Module 3: Classes and Objects

Learning Objectives

1. Write code that creates and uses objects.
2. Explain the relationship between member variables and member functions.
3. Interpret constructor and member function declarations.

Process Skills

1. Information processing. Extract structural patterns from sample code.
2. Critical thinking. Apply previous knowledge to understand classes and objects.

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. Classes (5 min) Start time: _____

Class diagram

Refrigerator	← Class name
std::string food_type; int capacity; int current_count;	← member variables

Refrigerator();
Refrigerator(std::string food_type, int capacity);
void SetFoodType(std::string food_type);
bool Add(int quantity);
bool Get(int quantity);
	← member functions

*For simplicity, our Refrigerator class assumes you can only store one type of food.

Member variable descriptions

food_type - the type of food in the refrigerator (e.g., apple, milk, banana)
capacity - the maximum amount of food that can fit the refrigerator
current_count - the current amount of food in the refrigerator

Member function descriptions

void SetFoodType(std::string food_type); - change the type of food stored in the refrigerator

bool Add(int quantity); - add food to the refrigerator; quantity is added to current_count if it does not exceed capacity; returns true if current_count was updated, false otherwise.

bool Get(int quantity); - gets food from the refrigerator; quantity is only subtracted from current_count if it is less than or equal to current_count; returns true if current_count was updated, false otherwise

Constructors

Refrigerator(); - instantiate a Refrigerator object that contains an "unspecified" food type, capacity of 30 and current_count of 0.

Refrigerator(std::string food_type, int capacity); - instantiate a Refrigerator object using the given food type and capacity with a current_count of 0.

1. How many member variables does the Refrigerator class contain?

3

2. Which member function can we use to add items to the refrigerator? Write the function's name below.

Add

3. According to the member function descriptions, when will the Get member function return false?

Get will return false when current_count is not updated (if quantity is greater than current_count).

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

Model 2. Objects (12 min) Start time: _____

Actions	**Object diagram after performing the action**
1. **Instantiate** a Refrigerator named apple_fridge	**apple_fridge: Refrigerator** food_type = "unspecified" capacity = 30 current_count = 0
2. **Set** apple_fridge's food type to "apple"	**apple_fridge: Refrigerator** food_type = "apple" capacity = 30 current_count = 0
3. **Instantiate** a Refrigerator named milk_fridge for "milk" with a capacity of 18.	**apple_fridge: Refrigerator** food_type = "apple" capacity = 30 current_count = 0 **milk_fridge: Refrigerator** food_type = "milk" capacity = 18 current_count = 0

4. **Add** 25 items to apple_fridge	**apple_fridge: Refrigerator** food_type = "apple" capacity = 30 current_count = 25 **milk_fridge: Refrigerator** food_type = "milk" capacity = 18 current_count = 0
5. **Add** 15 items to milk_fridge	**apple_fridge: Refrigerator** food_type = "apple" capacity = 30 current_count = 25 **milk_fridge: Refrigerator** food_type = "milk" capacity = 18 current_count = 15
6. **Get** 8 items from milk_fridge	**apple_fridge: Refrigerator** food_type = "apple" capacity = 30 current_count = 25 **milk_fridge: Refrigerator** food_type = "milk" capacity = 18 current_count = 7
7. **Get** 10 items from milk_fridge	**apple_fridge: Refrigerator** food_type = "apple" capacity = 30 current_count = 25 **milk_fridge: Refrigerator** food_type = "milk" capacity = 18 current_count = 7

4. The term *instantiate* refers to the process of creating an object from a class. The first action instantiates a Refrigerator object. What name refers to this object?

apple_fridge

5. The data type of an object is the class that was used to instantiate it. In our example, what is the data type of apple_fridge?

Refrigerator

6. *Constructors* are special member functions used to instantiate objects. Which constructor do you think instantiates apple_fridge in the first action? Hint: Review the constructor descriptions in Model 1. Place a check (✓) beside your answer.

    a. Refrigerator(); ✓
    b. Refrigerator(std::string food_type, int capacity);

7. Why did we set the food type in the second action? Place a check (✓) beside your answer.

    a. We want to change the food type to "apple" instead of the default name, "unspecified." ✓
    b. We actually don't need to change the food type because the name accurately represents the refrigerator's contents.
    c. We want to change the capacity of the refrigerator.

8. The third action instantiates another Refrigerator. Which constructor do you think was used to instantiate milk_fridge? Hint: Review the constructor descriptions in Model 1. Place a check (✓) beside your answer.

    a. Refrigerator();
    b. Refrigerator(std::string food_type, int capacity); ✓

9. Why didn't we need to set the food type of the milk_fridge object? Place a check (✓) beside your answer.

    a. We actually need to change the capacity because the constructor does not assign it to our expected value.
    b. The constructor has already changed the name to our intended values. ✓
    c. We actually need to change the name because the constructor does not assign it to our expected value.

10. How many refrigerator objects have been instantiated after the third action?

2

11. The fourth action adds 25 items. Why does the apple_fridge object's current_count change to 25, but not milk_fridge's current_count? Place a check (✓) beside your answer.

    a. Adding 25 items to milk_fridge will exceed the capacity, so it does not update it.
    b. milk_fridge's current_count should also change to 25.
    c. We added items to the apple_fridge object, so it does not affect milk_fridge's member variables.✓

12. The sixth action gets eight items from milk_fridge, which reduces its current_count to 7. Why doesn't milk_fridge's current_count change when we get 10 items in the seventh action? Place a check (✓) beside your answer.

    a. milk_fridge does not contain enough items, so the current_count is not updated according to the Get member function. ✓
    b. current_count should change because the items taken are less than milk_fridge's capacity according to the Get member function.
    c. current_count is not updated because the items taken exceed milk_fridge's capacity according to the Get member function.

13. What is the relationship between member functions and member variables? Place a check (✓) beside your answer.

    a. Member functions of an object cannot access or change its member variables.
    b. Member functions of an object can access or change its member variables. ✓
    c. Member functions change the member variable values of all objects instantiated from the same class.

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

Model 3. Constructors (6 min) Start time: _____

Action	Constructor	C++ code equivalent
1. Instantiate a Refrigerator named apple_fridge	Refrigerator();	Refrigerator apple_fridge;
3. Instantiate a Refrigerator named milk_fridge for milk with a capacity of 18.	Refrigerator(std::string food_type, int capacity);	Refrigerator milk_fridge("milk", 18);

14. The constructor used in the first action is called a *default constructor*, which is a constructor that does not take any parameters. Which syntax describes the correct way of instantiating an object using a default constructor? Place a check (✓) beside your answer.

    Take note that < > symbols indicate placeholders. They will be replaced by elements that match the description. For example, <Class name> may be replaced by Refrigerator because it is a class name.

    a. <Class name> <Identifier>; ✓
    b. <Class name> <Identifier>();
    c. <Class name> <Identifier>(<parameters>);

15. Which syntax describes the correct way of instantiating an object using the constructor shown in the third action? Place a check (✓) beside your answer.

    a. <Class name> <Identifier>;
    b. <Class name> <Identifier>();
    c. <Class name> <Identifier>(<arguments>); ✓

16. Discuss as a group the Food constructor declaration below. Write code to instantiate a Food object into a variable called my_apple, with a name "Apple" (name), that spoils in 7 days (spoil_age), and is currently 2 days old (age).

    Food(std::string name, int spoil_age, int age);

```
Food my_apple("Apple", 7, 2);
```

Model 4. Member Functions (15 min) Start time: _____

Action	Member function	C++ code equivalent
2. Change apple_fridge's food type to "apple"	void SetFoodType(std::string food_type);	apple_fridge.SetFoodType("apple");
4. **Add** 25 items to apple_fridge	bool Add(int quantity);	bool added = apple_fridge.Add(25);

17. Which symbol do we use to call an object's member function? Place a check (✓) beside your answer.

    a. . ✓
    b. ;
    c. :

18. Why do we need to specify an object when calling a member function? For example, apple_fridge.Add(25) instead of Add(25); Select all that apply. Place a check (✓) beside your answer.

    a. It identifies the intended object when there are multiple objects of the same class. ✓
    b. The object's data type tells the system which class' member function should be used. ✓
    c. It promotes readability and helps developers trace code. ✓

19. Why do we assign the result of the Add member function call to a variable in the fourth action, but not with SetFoodType in the second action? Place a check (✓) beside your answer.

    a. The Add member function returns a boolean value. ✓
    b. The SetFoodType member function returns a value.
    c. The Add member function does not return a value.

20. Discuss as a group Food's SetSpoilAge member function below. Write code that will (1) instantiate a Food object using its default constructor, and (2) call the object's SetSpoilAge member function to indicate that it will spoil in 5 days instead.

    void SetSpoilAge(int spoil_age)

```
Food my_food;
my_food.SetSpoilAge(5);
```

Instantiating objects

1. Select the constructor you want to use for instantiating the object.
2. When using the default constructor use the syntax <Class name> <identifier>;
3. When using a non-default constructor use the syntax <Class name> <identifier>(<arguments>);

Using objects

1. Identify the name of the object and its data type (Class used to instantiate it).
2. Identify the member function that performs your intended action.
3. Call the member function by using the syntax <object name>.<member function name>(<arguments);
4. If the member function returns a value, assign it to a variable with the same data type as the member function's return type.

21. Apply the Instantiating and Using objects processes to perform the actions described below.

    The DonationTracker class is designed to track the number and total amount of donations received.

DonationTracker
int donation_count; double total_donations; //total donations received

DonationTracker();
DonationTracker(int donations_count, double total_donations);
void Donate(double amount);
int DonationCount();
double TotalDonations();
double AverageDonation();

    Write code to instantiate a DonationTracker object using its default constructor. Use the object to donate three amounts: $100.00, $224.25, and $5.75. Get the object's donation

count and store it in a variable called num_donations. Get the average donations from the object and store it in a variable called avg_donations.

```
int main() {
  DonationTracker tracker;
  tracker.Donate(100.00);
  tracker.Donate(224.25);
  tracker.Donate(5.75);
  int num_donations = tracker.DonationCount();
  double avg_donations = tracker.AverageDonation();

  std::cout << "Number of donations: " << num_donations << "\n";
  std::cout << "Average donation: $" << avg_donations << "\n";
  return 0;
}
```

Reflector questions

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

2. Did you apply a different strategy for finding patterns from the sample code compared to what you did in the previous module?

3. Recall our previous discussions and the concepts you learned from your prior programming course (CPSC 120 or equivalent). Write the names of the concepts that helped you complete today's activity.