Lab 1

Lab 1 - Making the Base Class and Derived Class

In this lab, we will be creating a **base** class called `Line`. `Line` has only one attribute, `int length`, which is used in the class function `DrawLine`. `DrawLine` takes an integer parameter and outputs `*` as many times as specified in `length`. To retrieve `length`, we also have the getter function `GetLength`. `Line` does not have a setter function.

```
//add class definitions below this line

class Line {
  public:
    Line(int l) {
      length = l;
    }

    int GetLength() {
      return length;
    }

    void DrawLine() {
      for (int i = 0; i < length; i++) {
        cout << '*';
      }
      cout << endl;
    }

  private:
    int length;
};

//add class definitions above this line
```

To test our `Line` class, create its object and then call the `DrawLine` function on it in `main`.

```
  //add code below this line

  Line line(10);
  line.DrawLine();

  //add code above this line
```

The output is a line drawn with 10 `*` symbols. Next, we will create a **derived** class, `Box`, that inherits from `Line`.

`Box` has one attribute, `int width`, which will present the width of the Box object. The `Box` constructor takes two parameters, one of which is presented by `width` and the other is presented by `length` which is inherited from the `Line` constructor. `Box` has two class functions, the getter `GetWidth` and `DrawBox`. Notice how inheritance enables us to borrow functions and attributes from the base class to further extend the derived class.

`DrawBox` utilizes the `width` attribute to tell the system how many times to call `DrawLine`. The end result is a draw of a “box” that is created from multiple “lines”.

```
//add class definitions below this line

class Line {
  public:
    Line(int l) {
      length = l;
    }

    int GetLength() {
      return length;
    }

    void DrawLine() {
      for (int i = 0; i < length; i++) {
        cout << '*';
      }
      cout << endl;
    }

  private:
    int length;
};

class Box : public Line {
  public:
    Box(int l, int w) : Line(l) {
      width = w;
    }

    int GetWidth() {
      return width;
    }

    void DrawBox() {
      for (int i = 0; i < width; i++) {
        DrawLine();
      }
    }

  private:
    int width;
};

//add class definitions above this line
```

Run the code below in `main` to see the output.

```
  //add code below this line

  Box box(10, 10);
  box.DrawBox();

  //add code above this line
```

challenge

Try these variations:

- Change `Box box(10, 10);` to `Box box(3, 6);`.
- Change `Box box(3, 6);` to `Box box(8, 4);`.

Lab 2

Lab 2 - Applying Multilevel Inheritance

Previously in Lab 1, we created the base class `Line` and the derived class `Box`. In this lab, we will create another derived class that inherits from `Box` directly and `Line` indirectly. This concept of a derived class inheriting from another derived class is called **multilevel inheritance**.

▼ **Given Code**

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

//add class definitions below this line

class Line {
  public:
    Line(int l) {
      length = l;
    }

    int GetLength() {
      return length;
    }

    void DrawLine() {
      for (int i = 0; i < length; i++) {
        cout << '*';
      }
      cout << endl;
    }

  private:
    int length;
};

class Box : public Line {
  public:
    Box(int l, int w) : Line(l) {
      width = w;
    }
```

```
    int GetWidth() {
      return width;
    }

    void DrawBox() {
      for (int i = 0; i < width; i++) {
        DrawLine();
      }
    }

  private:
    int width;
};

//add class definitions above this line

int main() {

  //add code below this line



  //add code above this line

  return 0;

}
```

Our new derived class is called `Pattern`. This class builds off of `Box` and `Line` by utilizing their getter functions `GetLength` and `GetWidth`. First we need to build the `Pattern` constructor which inherits the `Box` constructor exactly. Then we will create a new function called `DrawPattern` that will output a modified "box" with a pattern. Note that `Pattern` does not have any private members. It is simply an extension of the `Box` class.

```
//add class definitions below this line

class Line {
  public:
    Line(int l) {
      length = l;
    }

    int GetLength() {
```

```
      return length;
    }

    void DrawLine() {
      for (int i = 0; i < length; i++) {
        cout << '*';
      }
      cout << endl;
    }

  private:
    int length;
};

class Box : public Line {
  public:
    Box(int l, int w) : Line(l) {
      width = w;
    }

    int GetWidth() {
      return width;
    }

    void DrawBox() {
      for (int i = 0; i < width; i++) {
        DrawLine();
      }
    }

  private:
    int width;
};

class Pattern : public Box {
  public:
    Pattern(int l, int w) : Box(l, w) {}

    void DrawPattern() {
      for (int i = 0; i < GetLength(); i++) {
        if (i % 2 == 0) {
          for (int j = 0; j < GetWidth(); j++) {
            if ( (j % 2 == 0) ) {
              cout << '*';
            }
            else {
              cout << ' ';
            }
```

```
            }
            cout << endl;
          }
          if (i % 2 == 1) {
            for (int j = 0; j < GetWidth(); j++) {
              if ( (j % 2 == 0) ) {
                cout << ' ';
              }
              else {
                cout << '*';
              }
            }
            cout << endl;
          }
        }
      }
};

//add class definitions above this line
```

Rather than create a “box” of “lines”, a “pattern” is created by specifying certain indices to output `*` while others output a white space (`' '`). In particular, this is the pattern (note the pattern starts at index `0` for both rows and columns):

- even row + even column = `*`
- even row + odd column = `' '`
- odd row +even column = `' '`
- odd row + odd column = `*`

Try the following code in `main` to see the output.

```
  //add code below this line

  Pattern pattern(10, 10);
  pattern.DrawPattern();

  //add code above this line
```

challenge

Try these variations:

- Change `Pattern pattern(10, 10);` to `Pattern pattern(3, 6);`.
- Change `Pattern pattern(3, 6);` to `Pattern pattern(8, 20);`.

Lab Challenge

Problem

In the IDE to the left, the class `MP3` is already defined. Complete the class `Podcast` that inherits from `MP3`. This class should do the following things:
* Inherit the constructor such that `Podcast` has the following attributes:
* `title` - a string that is the title of the episode
* `length` - an integer that has the length of the podcast **in seconds**
* `genre` - a string that is the genre of the podcast
* `name` - a string that is the name of the podcast
* `date` - a string that represents when the podcast was released to the public
* **DO NOT EDIT** the specified code or you might not receive credit for your work!

▼ Hint: Connecting the Constructors

Note that a few of the attributes are present in both the `MP3` and `Podcast` classes. To connect their constructors, you can use the same parameter values for the attributes that are the same, then use different parameter values for the attributes that are different. Finally, set the attributes to the parameters appropriately for the ones that are different. For example:

```
Podcast(string t, int l, string g, string n, string d) : MP3(t,
        l, g, n, d) {
  name = n;
  date = d;
}
```

▼ Given Code

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

//DO NOT EDIT code below this line

class MP3 {
  public:
    MP3(string t, int l, string g, string al, string ar) {
      title = t;
      album = al;
```

```
    length = l;
    genre = g;
    artist = ar;
  }

  string GetTitle() {
    return title;
  }

  void SetTitle(string new_title) {
    title = new_title;
  }

  int GetLength() {
    return length;
  }

  void SetLength(int new_length) {
    length = new_length;
  }

  string GetGenre() {
    return genre;
  }

  void SetGenre(string new_genre) {
    genre = new_genre;
  }

  string GetAlbum() {
    return album;
  }

  void SetAlbum(string new_album) {
    album = new_album;
  }

  string GetArtist() {
    return artist;
  }

  void SetArtist(string new_artist) {
    artist = new_artist;
  }

private:
  string title;
  int length;
```

```
    string genre;
    string album;
    string artist;
};

//DO NOT EDIT code above this line

//add class definitions below this line

//DO NOT EDIT//////////////////
class Podcast : public MP3 {  //
//////////////////////////////




//add class definitions above this line

int main() {

  //DO NOT EDIT code below this line

  Podcast p("Hollywood Black List", 1460, "economics", "Planet
        Money", "10 July 2020");
  p.DisplayTitle();
  p.DisplayLength();
  p.DisplayGenre();
  p.DisplayName();
  p.DisplayDate();

  //DO NOT EDIT code above this line

  return 0;

}
```

Expected Output

```
The title is Hollywood Black List
The length is 1460
The genre is economics
The name is Planet Money
The date is 10 July 2020
```

Testing Your Code

Use the button below to test your code before submitting it for evaluation.