CISP 400 - Lab Assignment #7
Due Thursday, October 18, 2018 at 11:59 PM

This lab is intended as an exercise with simple class inheritance, as well as friend overloaded operators, specifically the double chevron.

SCENARIO:

We are writing a couple of objects for software for a theater. The theater offers two ways to sell tickets: just the tickets by themselves, or a ticket packaged with a snack for a discount.

The ticket objects need to "know" about the movie. In this case, they need to store the name of the movie and the run time of the movie. The package objects need to know what the ticket knows, plus the name of the snack.

CLASSES:

There are two classes you will write – Ticket and Package.

TICKET CLASS

Ticket will have methods to get the movie name and run time. This means it will have getMovieName() and getRunTime() getter methods. The only constructor will take two parameters that set these two values. The constructor will use the initializer list to set these values, so the constructor body should be empty. There will also be two setters – setMovieName() and setRunTime() – that set these variables outside of the constructor. THERE WILL BE NO CONSTRUCTOR WITH NO PARAMETERS.

There will be two member variables – an int to hold the run time, and a string to hold the movie name.

There will also be a friend overloaded operator, a <<, which can be used to output to cout or other files. We'll talk about that in a moment.

PACKAGE CLASS

The second class will be Package. **Package will be the child class of Ticket.** It will also include additional members to set and get the snack name, getSnackName() and setSnackName(). The only constructor will take the parameters the Ticket constructor does, as well as the snack name. The Ticket constructor will be called in the initializer list, and the snack name will be initialized in the initializer list, so the constructor body will be empty. THERE WILL BE NO CONSTRUCTOR WITH NO PARAMETERS.

There will be one member variable, a string to hold the snack name.

As with Ticket, there will be a friend overloaded operator, <<. So you will need to write two of them. One will be for Ticket, and one will be for Package. The one for Ticket will need to output the movie name and run time. The one for Package will need to output the movie name, the run time, and the snack name.

As such, if we use this code…

```
Package p("My Movie Name", 3600, "Popcorn");
cout << p << endl;
```

We should get something that resembles this:

```
Movie: My Movie Name
Run Time: 3600
Snack: Popcorn
```

DRIVER PROGRAM:

The driver program will print out a main menu that asks the user to select whether or not they want to input data for a ticket, input data for a package, or exit.

When the user picks a ticket, they should be prompted for the movie name and run time. The program will then DYNAMICALLY allocate a Ticket object. It will use the paramaterized CONSTRUCTOR to give the movie name "N/A" and run time of zero. It will then use the SETTERS to input the user's response for movie name and run time. Finally it will use cout << to print it the Ticket object out. Ticket objects should NOT take any user input by themselves.

When the user picks a package, they should be prompted for the movie name, run time, and snack name. The program will then DYNAMICALLY allocate a Package object. The paramaterized CONSTRUCTOR will be given the movie name "N/A PACKAGE" with a run time of 0 and a snack name of "NONE". It will then use the SETTERS to put in the user responses, specifically the movie name, run time, and snack name (remember, its parent class is Ticket!), and then use cout << to print the Package object out. Package objects should NOT take any user input by themselves.

Either of these options should end with the Ticket or Package deleted.

The user will then be returned to the main menu. This will continue until the user chooses to exit.

When the user exits, the menu loop stops and the program ends.

HINTS:

- You will need to use the Ticket member functions for the Package overloaded operator.
- **<u>You must make Package the child class/subclass of Ticket to get credit.</u>**
- Remember to dereference pointers, especially with overloaded operators.
- There are examples posted for both the friend overloaded operator and basic inheritance. Use them.

DELIVERABLES:

Turn the .cpp file into Canvas by the due date.