```
//Copied from Program 0 V3
/*      Grading ID: D1499
 *      Program 1A
 *      Due Date: February 25,2017
 *      Course: CIS 200
 *      Course Section: 01
 *      Program:
 *       Show uses of polymorphsim and inhertiance
 *
 */

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Program_1A
{
   public  abstract class LibraryItem
    {
        public const int DEFAULT_YEAR = 2016; // Default copyright year

        private string _title;      // The LibraryItem's title
        private string _publisher;  // The LibraryItem's publisher
        private int _copyrightYear; // The LibraryItem's year of copyright
        private int _loanPeriod; //Library Item's loan period
        private string _callNumber; // The LibraryItem's call number in the library
        private bool _checkedOut;   // The LibraryItem's checked out status
        private LibraryPatron _patron; // the LibraryItem's patron


        // Precondition:  theCopyrightYear >= 0
        // Postcondition: The library LibraryItem has been initialized with the specified
        //                values for title, author, publisher, copyright year, and
        //                call number. The LibraryItem is not checked out.
        public LibraryItem(string theTitle, string thePublisher,
            int theCopyrightYear, int theLoanPeriod, string theCallNumber)
        {
            Title = theTitle;
            LoanPeriod = theLoanPeriod;
            Publisher = thePublisher;
            CopyrightYear = theCopyrightYear;
            CallNumber = theCallNumber;
            Patron = null;
            ReturnToShelf(); // Make sure LibraryItem is not checked out
        }

        public string Title
        {
            // Precondition:  None
            // Postcondition: The title has been returned
            get
            {
                return _title;
            }

            // Precondition:  value must not be null or empty
            // Postcondition: The title has been set to the specified value
            set
            {
                if (!string.IsNullOrWhiteSpace(value)) // IsNullOrWhiteSpace includes
tests for null, empty, or all whitespace
                    _title = value.Trim();
                else
                    throw new ArgumentOutOfRangeException($"{nameof(Title)}", value,
                        $"{nameof(Title)} must not be null or empty");
            }
        }

        public int LoanPeriod
        {
            // Precondition:  None
            // Postcondition: The LoanPeriod has been returned
            get
            {
                return _loanPeriod;
            }

            // Precondition:  Input must be >= 0
            // Postcondition: The LoanPeriod has been set to the specified value
            set
            {
                if (value >= 0)
                    _loanPeriod = value;
                else
                    throw new ArgumentOutOfRangeException("LoanPeriod", value,
                        $"{nameof(LoanPeriod)} must be >= 0");
            }
        }

        public string Publisher
        {
            // Precondition:  None
            // Postcondition: The publisher has been returned
            get
            {
                return _publisher;
            }

            // Precondition:  None
            // Postcondition: The publisher has been set to the specified value
            set
            {
                // Since empty author is OK, just change null to empty string
                _publisher = (value == null ? string.Empty : value.Trim());
            }
        }

        public int CopyrightYear
        {
            // Precondition:  None
            // Postcondition: The copyright year has been returned
            get
            {
                return _copyrightYear;
            }

            // Precondition:  value >= 0
            // Postcondition: The copyright year has been set to the specified value
            set
            {
                if (value >= 0)
                    _copyrightYear = value;
                else
                    throw new ArgumentOutOfRangeException($"{nameof(CopyrightYear)}",
value,
                        $"{nameof(CopyrightYear)} must be >= 0");
            }
        }

        public string CallNumber
        {
            // Precondition:  None
            // Postcondition: The call number has been returned
            get
            {
                return _callNumber;
            }

            // Precondition:  value must not be null or empty
            // Postcondition: The call number has been set to the specified value
            set
            {
                if (!string.IsNullOrWhiteSpace(value)) // IsNullOrWhiteSpace includes
tests for null, empty, or all whitespace
                    _callNumber = value.Trim();
                else
                    throw new ArgumentOutOfRangeException($"{nameof(CallNumber)}", value,
                        $"{nameof(CallNumber)} must not be null or empty");
            }
        }

        // Create HAS-A
        public LibraryPatron Patron
        {
            // Precondition:  None
            // Postcondition: The LibraryItem's patron has been returned
            get; // Auto-implement is fine

            // Helper
            // Precondition:  None
            // Postcondition: The LibraryItem's patron has been set to the specified
value
            private set; // Auto-implement is fine
        }

        // Precondition:  thePatron != null
        // Postcondition: The LibraryItem is checked out
        public void CheckOut(LibraryPatron thePatron)
        {
            _checkedOut = true;
            if (thePatron != null)
                Patron = thePatron;
            else
                throw new ArgumentNullException($"{nameof(thePatron)}",
$"{nameof(thePatron)} must not be null");
        }

        // Precondition:  None
        // Postcondition: The LibraryItem is not checked out
        public void ReturnToShelf()
        {
            _checkedOut = false;
            Patron = null; // Remove previously stored reference to patron
        }

        // Precondition:  None
        // Postcondition: true is returned if the LibraryItem is checked out,
        //                otherwise false is returned
        public bool IsCheckedOut()
        {
            return _checkedOut;
        }

        //abstract class for CalcLateFee to be implemented
        abstract public decimal CalcLateFee(int lateDays);

        public string NL = Environment.NewLine; // NewLine shortcut (added here to be
used for all inherited classes easily)

        // Precondition:  None
        // Postcondition: A string is returned presenting the libary LibraryItem's data
on
        //                separate lines
        public override string ToString()
        {
            string NL = Environment.NewLine; // NewLine shortcut
            string checkedOutBy; // Holds checked out message

            if (IsCheckedOut())
                checkedOutBy = $"Checked Out By: {NL}{Patron}";
            else
                checkedOutBy = "Not Checked Out";

            return $"Title: {Title}{NL}LoanPeriod: {LoanPeriod}{NL}Publisher:
{Publisher}{NL}" +
                $"Copyright: {CopyrightYear:D4}{NL}{checkedOutBy}";
        }
    }
}
```

```
/*     Grading ID: D1499
```

```
 *     Program 1A
 *     Due Date: February 25,2017
 *     Course: CIS 200
 *     Course Section: 01
 *     Program:
 *      Show uses of polymorphsim and inhertiance
 *
 */
 using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Program_1A
{
    abstract class LibraryMediaItem : LibraryItem
    {
        private double _duration; //MediaItem Duartion

        // Precondition:  theCopyrightYear >= 0, LoanPeriod >= 0, volume and theDuartion
>=0
        // Postcondition: The Object has been initialized.
        public LibraryMediaItem(string theTitle, string thePublisher,
        int theCopyrightYear, int LoanPeriod, string theCallNumber, double theDuration) :
base(theTitle, thePublisher, theCopyrightYear, LoanPeriod, theCallNumber)
        {
            Duration = theDuration;
        }


        public double Duration {
            // Precondition:  None
            // Postcondition: Duration returned
            get
            {
                return _duration;
            }

            set
            {
                // Precondition:  Duration >= 0
                // Postcondition: Duration Set
                if (value >= 0)
                {
                    _duration = value;
                } else
                {
                    throw new ArgumentOutOfRangeException("Duration", value, "Duration
must be >= 0");
                }
            }
        }

        //Describes types of Media a LibraryMediaItem derivitve may be.
        public enum MediaType
        {
            DVD,
            BLUERAY,
            VHS,
            CD,
            SACD,
            VINYL
        };

        //Forces inheritors to set medium
       public abstract MediaType Medium {
            get; set;
        }

        // Precondition:  None
        // Postcondition: Return object info in string
        public override string ToString()
        {
            return base.ToString() + NL + $"Duration: {Duration}";
        }

    }
}
```

```
/*     Grading ID: D1499
 *     Program 1A
 *     Due Date: February 25,2017
 *     Course: CIS 200
 *     Course Section: 01
 *     Program:
 *      Show uses of polymorphsim and inhertiance
 *
 */
 //copied from Program V3
// Program 0
// CIS 200-01
// By: Andrew L. Wright (Student's use Grading ID)
// Due: 1/20/2017

// File: LibraryPatron.cs
// This file creates a simple LibraryPatron class capable of tracking
// the patron's name and ID.

// Version 3
// Added validation in set accessors of properties, trims strings
// Used string.IsNullOrWhitespace to test

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;


public class LibraryPatron
{
    private string _patronName; // Name of the patron
    private string _patronID;   // ID of the patron

    // Precondition:  name and id must not be null or empty
    // Postcondition: The patron has been initialized with the specified name
    //                and ID
    public LibraryPatron(string name, string id)
    {
        PatronName = name;
        PatronID = id;
    }

    public string PatronName
    {
        // Precondition:  None
        // Postcondition: The patron's name has been returned
        get
        {
            return _patronName;
        }

        // Precondition:  value must not be null or empty
        // Postcondition: The patron's name has been set to the specified value
        set
        {
            if (!string.IsNullOrWhiteSpace(value)) // IsNullOrWhiteSpace includes tests
for null, empty, or all whitespace
```

```
                _patronName = value.Trim();        // Store trimmed value
            else
                throw new ArgumentOutOfRangeException($"{nameof(PatronName)}", value,
                    $"{nameof(PatronName)} must not be null or empty");
        }
    }

    public string PatronID
    {
        // Precondition:  None
        // Postcondition: The patron's ID has been returned
        get
        {
            return _patronID;
        }

        // Precondition:  value must not be null or empty
        // Postcondition: The patron's ID has been set to the specified value
        set
        {
            if (!string.IsNullOrWhiteSpace(value)) // IsNullOrWhiteSpace includes tests
for null, empty, or all whitespace
                _patronID = value.Trim();          // Store trimmed value
            else
                throw new ArgumentOutOfRangeException($"{nameof(PatronID)}", value,
                    $"{nameof(PatronID)} must not be null or empty");
        }
    }

    // Precondition:  None
    // Postcondition: A string is returned presenting the libary patron's data on
    //                separate lines
    public override string ToString()
    {
        string NL = Environment.NewLine; // NewLine shortcut

        return $"Name: {PatronName}{NL}ID: {PatronID}";
    }

}
```

```
/*     Grading ID: D1499
```

```
 *     Program 1A
 *     Due Date: February 25,2017
 *     Course: CIS 200
 *     Course Section: 01
 *     Program:
 *      Show uses of polymorphsim and inhertiance
 *
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Program_1A
{
    class LibraryBook : LibraryItem
    {
        private string _author; //the book's author

        // Precondition:  theCopyrightYear >= 0
        // Postcondition: The library LibraryBook has been initialized.
        public LibraryBook(string theTitle, string theAuthor, string thePublisher,
        int theCopyrightYear, int LoanPeriod, string theCallNumber) : base(theTitle,
thePublisher, theCopyrightYear, LoanPeriod, theCallNumber)
        {
            Author = theAuthor;
        }



        public string Author
        {
            // Precondition:  None
            // Postcondition: The author has been returned
            get
            {
                return _author;
            }

            // Precondition:  None
            // Postcondition: The author has been set to the specified value
            set
            {
                // Since empty author is OK, just change null to empty string
                _author = (value == null ? string.Empty : value.Trim());
            }


        }
        private const decimal _fee = .25m; //late fee

        // Precondition:  daysLate >= 0
        // Postcondition: returned late fee in $'s
        public override decimal CalcLateFee(int daysLate)
        {

            if (daysLate >= 0)
            {
                return daysLate * _fee;
            } else
            {
                throw new ArgumentOutOfRangeException("CalcLateFee",daysLate, "Days late 
must be >= 0");
            }


        }

        // Precondition: None
        // Postcondition: returned string of info
        public override string ToString()
        {

            return base.ToString() + NL + $"Author: {Author}";
        }
    }
}
```

```
/*     Grading ID: D1499
 *     Program 1A
 *     Due Date: February 25,2017
 *     Course: CIS 200
 *     Course Section: 01
 *     Program:
 *      Show uses of polymorphsim and inhertiance
 *
 */
 using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Program_1A
{
    abstract class LibraryPeriodical : LibraryItem
    {
        private int _volume; //This Item's volume #
        private int _number; //This iem's number


        // Precondition:  theCopyrightYear >= 0, LoanPeriod >= 0, volume and number >=1
        // Postcondition: The Object has been initialized.
        public LibraryPeriodical(string theTitle, string thePublisher,
         int theCopyrightYear, int LoanPeriod, string theCallNumber, int theVolume, int
theNumber) : base(theTitle, thePublisher, theCopyrightYear, LoanPeriod, theCallNumber)
        {
            Volume = theVolume;
            Number = theNumber;
        }

        public int Volume
        {
            // Precondition:  None
            // Postcondition: Volume returned
            get
            {
                return _volume;
            }
            // Precondition: input >= 1
            // Postcondition: Volume has been set
            set
            {
                if (value >= 1)
                {
                    _volume = value;
                }
                else
                {
                    throw new ArgumentOutOfRangeException("Volume", value, "Volume must
be >= 1");
                }
            }
        }

        public int Number
        {
            // Precondition:  Number has been returend
            get
            {
                return _number;
            }

            // Precondition: input >= 1
            // Postcondition: Number has been set
            set
            {
                if (value >= 1)
                {
                    _number = value;
                }
                else
                {
                    throw new ArgumentOutOfRangeException("Number", value, "Number must
be >= 1");
                }
            }
        }

        // Precondition:  None
        // Postcondition: Object info returned via string
        public override string ToString()
        {
            return base.ToString() + NL + $"Volume: {Volume} {NL} Number: {Number} {NL}";
        }


    }
}
```

```
/*      Grading ID: D1499
 *      Program 1A
 *      Due Date: February 25,2017
 *      Course: CIS 200
 *      Course Section: 01
 *      Program:
 *       Show uses of polymorphsim and inhertiance
 *
 */
 using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Program_1A
{
    class LibraryJournal : LibraryPeriodical
    {
        private string _editor; //the item's editor
        private string _discipline; //item's discipline
        // Precondition:  theCopyrightYear >= 0, LoanPeriod >= 0, volume and number >=0
        // Postcondition: The Object has been initialized.
        public LibraryJournal(string theTitle, string thePublisher,
    int theCopyrightYear, int LoanPeriod, string theCallNumber, int theVolume, int
theNumber, string theDiscipline, string theEditor) : base(theTitle, thePublisher,
theCopyrightYear, LoanPeriod, theCallNumber, theVolume, theNumber)
        {
            Editor = theEditor;
            Discipline = theDiscipline;
        }


        public string Editor
        {
            // Precondition: None
            // Postcondition:Returned Editor
            get
            {
                return _editor;
            }
            // Precondition:  Not null input
            // Postcondition: Editor has been set
            set
            {
                // Since empty author is OK, just change null to empty string
                if (string.IsNullOrWhiteSpace(value.Trim()))
                {
                    throw new ArgumentOutOfRangeException("Editor", value, "Editor can
not be null or whitespace.");
                }
                else
                {
                    _editor = value;
                }

            }
        }
        public string Discipline
        {
            // Precondition:  None
            // Postcondition: Discipline returned
            get
            {
                return _discipline;
            }
            // Precondition:  Input not blank/null
            // Postcondition: Discipline has been set
            set
            {
                // Since empty author is OK, just change null to empty string
                if (string.IsNullOrWhiteSpace(value.Trim()))
                {
                    throw new ArgumentOutOfRangeException("Discipline", value,
"Discipline can not be null or whitespace.");
                }
                else
                {
                    _discipline = value;
                }

            }
        }


        private const decimal fee = .75m; //SameFee all around

        // Precondition:  lateDays >= 0
        // Postcondition: Returned late fee in decimal
        public override decimal CalcLateFee(int lateDays)
        {

            if (lateDays >= 0)
            {
              return (lateDays * fee);


            }

            else
            {
                throw new ArgumentOutOfRangeException("CalcLateFee", lateDays, "Days late
must be >= 0");
            }
        }

        // Precondition:  None
        // Postcondition: Returned Object information
        public override string ToString()
        {
            return base.ToString() + NL + $"Discipline: {Discipline} {NL} Editor:
{Editor} {NL}";
        }
```

```
    }
}


/*     Grading ID: D1499
 *     Program 1A
 *     Due Date: February 25,2017
 *     Course: CIS 200
 *     Course Section: 01
 *     Program:
 *      Show uses of polymorphsim and inhertiance
 *
 */
 using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Program_1A
{
    class LibraryMagazine : LibraryPeriodical
    {
        // Precondition:  theCopyrightYear >= 0, LoanPeriod >= 0, volume and number >=0
        // Postcondition: The Object has been initialized.
        public LibraryMagazine(string theTitle, string thePublisher,
       int theCopyrightYear, int LoanPeriod, string theCallNumber, int theVolume, int
theNumber)
            : base(theTitle, thePublisher, theCopyrightYear, LoanPeriod, theCallNumber,
theVolume, theNumber)
        {

        }


        private const decimal fee = .25m; //SameFee all around
        private const decimal limit = 20m; //Fee Maximum for all medium
         // Precondition: lateDays >= 0
        // Postcondition: returned decimal of late fee calucation
        public override decimal CalcLateFee(int lateDays)
        {

            if (lateDays >= 0)
            {
                if ((lateDays * fee) >= limit)
                {
                    return limit;
                }
                else
                {
                    return (lateDays * fee);
                }

            }

            else
            {
                throw new ArgumentOutOfRangeException("CalcLateFee", lateDays, "Days late 
must be >= 0");
            }
        }

        // Precondition: None
        // Postcondition: Returned string of object info
        public override string ToString()
        {
            return base.ToString();
        }

    }
}
```

```
/*     Grading ID: D1499
 *     Program 1A
 *     Due Date: February 25,2017
 *     Course: CIS 200
 *     Course Section: 01
 *     Program:
 *      Show uses of polymorphsim and inhertiance
 *
 */
 using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Program_1A
{
    class LibraryMovie : LibraryMediaItem
    {
        private MediaType _medium; //This Movie's Medium
        private string _director; //This movie's director
        private MPAARatings _ratings; //This movie's ratings

        //Defines MPAA Ratings for LibraryMovie
       public  enum MPAARatings
        {
            G, PG, PG13, R, NC17, U
        };

        // Precondition:  theCopyrightYear >= 0, LoanPeriod >= 0, duration >=0, not null
director, appropiate Medium, and valid MPPA Ratings
        // Postcondition: The Object has been initialized.
        public LibraryMovie(string theTitle, string thePublisher, int theCopyrightYear,
int LoanPeriod, string theCallNumber, double theDuration, string theDirector, MediaType
theMedium, MPAARatings theRatings) : base(theTitle, thePublisher,
        theCopyrightYear, LoanPeriod, theCallNumber, theDuration)
        {
            Medium = theMedium;
            Ratings = theRatings;
            Director = theDirector;
        }


        override public MediaType Medium
            {

            // Precondition:  None
            // Postcondition: Medium returned
            get
            {
                return _medium;
            }
            // Precondition:  Valid Medium for a movie
            // Postcondition: Medium has been set
            set
            {
                if (value != MediaType.BLUERAY && value != MediaType.DVD && value !=
MediaType.VHS)
```

```
                {
                    throw new ArgumentOutOfRangeException("Medium", value, "Medium must
be Blueray, DVD, or VHS");
                } else
                {
                    _medium = value;
                }
            }
        }


        public string Director
        {

            // Precondition:  None
            // Postcondition: Director returned
            get
            {
                return _director;
            }

            // Precondition:  Input not blank
            // Postcondition: Director Set
            set
            {
                // Since empty author is OK, just change null to empty string
                if (string.IsNullOrWhiteSpace(value.Trim())) {
                    throw new ArgumentOutOfRangeException("Director", value, "Director
can not be null or whitespace.");
                } else
                {
                    _director = value;
                }

            }
        }

        // Precondition:  Valid ratings
        // Postcondition: Ratings has been set

        public MPAARatings Ratings
        {

            // Precondition:  None
            // Postcondition: ratings returned
            get { return _ratings; }

            // Precondition:  Valid Input
            // Postcondition: Ratings Set
            set
            {

                _ratings = value;
            }
        }


        private const decimal lowerFee = 1m; //Lower fee for DVD/VHS
        private const decimal higherFee = 1.5m; //Higher fee for BlueRay
        private const decimal limit = 25m; //Fee Maximum for all medium
        private decimal tempCalc; //Tempoary holder for Fee Calcuation
         // Precondition:  lateDays >= 0
        // Postcondition: Return decimal of late fee
        public override decimal CalcLateFee(int lateDays)
        {

            if (lateDays >= 0)
            {

                switch (Medium)
                {
                    case MediaType.DVD:
                    case MediaType.VHS:
                        tempCalc = lowerFee * lateDays;
                        break;

                    case MediaType.BLUERAY:
                        tempCalc = higherFee * lateDays;
                        break;
                  }
                if (tempCalc >= limit)
                {
                    return limit;
                } else
                {
                    return tempCalc;
                }

            } else
            {
                throw new ArgumentOutOfRangeException("CalcLateFee", lateDays, "Days late
must be >= 0");
            }
        }

        // Precondition: None
        // Postcondition: Returned string of object info
        public override string ToString()
        {
            return base.ToString() + NL + $"Director: {Director} {NL} Medium: {Medium}
{NL} MPAARatings: {Ratings}";
        }

    }
}
```

```
/*     Grading ID: D1499
```

```
 *     Program 1A
 *     Due Date: February 25,2017
 *     Course: CIS 200
 *     Course Section: 01
 *     Program:
 *      Show uses of polymorphsim and inhertiance
 *
 */
 using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Program_1A
{

    class LibraryMusic : LibraryMediaItem
    {

        private string _artist; //the songs artist
        private MediaType _medium; //The MediaType
        private int _TrackCount; //The trackcount of this music
         // Precondition:  theCopyrightYear >= 0, LoanPeriod >= 0, artist not null,
Medium approiate, trackcount >=1
        // Postcondition: The Object has been initialized.
        public  LibraryMusic(string theTitle, string thePublisher, int theCopyrightYear,
int LoanPeriod, string theCallNumber, double theDuration, string theArtist, MediaType
theMedium, int theNumberOfTracks) : base(theTitle, thePublisher,
        theCopyrightYear, LoanPeriod, theCallNumber, theDuration)
        {
            Artist = theArtist;
            Medium = theMedium;
            TrackCount = theNumberOfTracks;
        }


        override public MediaType Medium
        {
            // Precondition:  None
            // Postcondition: Return Medium
            get
            {
                return _medium;
            }
            // Precondition: Approiate medium
            // Postcondition: Medium has been set
            set
            {
                if (value != MediaType.CD && value != MediaType.SACD && value !=
MediaType.VINYL)
                {
                    throw new ArgumentOutOfRangeException("Medium", value, "Medium must
be CD, SACD, VINYL");
                }
                else
                {
                    _medium = value;
                }
            }
        }


        public string Artist
        {
            // Precondition:  None
            // Postcondition: Returned Artist
            get
            {
                return _artist;
            }
            // Precondition: Not null artist
            // Postcondition: artist has been set
            set
            {
                // Since empty author is OK, just change null to empty string
                if (string.IsNullOrWhiteSpace(value.Trim()))
                {
                    throw new ArgumentOutOfRangeException("Director", value, "Director
can not be null or whitespace.");
                }
                else
                {
                    _artist = value;
                }

            }
        }

        public int TrackCount
        {
            // Precondition:  None
            // Postcondition: Retured track count
            get
            {
                return _TrackCount;
            }
            // Precondition:  trackcount >= 1
            // Postcondition: track count is set
            set
            {
                if (value >= 1)
                {
                    _TrackCount = value;
                }
                else
                {
                    throw new ArgumentOutOfRangeException("TrackCount", value, "Count
must be >= 1");
                }
            }
        }


        private const decimal fee = .5m; //SameFee all around
        private const decimal limit = 20m; //Fee Maximum for all medium
```

```
        // Precondition:  late days >= 0
        // Postcondition: Late fee returned in decimal
        public override decimal CalcLateFee(int lateDays)
        {

            if (lateDays >= 0)
            {
                if ((lateDays * fee) >= limit)
                {
                    return limit;
                }
                else
                {
                    return (lateDays * fee);
                }

            }

            else
            {
                throw new ArgumentOutOfRangeException("CalcLateFee", lateDays, "Days late
must be >= 0");
            }
        }

        // Precondition:  None
        // Postcondition: Returned String of object info
        public override string ToString()
        {
            return base.ToString() + NL + $"Artist: {Artist} {NL} Medium: {Medium} {NL}
Number of Tracks: {TrackCount}";
        }






    }


}
```

```
/*     Grading ID: D1499
 *     Program 1A
 *     Due Date: February 25,2017
 *     Course: CIS 200
 *     Course Section: 01
 *     Program:
 *      Show uses of polymorphsim and inhertiance
 *
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Program_1A
{
    class Program
    {
        static void Main(string[] args)
        {


            List<LibraryItem> ItemList = new List<LibraryItem>();

            LibraryBook b1 = new LibraryBook("Test", "Frank", "Frank Interprises", 2016,
2, "r");
            LibraryJournal J1 = new LibraryJournal("Journal", "Franklin", 2016, 2, "333",
1, 1, "idk", "me");
            LibraryMagazine m1 = new LibraryMagazine("Mag", "Frankie", 2016, 2, "444", 1,
1);
            LibraryMovie mo1 = new LibraryMovie("The Movie", "Frank1", 2016, 2, "Mo", 30,
"Frank", LibraryMediaItem.MediaType.BLUERAY, LibraryMovie.MPAARatings.G);
            LibraryMusic mu1 = new LibraryMusic("Song", "Frank2", 2016, 2, "Song", 5,
"F", LibraryMediaItem.MediaType.CD, 1);
            LibraryPatron write = new LibraryPatron("Wright", "Wright1");



            mu1.CheckOut(write);


            ItemList.Add(b1);
            ItemList.Add(J1);
            ItemList.Add(m1);
            ItemList.Add(mo1);
            ItemList.Add(mu1);

            foreach(LibraryItem a in ItemList)
            {
                Console.WriteLine(a);
            }




        }
        }
}
```