Final Project

Create a program that maintains an inventory of items and displays unique information about each item.

1. Pick a problem domain that you are interested in modeling (e.g., books, movies, pets, sports teams...).

2. Create the classes that model your domain. There must be at least one example of a class that is derived from a base class. For exam TextBook is dirvied from Book, TelevisionMovie is dirvied from Movie, AquaticPet is dirived from Pet, BasketballTeam is derived from Team.

Hint: make sure you inherit with public, this will be important for step 5, for example:

class TelevisionMovie: public Movie

3. Create a menu system similar to the following.

Main menu: Select an option.

1. Add item

2. Remove item

3. Display all items  
  
Menu option 1: What type of item will be added?  
1. Add item type 1  
2. Add item type 2  
3. Add item type 3  
  
Menu option 2: Enter the index of the item that should be removed?  
  
Menu option 3: Displaying all items...

4. Each class should have its own display function that is used to display an object of that class.

5. You must use an STL vector to store the items. The problem can be solved with regular pointers or with smart pointers. Because we have not covered smart pointers all of the relevant operations are provided below. For smart pointers to work you must change c++11 to c++14 in the projects .pro file. You can also use the online compiler at https://onlinegdb.com to test your smart pointers using c++14.

// note: you must change c++11 to c++14 in the .pro file

// create a vector of your base class

vector<unique\_ptr<Movie>> movies;

// add a new objects of the base class to the vector

movies.push\_back(make\_unique<Movie>("The Wizrd of Oz", 1939));

// add a new object of a derived class to the vector

movies.emplace\_back(make\_unique<TelevisionMovie>("Saga of a Star World", 1978, 3));

// remove an item from the vector

// we will use a variable to indicate which item to remove (location 0 means remove the first item)

int index = 0;

movies.erase(movies.begin() + index);

// display all items in the vector

for (auto& movie: movies) {

// note: this depends on polymorphism,

// it only works if display is virtual

movie->display();

}

**For full credit the project must have the following components:**

1. A problem statement.
2. UML diagrams of all of the classes.
3. The final program (all .cpp and .h files). The code should be documented with comments.
4. Follow the code guidlines below.

**Code guidelines:**

* **Good style:** Use proper indentation, no magic numbers, good naming conventions.
* **Usability:** Always prompt the user for input so they know what to do and provide meaningful output messages.
* **Input Validation:** The program should not accept invalid input, prompt the user to reenter an input that is invalid.
* **Documentation:** Add a comments that document what each part of your code does, at a minimum each function should be clearly documented by describing the expected argument values and what the function returns.
* **Testing:** Make sure your code compiles and works.