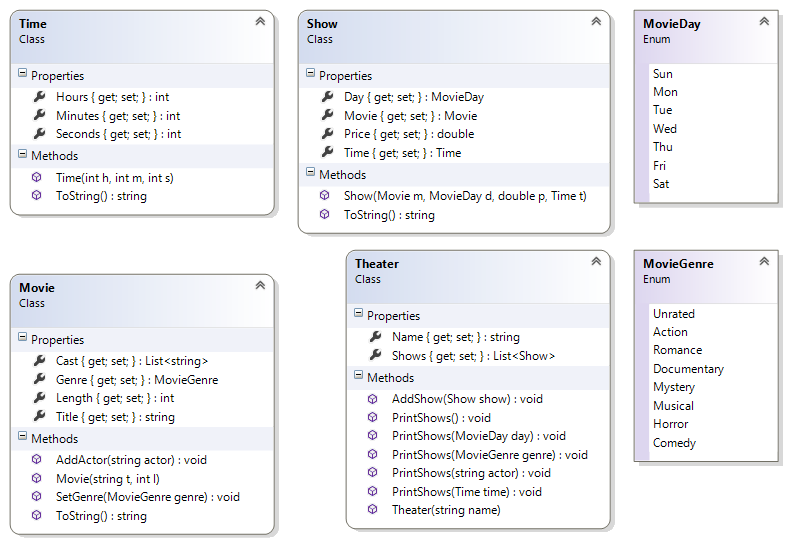
Programming II

# Assignment 3 – Implement a Theatre Application (Groups of four students)

### Due: Demonstration of your code due at the beginning of the last class in Week 7

This is the most complex application that you have seen so far. It is comprised of six user defined types: four classes and two enums. This assignment attempts to simulate a simple version of a Theater application. In this application the user is able to search for a movie based on genre, name of actor, the time of the day and the day of the week.

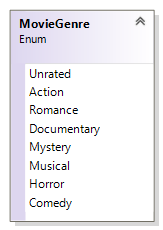
You should define your types in the following order: the two enums, Time, Movie, Show and then Theater You will perform some simple queries on this collection.



## The MovieDay Enum

This type represents the days of the week and is comprised of seven constants. The constants are the first three letters of the day of the week.

## The MovieGenre Enum

This type represents the various categories of movie. Because a movie may fall under multiple categories, you will have to use the [Flags] to decorate this enum.

The bit-wise operator is used to combine more than one genre. E.g. **MovieGenre** genre = **MovieGenre**.Action | **MovieGenre**.Romance | **MovieGenre**.Comedy;

For this to work correctly, you will have to assign appropriate values for each name. Appropriate values are 0, 1, 2, 4, 8, 16, 32 etc.

## The Time Class

This type represents the days of the week.

|  |
| --- |
| **Time**  Class |
| **Properties** |
| + Hours {get; set;} : int  + Minutes {get; set;}: int  + Seconds {get; set;}: int |
| **Methods** |
| + Time(hours : int, minutes : int, seconds : int)  + ToString() : string |

### Properties:

All of the properties have public getters and private setters and are self-explanatory.

1. **Hours** – this property is an int representing the hour. The getter is public and the setter is private.
2. **Minutes** – this property is an int representing the minute. The getter is public and the setter is private.
3. **Seconds** – this property is an int representing the second. The getter is public and the setter is private.

### Methods:

1. **Time(int hours, int minutes, int seconds)** – This public constructor takes three int parameters and assigns them to the appropriate properties
2. **ToString()** – This method overrides the same method of the Object class. It does not take any parameter but return a string representation of itself. You decide on the format for the output.

## The Movie Class

This class will model a movie.

|  |
| --- |
| **Movie**  Class |
| **Properties** |
| + Length {get; set;} : int  + Title {get; set;}: string  + Genre {get; set;}: MovieGenre  + Cast {get; set;}: List<string> |
| **Methods** |
| + Movie(title : string, length : int)  + AddActor(actor : string) : void  + SetGenre(genre : MovieGenre) : void  + ToString() : string |

### Properties:

All of the properties have public getters and private setters and are self-explanatory.

1. **Length** – this property is an int representing the length of the movie in minutes. The getter is public and the setter is private.
2. **Title** – this property is a string representing the title of the movie. The getter is public and the setter is private.
3. **Genre** – this property is an enum representing the genre of this movie. The getter is public and the setter is private.
4. **Cast** – this property is a list of string representing the names of the actors in this movie. The getter is public and the setter is private.

### Methods:

1. **Movie(string name, int length)** – This public constructor takes one string and one int parameter. It does the following:
   1. Assigns the arguments to the appropriate properties.
   2. Initialize the Cast property to
2. **AddActor(string actor)** – This public method takes a single a string argument and adds it to the collection of actors (**Cast**).
3. **SetGenre(MovieGenre genre)** – This public method takes a single enum argument and assigns it to the property of the same name.
4. **ToString()** – This method overrides the same method of the Object class. It does not take any parameter but return a string representation of itself. You will need to build a single string comprising all of the actors in this movie. You decide on the format for the output.

## The Show Class

This class models a movie show. You will also implement this in Visual Studio. A short description of the class members is given below:

|  |
| --- |
| **Show**  Class |
| **Properties** |
| + Price {get; set;} : double  + Day {get; set;} : MovieDay  + Movie {get; set;} : Movie  + Time {get; set;} : Time |
| **Methods** |
| + Show(movie : Movie, day : MovieDay, price : double, time : Time )  + ToString() : string |

### Properties:

1. **Price** – this property is a double representing the price of admission to this show. The getter is public and the setter is private.
2. **Day** – this property is an enum representing the day of the week of this show. The getter is public and the setter is private.
3. **Movie** – this property is an object reference of the movie class. The getter is public and the setter is private.
4. **Time** – this property is an object of the Time class representing the time of this show. The getter is public and the setter is private.

### Methods:

1. **Show(Movie movie, MovieDay day, double price, Time time)** – This is the public constructor that takes four arguments and assigns them to the appropriate properties.
2. **ToString()** – This is the public method overrides the method of the same name in the object class to return a meaningful description of this object.

## The Theater Class

This class models a theater. You will also implement this in Visual Studio. A short description of the class members is given below:

|  |
| --- |
| **Theatre**  Class |
| **Properties** |
| + Shows {get; set;} : List<Show>  + Name {get; set;} : string |
| **Methods** |
| + Theater(name : string)  + AddShow(show : Show) : void  + PrintShows() : void  + PrintShows(genre : MovieGenre) : void  + PrintShows(day : MovieDay) : void  + PrintShows(time : Time) : void  + PrintShows(actor : string) : void |

### Properties:

1. **Shows** – this private field is a list of Show objects. The getter is public and the setter is private.
2. **Name** – this private field is a string representing the name of the theater.

### Methods:

1. **Theater(string name)** – This is the public constructor that takes the name of the theater. This constructor does the following:
   1. Assigns the argument to the property.
   2. Initialize the **Shows** property to a new list of show
2. **AddShow(Show show)** – This public method takes a show object and adds it to the collection of shows.

Again, this is a good example of method overloading

1. **PrintShows()** – This public method does not take any argument neither does it return a value. It displays all the shows that is available.
2. **PrintShows(MovieGenre genre)** – This public method takes a genre as an argument and display all the shows that contains the flag of this genre.
3. **PrintShows(MovieDay day)** – This public method takes a day object as an argument and display all the shows matching this day object.
4. **PrintShows(Time time)** – This public method takes a time object as an argument and display all the shows matching the hour value of this time object.

A better way of approaching this problem, would be to overload the **==** operator of the **Time** class.   
Also, you may choose to match two **Time** object if there are within 15 minutes of each other.

1. **PrintShows(string actor)** – This public method takes a string representing the name of an actor as an argument and display all the shows that this actor appears in.

## Testing

In your test harness (the Main() method in the Program Class), copy and paste the following code:

Movie terminator = new Movie("Judgement Day", 105);

terminator.AddActor("Arnold Schwarzenegger");

terminator.SetGenre(MovieGenre.Horror | MovieGenre.Action);

terminator.AddActor("Linda Hamilton");

Show s1 = new Show(terminator, MovieDay.Mon, 5.95, new Time(11, 35, 0));

Console.WriteLine(s1); //displays one object

Theater eglinton = new Theater("Cineplex");

eglinton.AddShow(s1);

eglinton.PrintShows(); //displays one object

Movie godzilla = new Movie("Godzilla 2014", 123);

godzilla.AddActor("Aaron Johnson");

godzilla.AddActor("Ken Watanabe");

godzilla.AddActor("Elizabeth Olsen");

godzilla.SetGenre(MovieGenre.Action);

Movie trancendence = new Movie("Transendence", 120);

trancendence.AddActor("Johnny Depp");

trancendence.AddActor("Morgan Freeman");

trancendence.SetGenre(MovieGenre.Comedy);

eglinton.AddShow(new Show(trancendence, MovieDay.Sun, 7.87, new Time(18, 5, 0)));

Movie m1 = new Movie("The Shawshank Redemption", 120);

m1.AddActor("Tim Robbins");

m1.AddActor("Morgan Freeman");

m1.SetGenre(MovieGenre.Action);

eglinton.AddShow(new Show(m1, MovieDay.Sun, 8.87, new Time(14, 5, 0)));

m1 = new Movie("Olympus Has Fallen", 120);

m1.AddActor("Gerard Butler");

m1.AddActor("Morgan Freeman");

m1.SetGenre(MovieGenre.Action);

eglinton.AddShow(new Show(m1, MovieDay.Sun, 8.87, new Time(16, 5, 0)));

m1 = new Movie("The Mask of Zorro", 136);

m1.AddActor("Antonio Banderas");

m1.AddActor("Anthony Hopkins");

m1.AddActor(" Catherine Zeta-Jones");

m1.SetGenre(MovieGenre.Action | MovieGenre.Romance);

eglinton.AddShow(new Show(m1, MovieDay.Sun, 8.87, new Time(16, 5, 0)));

m1 = new Movie("Four Weddings and a Funeral", 117);

m1.AddActor("Hugh Grant");

m1.AddActor("Andie MacDowell");

m1.AddActor("Kristin Scott Thomas");

m1.SetGenre(MovieGenre.Comedy | MovieGenre.Romance);

eglinton.AddShow(new Show(m1, MovieDay.Sat, 8.87, new Time(15, 5, 0)));

m1 = new Movie("You've Got Mail", 119);

m1.AddActor("Tom Hanks");

m1.AddActor("Meg Ryan");

m1.SetGenre(MovieGenre.Comedy | MovieGenre.Romance);

eglinton.AddShow(new Show(m1, MovieDay.Sat, 8.87, new Time(15, 5, 0)));

Show s2 = new Show(godzilla, MovieDay.Mon, 6.89, new Time(13, 15, 0));

eglinton.AddShow(s2);

s2 = new Show(godzilla, MovieDay.Sun, 6.89, new Time(14, 15, 0));

eglinton.AddShow(s2);

s2 = new Show(godzilla, MovieDay.Sun, 6.89, new Time(16, 55, 0));

eglinton.AddShow(s2);

eglinton.PrintShows(); //displays ten objects

eglinton.PrintShows(MovieDay.Sun); //displays six objects

eglinton.PrintShows(MovieGenre.Action); //displays seven objects

eglinton.PrintShows(MovieGenre.Romance); //displays three objects

eglinton.PrintShows(MovieGenre.Action | MovieGenre.Romance); //displays one object

eglinton.PrintShows("Morgan Freeman"); //displays three objects

eglinton.PrintShows(new Time(14, 30, 0)); //displays two objects