**Lab Report**

**Understanding Inheritance, Polymorphism, and Abstract Classes in Object-Oriented Programming**

By: Ishaa Jain, Nipun Chandra, Shivam Kharangate

**Introduction:**

This lab focuses on demonstrating the object-oriented programming (OOP) concepts of Inheritance, Polymorphism, and Abstract Classes through a practical application. By creating a base class named ‘Show’ and two derived classes, ‘TVShow’ and ‘Movie’, we explored how these OOP principles helps with the design and implementation of a software system that models an online streaming platform.

**Objectives:**

1. **Inheritance**: To make the code more reusable and hierarchically organized by deriving **TVShow** and **Movie** classes from a common base class, **Show**.
2. **Polymorphism**: To utilize virtual functions to enable objects of derived classes to be treated as objects of the base class.
3. **Abstract Classes**: While the provided **Show** class isn't abstract (it doesn't contain pure virtual functions), the concept is touched upon using virtual functions intended for overriding.

These concepts are fundamental to creating efficient, maintainable, and scalable software, making them crucial for careers in Computer Science and Engineering.

**Task 1: Base Class Implementation**

The **Show** class serves as the foundation for representing various types of content on a streaming platform. It includes basic attributes (**title**, **genre**), constructors, getters, and setters for these attributes, a virtual **Play** method, and a **DE** to display the show's details.

**Features of Show**

* **Constructors**: Initialized objects with optional title and genre.
* **Accessors and Mutators**: Allowed reading and modification of the attributes of the object.
* **Virtual Play Method**: Was a placeholder for derived class-specific implementations.
* **DE Method**: Displayed the title and genre of the show.

**Task 2: Derived Classes and Polymorphism**

**TVShow Class** is derived from **Show**. **TVShow** introduced additional attributes to handle seasons and episodes, showing inheritance. The **Play** method was overridden to enable playing a specific episode, which showed polymorphism.

**Features:**

* **Array**: Helped manage episodes across seasons.
* **Constructor and Destructor**: Handled memory allocation.
* **Overridden Play Method**: Allowed selection and "playing" of a specific episode.
* **Details Method**: Extended **Show**'s **DE** method by adding season and episode information.

**Movie class** wasalso derived from **Show**, **Movie** focused on representing movies with an added **openingSequence** attribute. It demonstrated polymorphism by providing its version of the **Play** method.

* **openingSequence Attribute**: Stored movie-specific opening credits.
* **Overridden Play Method**: Displayed the opening credits.

**Task 3: Testing the System**

The main function presented interface in form of a menu that allowed users to create and interact with **Movie** and **TVShow** objects which invoked their respective functions, namely, ‘Play’ and ‘DE’. This made it possible to show inheritance and the use of abstract classes, as well as show polymorphic behavior through a single interface.

The program prompted users to choose between creating a **Movie** or **TVShow** instance. This design choice emphasized the polymorphism in OOP, where base class references could be used to interact with derived class objects.

**Conclusion**

This lab showed the importance of Inheritance, Polymorphism, and Abstract Classes in designing flexible software systems. Through hands-on implementation of a simplified streaming service model, we saw how these OOP principles enabled code reusability, extendibility, and its dynamic behavior. Such knowledge is important in the field of Computer Science and Engineering, where the ability to design and implement software systems is crucial.

Screenshots of the output:

A screenshot of a computer

Description automatically generated

A computer screen with white text

Description automatically generated

References:

* chatgpt