Module 10: Composition

Learning Objectives

1. Design and implement classes that use composition.
2. Write code that appropriately uses composed objects.
3. Write code that maintains class invariants.

Process Skills

1. Critical thinking. Use prior knowledge to interpret and explain new concepts.
2. Problem solving. Interpret a programming problem and use prior knowledge to solve it.

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. Object Composition (10 min) Start time: ______

```
// date.h
#include <iostream>
class Date {
 public:
  Date(int month, int day, int year) : month_(month), day_(day),
                                       year_(year) {}
  Date(const Date& other): month_(other.Month()),
day_(other.Day()),
                           year_(other.Year()) { }
  Date() : Date(1, 1, 1970) {}
  int Month() const { return month_; }
  void SetMonth(int month) { month_ = month; }
  int Day() const { return day_; }
  void SetDay(int day) { day_ = day; }
  int Year() const { return year_; }
  void SetYear(int year) { year_ = year; }

  std::string ToString() const {
    return std::to_string(month_) + "/" + std::to_string(day_) +
           "/" + std::to_string(year_);
  }

private:
 int month_;
 int day_;
 int year_;
};
```

```cpp
// fridge.h
#include "date.h"
#include <iostream>
#include <map>

class Refrigerator {
 public:
  Refrigerator(const Date &last_checkup_date, int max_contents,
               const std::map<std::string, int> &contents)
      : last_checkup_date_(last_checkup_date),
        max_contents_(max_contents),
        contents_(contents) { }
  Refrigerator() : max_contents_(10) {}

  int MaxContents() const { return max_contents_; }
  const std::map<std::string, int>& Contents() const {
    return contents_;
  }

  bool Fill(const std::string &name, int quantity) {
    if (contents_.size() + quantity > max_contents_) {
      return false;
    } else {
      if (contents_.count(name) == 0) {
        contents_.insert({name, quantity});
      } else {
        contents_.at(name) += quantity;
      }
      return true;
    }
  }

  void DisplayInfo() const {
    std::cout << "Max contents: " << max_contents_ << "\n"
              << "Current contents: " << contents_.size() << "\n";
  }

private:
 Date last_checkup_date_;
 int max_contents_;
 std::map<std::string, int> contents_;
};
```

1. Analyze the Refrigerator class and identify the data types of its member variables.

Member variable	**Data type**
last_checkup_date_	Date
max_contents_	int
contents_	std::map<std::string, int>

2. Which among the member variables are objects? List them below. We use the term *object composition* when a class contains a member variable that is an object of the same or another class.

last_checkup_date_ and contents_

3. Analyze the Refrigerator class' DisplayInfo member function. Why do we use the dot notation to call contents_'s size member function? Place a check (✓) beside your answer.

   a. contents_ is an object, so we use the dot notation to access its member function. ✓
   b. Refrigerator is a class, so we use the dot notation to access its member function.
   c. We don't need the dot notation to call the size member function.

4. Analyze the Date class. Which member function gives us back a std::string that summarizes the data stored by the Date object? Write the function name below.

ToString

5. Modify the Refrigerator class' DisplayInfo member function, so it also displays the last checkup date. For example, "Last check-up: 10/03/2022".

```
void DisplayInfo() const {
  std::cout << "Max contents: " << max_contents_ << "\n"
            << "Current contents: " << contents_.size() << "\n";
  std::cout << "Last check-up: ";
  // TODO: Display last checkup date.
  std::cout << last_checkup_date_.ToString() << "\n";





}
```

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

Model 2. Composed Object Construction (4 min)

Start time: ______

Review the Refrigerator class' constructors.

```
Refrigerator(const Date &last_checkup_date, int max_contents,
             const std::map<std::string, int> &contents)
    : last_checkup_date_(last_checkup_date),
      max_contents_(max_contents),
      contents_(contents) { }
Refrigerator() : max_contents_(10) {}
```

6. Analyze the Refrigerator class' non-default constructor. According to its member initializer, what value was assigned to max_contents_? Place a check (✓) beside your answer.

   a. The max_contents parameter ✓
   b. The last_checkup_date parameter
   c. The contents parameter.

7. What value was passed as a parameter to the contents_ initializer? Place a check (✓) beside your answer.

   a. The max_contents parameter
   b. The last_checkup_data parameter
   c. The contents parameter. ✓

8. When the member initializer list contains a member variable that is an object, we invoke its class' constructor. Which std::map constructor do you think is called by the contents_ initializer? Place a check (✓) beside your answer.

   a. map();
   b. map(const std::map& other); ✓
   c. map(std::initializer_list<value_type> init);

9. Analyze the Refrigerator class' last_checkup_date_ member variable. Take note that it was not initialized in the default constructor. In this case, the default constructor of the last_checkup_date_'s class is called when a Refrigerator object is instantiated.

   See the statement below, which invokes the default constructor to instantiate a Refrigerator object called milk_fridge. What do you think are the month, day, and year values of milk_fridge's last_checkup_date_? Write their values in the table below.

   ```
   Refrigerator milk_fridge;
   ```

milk_fridge's last_checkup_date_ member variable	**value**
month	1
day	1
year	1970

10. Modify Refrigerator's default constructor to include a member initializer for last_checkup_date_. Assign its month to 10, day to 7, and year to 2022. *Hint: Review the Date class' constructors.*

```
// TODO: Add a member initializer to set last_checkup_date_ to
//       10/7/2022
Refrigerator()
    : max_contents_(10), last_checkup_date_(10, 7, 2022)


{ }
```

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

Model 3. Class Invariant (12 min) Start time: ______

Review the member functions that interact with the Refrigerator's contents_ member variable.

```
const std::map<std::string, int>& Contents() const {
  return contents_;
}

bool Fill(const std::string &name, int quantity) {
  if (contents_.size() + quantity > max_contents_) {
    return false;
  } else {
    if (contents_.count(name) == 0) {
      contents_.insert({name, quantity});
    } else {
      contents_.at(name) += quantity;
    }
    return true;
  }
}
```

11. What is the return type of the Contents accessor?

const std::map<std::string, int>&

12. What do you think it means when the return type has a reference declarator (&)? Place a check (✓) beside your answer.

   a. We are returning a reference to the contents_ member variable. ✓
   b. We are returning a copy of the contents_ member variable.
   c. We are returning a reference to a member function's parameter.

13. What is the advantage of returning a reference to an object rather than a copy? Select all that apply. Place a check (✓) beside your answer(s).

   a. We reduce coding because it takes fewer characters to express returning a reference.
   b. We save memory because we don't create a new copy of the object. ✓
   c. We save time because we don't need to copy the values contained in the object. ✓

14. What does it mean when we add the const keyword to the return type? Place a check (✓) beside your answer.

    a. Whoever accepts the returned object can access its values and change them.
    b. Whoever accepts the returned object can access its values, but cannot change them. ✓
    c. You are not allowed to return a const return type.

15. It is good practice to provide member functions that manipulate composed objects (e.g., Fill function) rather than returning non-const references to ensure *class invariants*. *Class invariants* refer to rules the class should follow. If we remove the const keyword what are potential actions that can violate the Refrigerator class' invariants? Place a check (✓) beside your answer.

    a. Developers cannot display the contents of the refrigerator.
    b. Developers can display the contents of the refrigerator.
    c. Developers can add more food than the refrigerator's max_contents_. ✓

16. Create an accessor for the last_checkup_date_ that allows developers to view the object, but not manipulate it.

```
const Date& LastCheckupDate() const { return last_checkup_date_; }
```

17. Complete the code for Refrigerator's ConductCheckup member function, so it only updates the last_checkup_date_ if the provided date is later than its previous value. Two variables are provided for you to compare the last_checkup_date_ and the new checkup_date. The member function should return true if the last_checkup_date_ was updated and false otherwise. *Note: Feel free to disregard the two variables if you want to use your own strategy for comparing two dates.*

```
bool ConductCheckup(const Date &checkup_date) {
  int last_date_as_num = last_checkup_date_.Year() * 10000 +
                         last_checkup_date_.Month() * 100 +
                         last_checkup_date_.Day();
  int curr_date_as_num = (checkup_date.Year() * 100 +
                          checkup_date.Month()) * 100 +
                          checkup_date.Day();
  if (last_date_as_num < curr_date_as_num) {
    last_checkup_date_ = checkup_date;
    return true;
  } else {
    return false;
  }
}
```

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

Reflector questions

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

2. What prior concept we discussed was most useful in understanding the discussion today?

3. What was your team's strategy for completing questions involving writing C++ code?