# CS 200 Lab 07b: Inheritance

## Topics

- Classes

- Inheritance

## Starting out

Make sure to download the following code files:

- main.cpp

- Question.hpp

- Question.cpp

- Quizzer.hpp

- Quizzer.cpp

Starting off, we won't use all the files at first.

Create a new project, and only import **Question.hpp** and **Question.cpp** , and create **main.cpp** with just the standard program starting point.

Start off with the code on the next page.

**Question.hpp**

```cpp
#ifndef _QUESTION_HPP
#define _QUESTION_HPP

#include <string>
#include <iostream>
using namespace std;

class Question
{
};

class TrueFalseQuestion : public Question
{
};

class MultipleChoiceQuestion : public Question
{
};

#endif
```

**Question.cpp**

```cpp
#include "Question.hpp"
```

**main.cpp**

```cpp
#include <iostream>
#include <string>
using namespace std;

#include "Question.hpp"

int main()
{
    return 0;
}
```

## The Question family

First, we will build out the **Question** base class.

     We aren't going to create any Question objects, but it is the starting point for all other questions - this class will contain that which is in common to all other Questions.

### Question class

#### Member variables

| Accessibility | Data Type | Variable Name |
| --- | --- | --- |
| protected | string | m_questionText |

#### Member functions

| Accessibility | Return Type | Function Name | Parameters |
| --- | --- | --- | --- |
| public | void | Display | - |
| public | void | SetQuestionText | string text |

#### void Display()

Use a `cout` statement to display the value of `m_questionText`.

> **Display the value of the variable...**
>     `cout << VARIABLE << endl;`
>    The **variable** is `m_questionText`

### void SetQuestionText( string text )

This function will be reused by the child classes. Write an assignment statement that will set the member variable, `m_questionText`, to the value of the parameter passed in.

> **Assignment statement**
>   VARIABLE = VALUE;
>   The **parameter** is `string text`
>   The **variable** is `m_questionText`

> **Common mistake**
>   Make sure that you're not *redeclaring* the variable `m_questionText` – You shouldn't be using its data type here.
>
>   `m_questionText` is already a **member** of the Question class. Just use it by name.

### Testing
Before continuing, test your Question class!

Within `main()`, declare a Quesiton object, and make sure that both `SetQuestion` and `Display` function calls work correctly.

### TrueFalseQuestion class

This question type will display the question, then ask the user to enter *true* or *false* , and then figure out if they answered correctly.

This means that we need functions to set whether *true* or *false* is the correct answer, as well as to check the user's response.

#### Member variables

| Accessibility | Data Type | Variable Name |
|---|---|---|
| private | string | m_correctAnswer |

#### Member functions

| Accessibility | Return Type | Function Name | Parameters |
|---|---|---|---|
| public | void | Display | - |
| public | void | SetCorrectAnswer | string correctAnswer |
| public | bool | CheckAnswer | string userAnswer |

#### void Display()

This function should call the parents' version of the `Display` function first, then add its own unique code in.

```
void TrueFalseQuestion::Display()
{
    Question::Display();
}
```

Afterward, display another message, asking,
*"True or false?"*

### void SetCorrectAnswer( string correctAnswer )

This function is responsible for assigning the *value* of the parameter `correctAnswer` to the member variable `m_correctAnswer`.

### bool CheckAnswer( string userAnswer )

This function receives the answer that the user gave, as the parameter `userAnswer`.

This function should compare `userAnswer` to the member variable `m_correctAnswer` in order to decide if the user was correct or not.

- If `userAnswer` and `m_correctAnswer` match, then return `true`.

- Otherwise, return `false`.

### Testing
Before continuing, test your TrueFalseQuestion class!

Within `main()`, declare a TrueFalseQuestion object. Use `SetQuestionText` to set its question and `SetCorrectAnswer` to set the correct answer, then use `Display` to view the question text, and `CheckAnswer` to see if it correctly detects right and wrong answers.

```cpp
int main()
{
    TrueFalseQuestion tfQuestion;

    tfQuestion.SetQuestionText( "Is Kansas a state?" );
    tfQuestion.SetCorrectAnswer( "true" );

    tfQuestion.Display();
    string answer;
    cin >> answer;

    bool result = tfQuestion.CheckAnswer( answer );

    if ( result == true )
        cout << "Correct answer!" << endl;
    else
        cout << "Wrong answer!" << endl;

    return 0;
}
```

```
Is Kansas a state?
(true/false): true
Correct answer!
```

### MultipleChoiceQuestion class

This type of question will display four options for the user to choose from. The user will select 1, 2, 3, or 4, and only one answer will be right.

This means that we need functions to set the text for options 1, 2, 3, and 4, as well as store whether option 1, 2, 3, or 4 is the correct answer.

#### Member variables

| Accessibility | Data Type | Variable Name |
|---|---|---|
| private | string array, size 4 | m_answers |
| private | int | m_correctAnswer |

#### Member functions

| Accessibility | Return Type | Function Name | Parameters |
|---|---|---|---|
| public | void | Display | - |
| public | void | SetAnswerChoices | string a, string b, string c, string d |
| public | void | SetCorrectAnswer | int correctAnswer |
| public | bool | CheckAnswer | int userAnswer |

#### void SetAnswerChoices( string a, string b, string c, string d )

Set each of the elements of the array `m_answers` to one of the parameters.

Answer #0 = a    Answer #1 = b
Answer #2 = c    Answer #3 = d

#### void Display()

Once again, call the `Question` class' version of `Display()`, and then use a **for loop** to iterate over all 4 options in the member array, `m_answers`, displaying them to the screen.

#### void SetCorrectAnswer( int correct )

The parameter `correct` stores the *index* of the `m_answers` element that is storing the correct answer.

Store this value in the `m_correctAnswer` member variable.

### bool CheckAnswer( int userAnswer )

This function receives the answer that the user gave, as the parameter `userAnswer` .

This function should compare `userAnswer` to the member variable `m_correctAnswer` in order to decide if the user was correct or not.

- If `userAnswer` and `m_correctAnswer` match, then return `true` .

- Otherwise, return `false` .

### Testing

Write a test that creates a MultipleChoiceQuestion, sets the question with `SetQuestionText` , sets the answer choices with `SetAnswerChoices` , sets the correct answer with `SetCorrectAnswer` , displays the question with `Display` , and checks if the user's answer was right with `CheckAnswer` .

```cpp
int main()
{
    MultipleChoiceQuestion mcQuestion;

    mcQuestion.SetQuestionText( "What is the capital of
    Kansas?" );
    mcQuestion.SetAnswerChoices( "Topeka", "Wichita",
                                 "Kansas City", "Boise" )
    ;
    mcQuestion.SetCorrectAnswer( 0 );

    mcQuestion.Display();
    int answer;
    cout << ">> ";
    cin >> answer;

    bool result = mcQuestion.CheckAnswer( answer );

    if ( result == true )
        cout << "Correct answer!" << endl;
    else
        cout << "Wrong answer!" << endl;

    return 0;
}
```

```
What is the capital of Kansas?

OPTIONS:
0. Topeka
1. Wichita
2. Kansas City
3. Boise
>> 2
Wrong answer!
```

## The full program

Now that your Question classes are working, import in the **Quizzer.hpp** and **Quizzer.cpp** files to your project, and overwrite **main.cpp** with the file provided.

## Quizzer.hpp

```cpp
#ifndef _QUIZZER_HPP
#define _QUIZZER_HPP

#include "Question.hpp"

class Quizzer
{
public:
    Quizzer();

    void AddTrueFalseQuestion( TrueFalseQuestion* q );
    void AddMultipleChoiceQuestion(
    MultipleChoiceQuestion* q );

    void Run();

private:
    TrueFalseQuestion* m_tfQuestions[3];
    MultipleChoiceQuestion* m_mcQuestions[3];

    int m_count_tfQuestions;
    int m_count_mcQuestions;
};
```

## Quizzer.cpp

```cpp
#include "Quizzer.hpp"

#include <iostream>
#include <string>
using namespace std;

Quizzer::Quizzer()
{
    m_count_mcQuestions = 0;
    m_count_tfQuestions = 0;
}

```

```
13  void Quizzer::AddTrueFalseQuestion( TrueFalseQuestion* q
        )
14  {
15      if ( m_count_tfQuestions >= 3 ) { return; }
16      m_tfQuestions[ m_count_tfQuestions++ ] = q;
17  }
18
19  void Quizzer::AddMultipleChoiceQuestion(
        MultipleChoiceQuestion* q )
20  {
21      if ( m_count_mcQuestions >= 3 ) { return; }
22      m_mcQuestions[ m_count_mcQuestions++ ] = q;
23  }
24
25  void Quizzer::Run()
26  {
27      int totalQuestions = m_count_tfQuestions +
        m_count_mcQuestions;
28      int totalRight = 0;
29
30      for ( int i = 0; i < m_count_tfQuestions; i++ )
31      {
32          m_tfQuestions[ i ]->Display();
33
34          string answer;
35          cin >> answer;
36
37          bool correct = m_tfQuestions[i]->CheckAnswer(
        answer );
38
39          if ( correct )
40          {
41              cout << "CORRECT!" << endl;
42              totalRight++;
43          }
44          else
45          {
46              cout << "INCORRECT!" << endl;
47          }
48      }
49
50      for ( int i = 0; i < m_count_mcQuestions; i++ )
51      {
52          m_mcQuestions[ i ]->Display();
53
54          int answer;
55          cin >> answer;
56
```

```
57         bool correct = m_mcQuestions[i]->CheckAnswer(
   answer );
58
59         if ( correct )
60         {
61             cout << "CORRECT!" << endl;
62             totalRight++;
63         }
64         else
65         {
66             cout << "INCORRECT!" << endl;
67         }
68     }
69
70     cout << endl << endl;
71     cout << "Final Score: " << totalRight << " out of "
   << totalQuestions << endl;
72 }
```

## main.cpp

```cpp
#include <iostream>
using namespace std;

#include "Quizzer.hpp"

int main()
{
    Quizzer quizzer;

    TrueFalseQuestion tf1, tf2, tf3;
    tf1.SetQuestionText( "Static arrays can be resized
at run-time." );
    tf1.SetCorrectAnswer( "false" );
    quizzer.AddTrueFalseQuestion( &tf1 );

    tf2.SetQuestionText( "Classes can contain member
variables and functions." );
    tf2.SetCorrectAnswer( "true" );
    quizzer.AddTrueFalseQuestion( &tf2 );

    tf3.SetQuestionText( "It is good practice to set a
pointer to nullptr when not in use." );
    tf3.SetCorrectAnswer( "true" );
    quizzer.AddTrueFalseQuestion( &tf3 );

    MultipleChoiceQuestion mc1, mc2, mc3;
    mc1.SetQuestionText( "Which of the following is the
address-of operator?" );
    mc1.SetAnswerChoices( "&", "*", "->", "::" );
    mc1.SetCorrectAnswer( 0 );
    quizzer.AddMultipleChoiceQuestion( &mc1 );

    mc2.SetQuestionText( "Dynamic variables are
allocated on the..." );
    mc2.SetAnswerChoices( "stack", "heap", "queue", "
array" );
    mc2.SetCorrectAnswer( 1 );
    quizzer.AddMultipleChoiceQuestion( &mc2 );

    mc3.SetQuestionText( "When a value is being passed
into a function call, it is known as a..." );
    mc3.SetAnswerChoices( "parameter", "structure", "
reference", "argument" );
    mc3.SetCorrectAnswer( 3 );
    quizzer.AddMultipleChoiceQuestion( &mc3 );

    quizzer.Run();
```

```
40
41        return 0;
42 }
```

## Run and test

Run the program and make sure it works with your code

```
Static  arrays  can  be  resized  at  run−time.
(true/false):  false
CORRECT!

Classes  can  contain  member  variables  and
    functions.
(true/false):  true
CORRECT!

It  is  good  practice  to  set  a  pointer  to  nullptr
    when  not  in  use.
(true/false):  true
CORRECT!

Which  of  the  following  is  the  address−of
    operator?

OPTIONS:
0.  &
1.  *
2.  −>
3.  ::
0
CORRECT!

Dynamic  variables  are  allocated  on  the...

OPTIONS:
0.  stack
1.  heap
2.  queue
3.  array
1
CORRECT!

When  a  value  is  being  passed  into  a  function
    call ,  it  is  known  as  a...
```

```
OPTIONS:
0. parameter
1. structure
2. reference
3. argument
3
CORRECT!


Final Score: 6 out of 6
```