Lab 4 Report:

By: Sam Graler and Randy Hucker (Group 25)

Instructions for Running / Compiling:
.cpp files for all Tasks are located in the same visual studio project. Open / run the visual studio project as normal.

Write a Lab Report that includes the following information:

a. A description of the objectives/concepts explored in this assignment including why you think they are important to this course and a career in CS and/or Engineering.

The topics examined in this lab were inheritance, polymorphism, and abstract classes (kind of). These concepts are important to the field of computer science because they expand what we are able to do with classes, which are essential organizational structures. Inheritance continues to expand our ability to reuse code by allowing us to make more specific derived classes from base classes without rewriting all the public members and functions in the base class. Polymorphism is what allows us to take advantage of virtual functions and function overriding, again expanding our ability to add / change functionality to derived classes without rewriting the base over and over again. Abstract classes are good to use essentially as templates, but we didn't use an abstract class in this assignment. These concepts all promote a helpful way of thinking about encapsulation and breaking down a complex issue into a series of more manageable problems.

b. The sections from each task indicated to be included in the lab report.

Task 1:
3. Include in the submission how each member will be available in derived classes (i.e. not available, available if not overridden, etc…)

Private variables in the Show Class won't be able to be accessed directly by either derived class, but they will be available to be updated or retrieved using the setters and getters that we have as public member functions. The constructors and destructors will be available as normal (with the deconstructor normally being run automatically) because they are public members. The Details member function will be available if it isn't overridden, and so will the Play function, however, if you try to access Show's Play function from a derived class instance it won't work because Play is a virtual function.

Task 2:

4. Include in the submission what version of the derived class members will be available in instances of the derived class and in instances of the derived class declared as the base class type.

When details is called, it will run dependant on the pointer that is pointing at it - ie. a Base class pointer is looking at a TvShow, the base class' details will still be invoked, for a TvShow instance, the TvShow's details function will be called. For a movie instance (whether it is created as a show or a movie), Show's details will be called because the movie class doesn't override details.

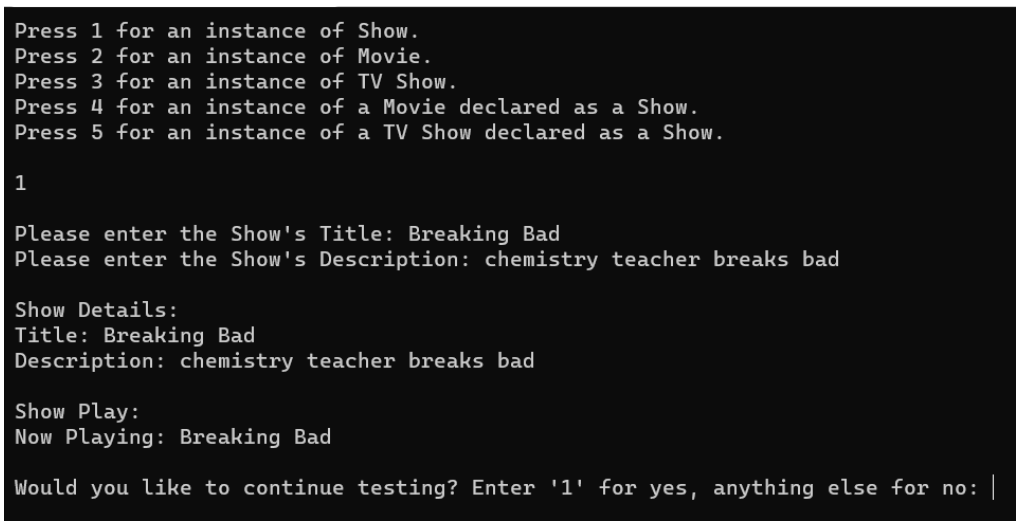
For play, a less specific pointer observing a more specific pointer will call the most specific associated version of "play" - ie. A pointer of Show pointing at an instance of TvShow will call Tv Show's Play, similarly in the case associated with Movie. Attributes / member functions specific to derived classes cannot be accessed from the base class.

Task 3:

3. Include in the lab report a screenshot(s) of the output of a test. Include a discussion of how the actual results compared with the expected results from Task 2.

The results appeared as expected when we thought about it in Task 2. The results needed reformatting to be more visually appealing, but the information that was delivered was correct. We had a conversation with the TA discussing how to properly deliver the Details function, which provoked us to use a template function which allowed us to pass in a pointer of any of the three classes we created. Without that piece, we were persistently invoking the Show class' details and play due to the inherent casting to Show made by the function described in the instructions (one function accepting a show pointer used for all of the cases described)

Screenshots:



Uses Show’s member functions

```
Press 1 for an instance of Show.
Press 2 for an instance of Movie.
Press 3 for an instance of TV Show.
Press 4 for an instance of a Movie declared as a Show.
Press 5 for an instance of a TV Show declared as a Show.

2

Please enter the Movie's Title: Doctor Strange
Please enter the Movie's Description: doctor becomes strange (a wizard)
Please enter the Movie's Credits: Marvel

Show Details:
Title: Doctor Strange
Description: doctor becomes strange (a wizard)

Movie Play:
Now Playing: Doctor Strange
Credits: Marvel

Would you like to continue testing? Enter '1' for yes, anything else for no:
```

Uses Movie’s play and show’s details (movie didn’t override details)

```
Press 1 for an instance of Show.
Press 2 for an instance of Movie.
Press 3 for an instance of TV Show.
Press 4 for an instance of a Movie declared as a Show.
Press 5 for an instance of a TV Show declared as a Show.

3

Please enter the TvShow's Title: Breaking Bad (again)
Please enter the TvShow's Description: chem teacher cooks meth
Please enter the number of Seaons: 4
Please enter the number of Episodes in each season: 3


TV Show Details:
Title: Breaking Bad (again)
Description: chem teacher cooks meth
Number of Seasons: 4

TV Show Play:
Please enter the Season you'd like to play: 2
Please enter the Episode you'd like to play: 3

Now Playing: Season: 2 Episode: 3 of Breaking Bad (again)

Would you like to continue testing? Enter '1' for yes, anything else for no:
```

Uses TvShow’s play and details because TvShow overrides details and play is virtual

```
Press 1 for an instance of Show.
Press 2 for an instance of Movie.
Press 3 for an instance of TV Show.
Press 4 for an instance of a Movie declared as a Show.
Press 5 for an instance of a TV Show declared as a Show.

4

Please enter the Movie's Title: The Bee Movie
Please enter the Movie's Description: bees
Please enter the Movie's Credits: DreamWorks

Show Details:
Title: The Bee Movie
Description: bees

Movie Play:
Now Playing: The Bee Movie
Credits: DreamWorks

Would you like to continue testing? Enter '1' for yes, anything else for no:
```

Uses Movie's play because it is a vitrul function, still uses show details (this time because is it a movie instance as a show (even if movie overrode details, this option would still call show's detials)

```
Press 1 for an instance of Show.
Press 2 for an instance of Movie.
Press 3 for an instance of TV Show.
Press 4 for an instance of a Movie declared as a Show.
Press 5 for an instance of a TV Show declared as a Show.

5

Please enter the TvShow's Title: Breaking Bad (a third time)
Please enter the TvShow's Description: jesse
Please enter the number of Seasons: 4
Please enter the number of Episodes in each season: 3

Show Details:
Title: Breaking Bad (a third time)
Description: jesse

TV Show Play:
Please enter the Season you'd like to play: 1
Please enter the Episode you'd like to play: 3

Now Playing: Season: 1 Episode: 3 of Breaking Bad (a third time)

Would you like to continue testing? Enter '1' for yes, anything else for no:
```

Uses TvShow's play because play is virtual, uses show's details because the TvShow instance in this case was created as a Show instance.