorder. Records all answers and questions and makes the necessary submission.	

For virtual activities: Once you select your roles, change your Zoom name using the format and example below.

Format: *Group X: First name, Last name initial / Role*

Example: *Group 1: Paul I / Presenter*

Model 1. Class design (12 min) Start time: _____

Task 1: Create a class to represent a donation tracker. When creating a donation tracker, it starts out with no donations added. Users can add donations and get the average donations.

Key Information

Name: Donation Tracker
Behaviors:

1. Name: construct
    a. Input: No input
    b. Output: No output
    c. Algorithm: No implementation
2. Name: donate
    a. Input: donation (double)
    b. Output: None (void)
    c. Algorithm:
        i. Add donation value to the container
3. Name: average donations
    a. Input: None
    b. Output: average donation (double)
    c. Algorithm:
        i. Loop through each donation and add them together.
        ii. Divide the sum of all donations and divide by the number of elements in the container.

State: container to track donations (std::vector<double>)

1. What key information provides a short and clear description of the class?

Name

2. According to the task description, how many behaviors does the class perform?

3

3. How many ways can we construct a Donation tracker?

1

4. What is the relationship between behaviors and the state? Take note that performing a behavior produces some information, such as the donation given. However, this is forgotten after completing the action. For example, after making a donation, you may forget how much you gave or how much was the current donation average. Place a check (✓) beside your answer.

   a. States provide inputs required to perform behaviors
   b. States store information produced while performing behaviors ✓
   c. There is no relationship between behaviors and state

5. Which previous discussion involved identifying the name, input, output, and algorithm for a specific goal? Place a check (✓) beside your answer.

   a. Vectors and maps
   b. Range-based loops
   c. Instantiating objects
   d. Function design ✓

Class design process

1. Read and understand the task/problem.
2. Identify a short name that clearly describes the class and its behaviors.
3. Analyze the task to identify how the class is constructed. There may be more than one way to construct the class.
4. Identify the inputs and algorithm for each construction behavior. Construction behaviors don't give back values.
5. Find other behaviors provided by the class according to the task. Use the Function Design Process to identify the name, input, output, and algorithm of each behavior.
6. For each behavior, identify information that needs to be kept after it is performed. Choose a name to describe the information and list them under state.
7. Identify the appropriate data type to represent the information listed under states.

6. Work as a group to apply the class design process. Fill in the key information for Task 2 described below.

Task 2: Create a class to represent a volunteer. For now, we do not perform any actions when constructing a volunteer. We can retrieve and change the volunteer's name.

Key Information

Class Name: Volunteer

Behaviors:

1. Name: construct
    a. Input: No input
    b. Output: No output
    c. Algorithm: No implementation
2. Name: retrieve name
    a. Input: No input
    b. Output: name (std::string)
    c. Algorithm: Return the name of the volunteer
3. Name: change name
    a. Input: new name (std::string)
    b. Output: None (void)
    c. Algorithm: Replace current name with new name

State: name (std::string)

🛑 **STOP HERE AND WAIT FOR FURTHER INSTRUCTIONS**

Model 2. Class diagram creation (6 min) Start time: _____

DonationTracker
std::vector<double> donations_;

DonationTracker();
void Donate(double);
double AverageDonation();

7. Recall Task 1's key information. What key information did we use to identify the name of the class in the class diagram? Place a check (✓) beside your answer.

    a. Name ✓
    b. State
    c. Behaviors

8. Recall Task 1's key information. What key information did we use to identify the member variables in the class diagram? Place a check (✓) beside your answer.

    a. Name
    b. State ✓
    c. Behaviors

9. Recall Task 1's key information. What key information did we use to identify the member functions of the class in the class diagram? Place a check (✓) beside your answer.

    a. Name
    b. State
    c. Behaviors ✓

10. Which previous discussion involved identifying the return type, name, parameters, and algorithm to create function declarations? Place a check (✓) beside your answer.

    a. Function call process
    b. Function definition process ✓
    c. Function design process

Class diagram creation process

1. Use the name from the key information as your class' name.
2. For each behavior, apply the Function Definition Process to create the member function declarations. Take note that for constructors, you do not include a return type.
3. Create a member variable for each information you listed under state*.

*According to Google's C++ Style Guide, we add an underscore at the end of member variable names to distinguish them from member function parameters.

11. Apply the class diagram creation process to create a class diagram using the key information you identified for Task 2.

Volunteer
std::string name_;

Volunteer();
std::string RetrieveName();
void ChangeName(std::string name);

🛑 **STOP HERE AND WAIT FOR FURTHER INSTRUCTIONS**

Model 3. Classes in C++ (15 min) Start time: _____

```
// donation.h
#include <vector>

class DonationTracker {
 public:
   DonationTracker() { }
   void Donate(double donation) { donations_.push_back(donation); }
   double AverageDonation();

 private:
   std::vector<double> donations_;
};
```

```
// donation.cc
#include "donation.h"

double DonationTracker::AverageDonation() {
   double sum = 0;
   for (double donation: donations_) {
       sum += donation;
   }
   return sum / donations_.size();
}
```

```
// main.cc
#include "donation.h"
#include <iostream>

int main() {
   DonationTracker titan_give;
   titan_give.Donate(100.0);
   titan_give.Donate(224.25);
   std::cout << "Titan Give Average Donations: "
             << titan_give.AverageDonation() << "\n";
   DonationTracker tuffy_basic_needs;
   tuffy_basic_needs.Donate(5.75);
   std::cout << "Tuffy Basic Needs Average Donations: "
             << tuffy_basic_needs.AverageDonation() << "\n";
   return 0;
}
```

Screen output

Titan Give Donations: 2
Titan Give Average Donations: 162.125
Tuffy Basic Needs Average Donations: 5.75

12. Analyze donation.h. What keyword do we use to create a class?

class

13. There are two sections in the class definition, *private* and *public*. In which section do we place our member variables?

private

14. In which section do we place our member functions? Elements under the *public* section can be accessed by using the dot notation on an object. Elements under the *private* section can only be accessed from inside the class.

public

15. Google's C++ style guide suggests we write *inline member functions* (function definition) inside the class if they are less than 10 lines long, do not contain loops or switch statements, are not virtual, and are not recursive. Otherwise, we only write the member function declaration. The class and its member functions are often written in the header file (.h). Which of DonationTracker's member functions is written inline? List its name below.

Donate

16. Member functions that are not inline are often written outside the class and inside the implementation file (.cc). What syntax do we follow to tell the compiler that a particular member function is part of a class? See the AverageDonation member function for reference. Place a check (✓) beside your answer.

    a. **<return type> <class name>::<member function name> { <function body> } ✓**
    b. <return type> <member function name> { <function body> }
    c. <return type> <member function name>::<class name> { <function body> }
    d. <return type> <class name> { <function body> }

17. Calling an object's member functions gives the function access to the object's member variables. For example, calling Donate modifies an object's donations_ vector. When we called tuffy_basic_needs.Donate(5.75), which donations_ vector was modified? Place a check (✓) beside your answer.

    a. titan_give's donations_ member variable
    b. tuffy_basic_needs' donations_ member variable ✓
    c. DonationTracker's donations_member variable

18. *Accessors* are special member functions whose only purpose is to retrieve the value of a member variable. You should have already created an accessor for your class. Write the member function's name below.

RetrieveName

19. *Mutators* are special member functions whose only purpose is to change the value of a member variable. You should have already created a mutator for your class. Write the member function's name below.

ChangeName

20. Google's style guide suggests we name accessors with the name of the member variable they are accessing in the upper camel case format without the trailing underscore. For example, the accessor for an int member variable called age_ might be int Age(); Rewrite your accessor in the box below using the suggested naming convention.

std::string Name();

21. Google's style guide suggests we name mutators with the word Set followed by the name of the member variable they are accessing in the upper camel case format without the trailing underscore. For example, the mutator for an int member variable called age_ might be void SetAge(int age); Rewrite your mutator in the box below using the suggested naming convention.

void SetName(std::string name);

Basic class definition process (based on a class diagram)

1. In the header file (.h), write the class keyword followed by the name of your class. Place curly braces ({ }) that will enclose your code and do not forget to place a semicolon (;) at the end.
2. Write the public section and copy over your member function declarations.
3. Write the private section and copy over your member variables.
4. Use the Function Definition Process to translate your functions into C++.
5. Write the full member function declaration inside the class if a member function has less than 10 lines, does not contain loops or switch statements, is not virtual, and is not recursive.
6. For other member functions, write only their declarations inside the class. Write the complete member function definition in the implementation file (.cc). Add the class name followed by:: to the name of the member function. Specifically, use the format <return type> <class name>::<member function name> { <function body> }.

22. Use the Basic Class Definition Process to write the C++ code for your volunteer class. Use your class diagram in Model 2 for reference. Don't forget to use the suggested naming conventions for your accessors and mutators.

```
// volunteer.h
class Volunteer {
 public:
  Volunteer() { }
  std::string Name() { return name_; }
  void SetName(std::string name) { name_ = name; }

 private:
  std::string name_;
};
```

🛑 **STOP HERE AND WAIT FOR FURTHER INSTRUCTIONS**

Reflector questions

1. What was the most useful thing your team learned during this session?

2. Describe any challenges your group faced while extracting key information from the given task.

3. Describe the benefits of working together as a group in designing your Volunteer class.