Project 6: Media Library

Using Inheritance and a Data Structure to Implement a Program

Overview

For this project, we will explore how to work with inheritance in Java. Inheritance is an extremely useful tool when you have multiple types of objects which are similar, as it allows you to treat them in a more general way. It also helps us reuse code, so we don't have to type out the same things repeatedly. For more information on inheritance in Java, refer to zybooks chapter 10.

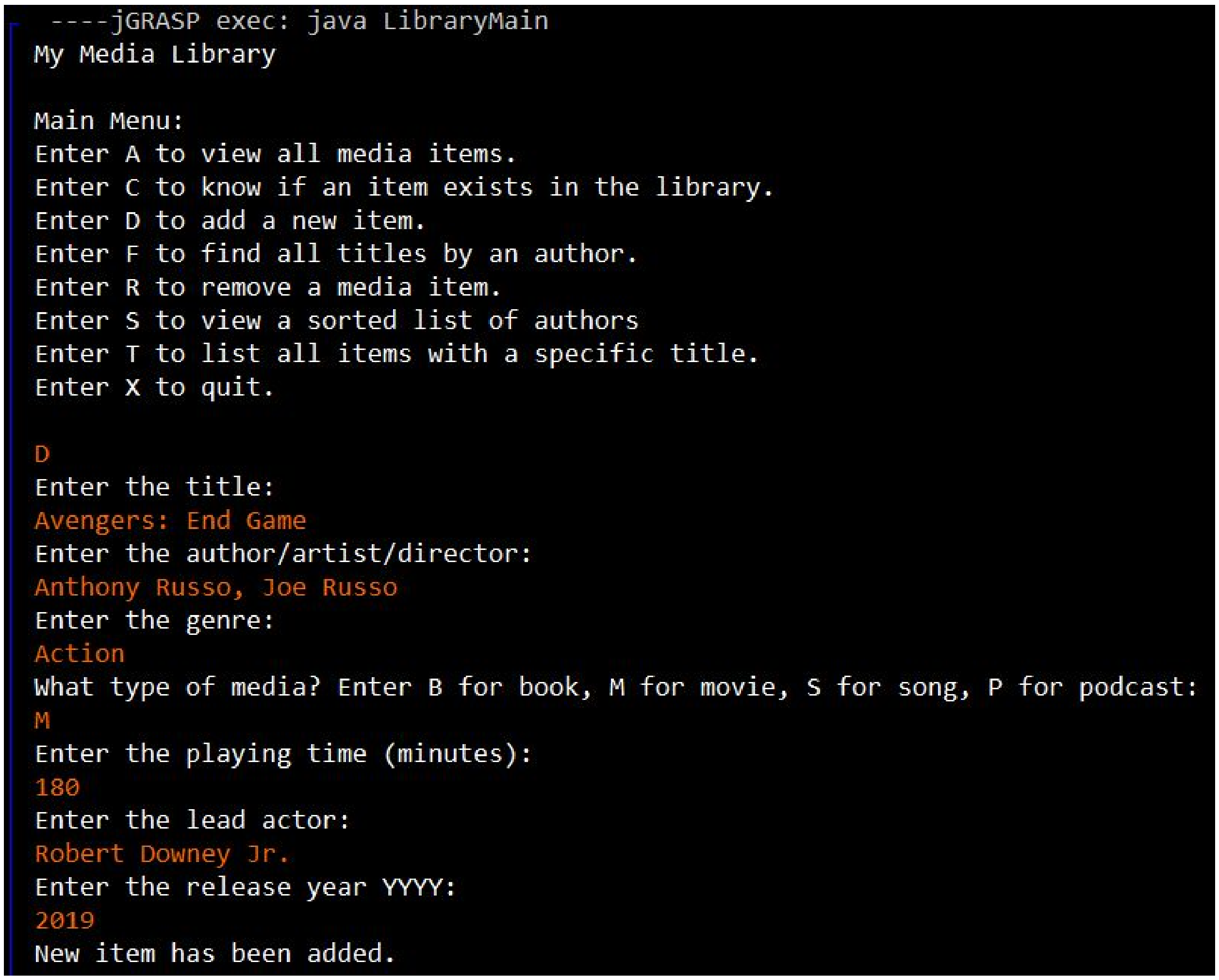
Learning Goals:

- Work with a larger multi-file Java project.
- Work with inheritance to handle similar object types.
- Use a data structure such as an array or an ArrayList.
- Write private helper methods (not required, but suggested).
- Practice using the JGrasp debugger.
- Work with methods and objects.
- Run JUnit tests.
- Use required style guide.

Project Specification: Your Tasks:

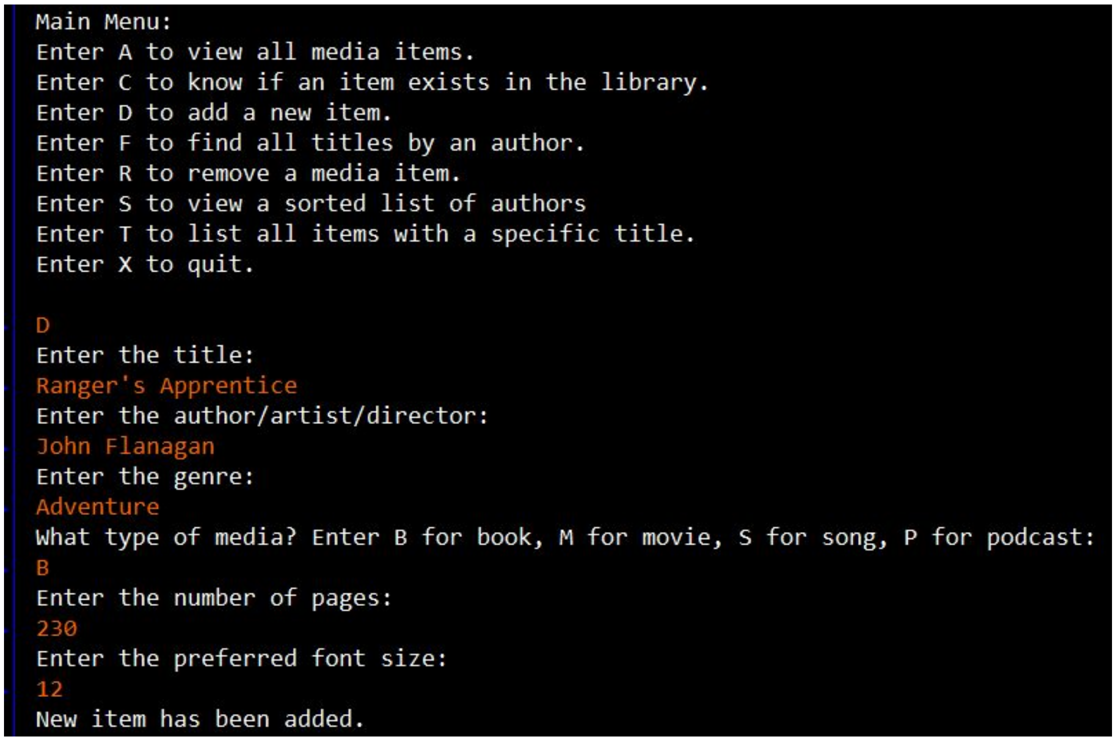
This time, we will be making a tool to help users organize their media. Media comes in many different types, and as our world changes we need to have ways of keeping track of more than just one type of thing. Our tool will help users find and organize all of their different types of media. In the starter code are instructions that detail more specifically what each part of the project is supposed to do, so make sure to read them. ***Do not delete the starter code!***

Sample output to enter the details of a movie:



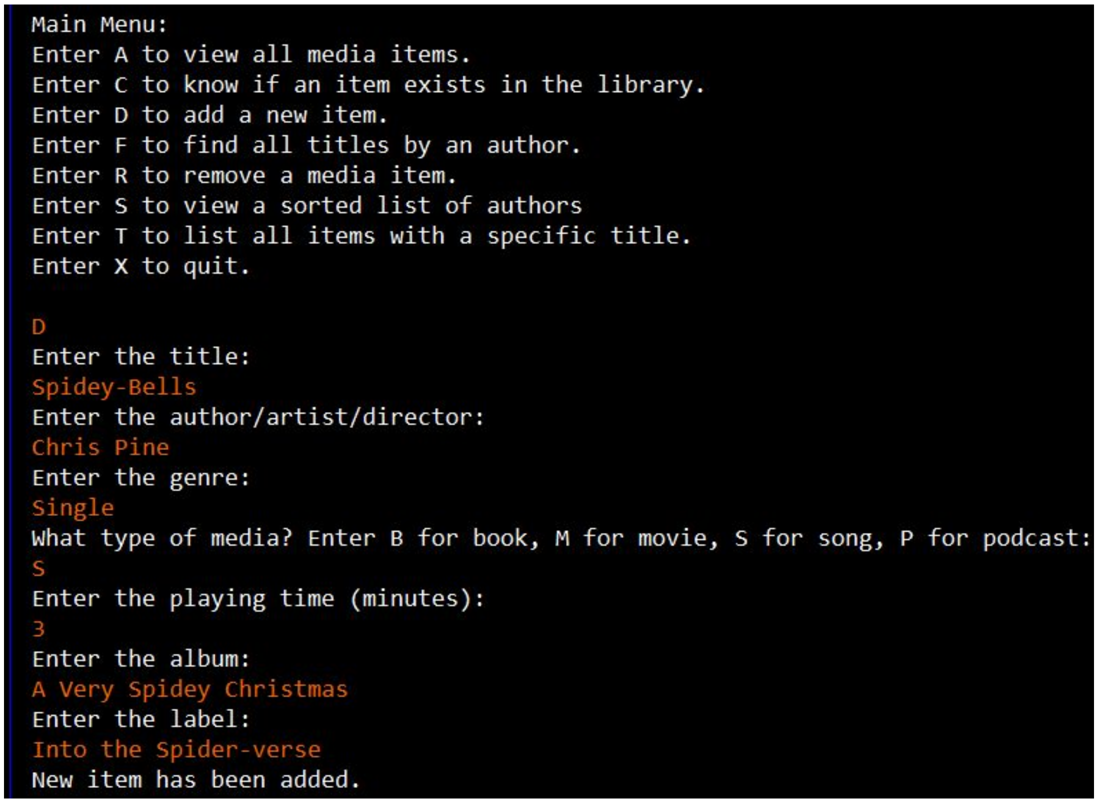
```
 ----jGRASP exec: java LibraryMain
My Media Library

Main Menu:
Enter A to view all media items.
Enter C to know if an item exists in the library.
Enter D to add a new item.
Enter F to find all titles by an author.
Enter R to remove a media item.
Enter S to view a sorted list of authors
Enter T to list all items with a specific title.
Enter X to quit.

D
Enter the title:
Avengers: End Game
Enter the author/artist/director:
Anthony Russo, Joe Russo
Enter the genre:
Action
What type of media? Enter B for book, M for movie, S for song, P for podcast:
M
Enter the playing time (minutes):
180
Enter the lead actor:
Robert Downey Jr.
Enter the release year YYYY:
2019
New item has been added.
```

Another sample output to enter the details of a book.



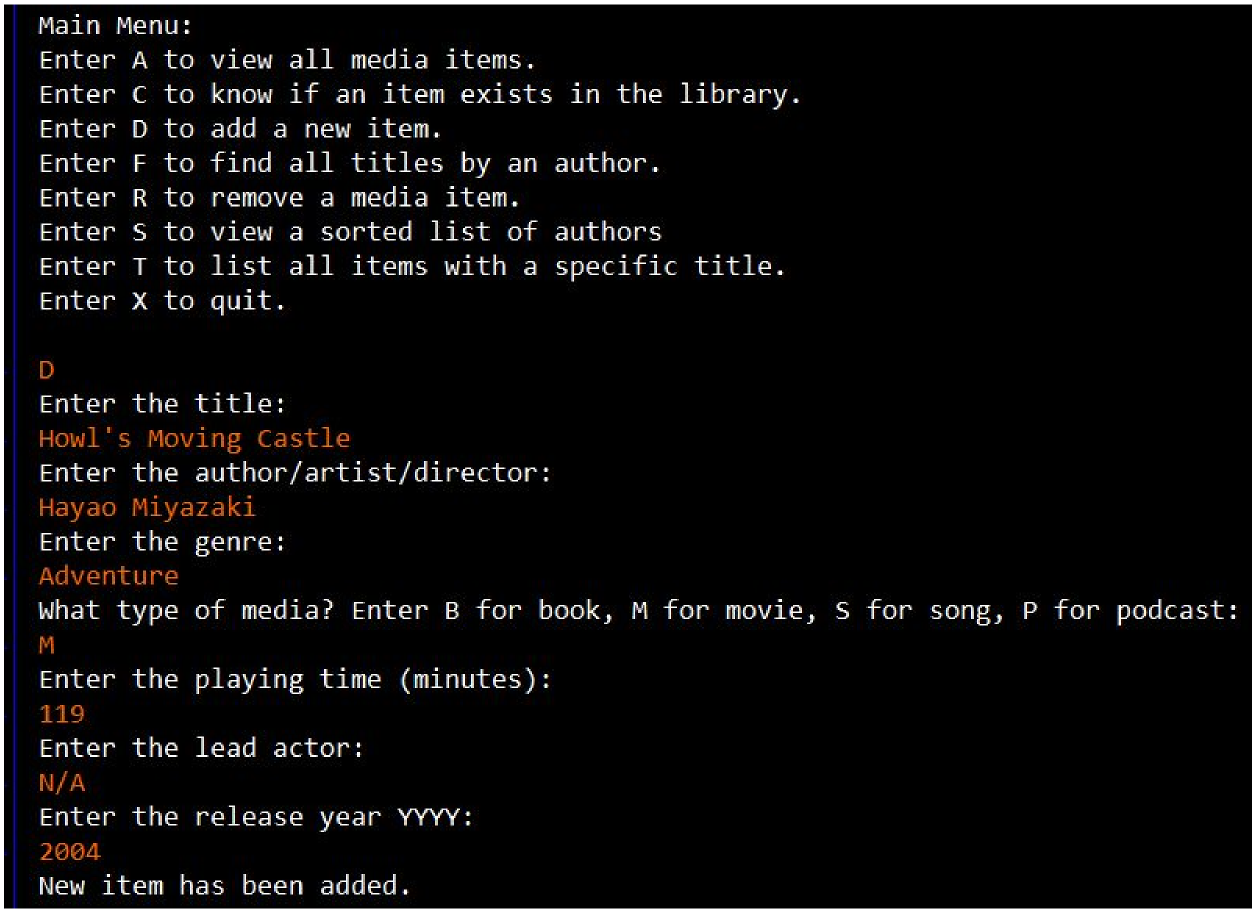
```
Main Menu:
Enter A to view all media items.
Enter C to know if an item exists in the library.
Enter D to add a new item.
Enter F to find all titles by an author.
Enter R to remove a media item.
Enter S to view a sorted list of authors
Enter T to list all items with a specific title.
Enter X to quit.

D
Enter the title:
Ranger's Apprentice
Enter the author/artist/director:
John Flanagan
Enter the genre:
Adventure
What type of media? Enter B for book, M for movie, S for song, P for podcast:
B
Enter the number of pages:
230
Enter the preferred font size:
12
New item has been added.
```

Another sample output to enter the details of a song.

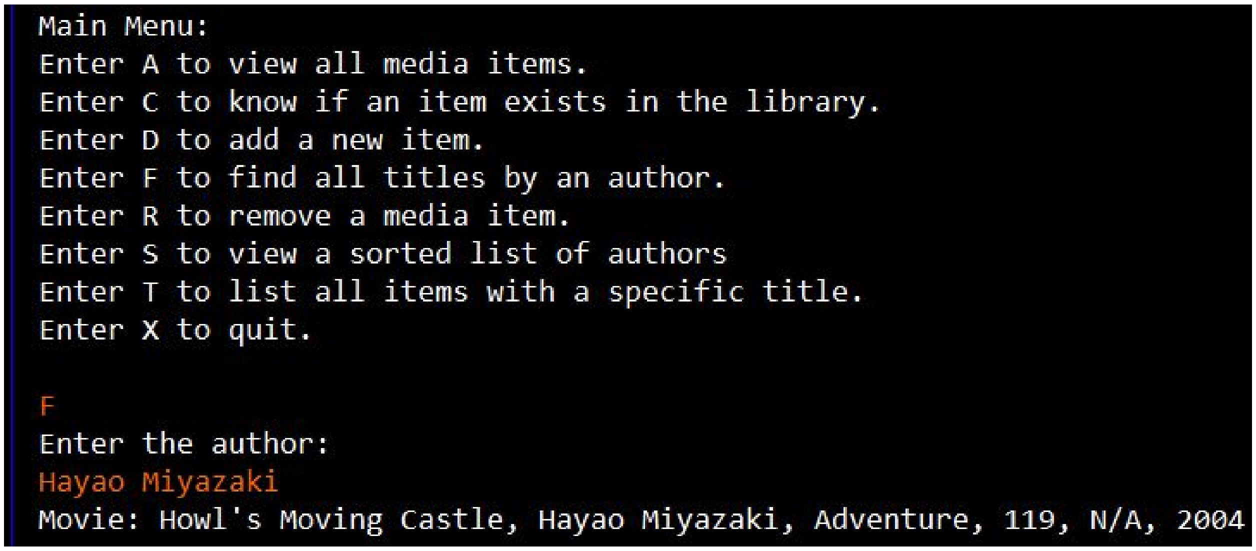


```
Main Menu:
Enter A to view all media items.
Enter C to know if an item exists in the library.
Enter D to add a new item.
Enter F to find all titles by an author.
Enter R to remove a media item.
Enter S to view a sorted list of authors
Enter T to list all items with a specific title.
Enter X to quit.

D
Enter the title:
Spidey-Bells
Enter the author/artist/director:
Chris Pine
Enter the genre:
Single
What type of media? Enter B for book, M for movie, S for song, P for podcast:
S
Enter the playing time (minutes):
3
Enter the album:
A Very Spidey Christmas
Enter the label:
Into the Spider-verse
New item has been added.
```

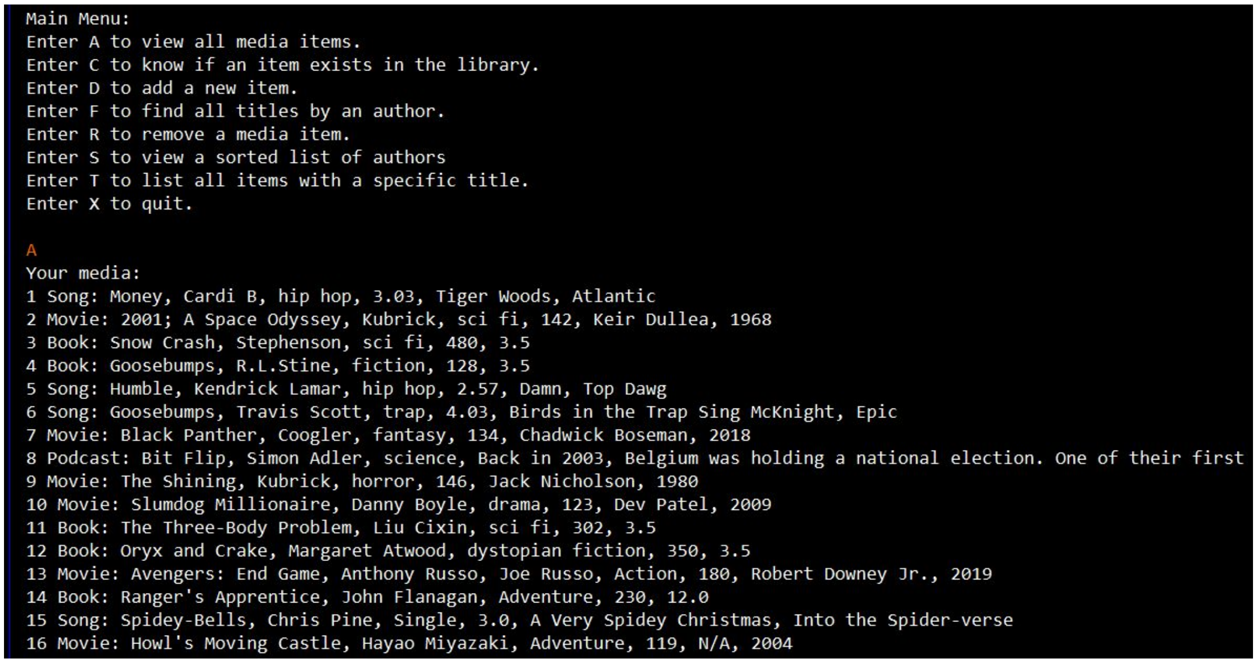
Another sample output to enter the details of another movie.


```
Main Menu:
Enter A to view all media items.
Enter C to know if an item exists in the library.
Enter D to add a new item.
Enter F to find all titles by an author.
Enter R to remove a media item.
Enter S to view a sorted list of authors
Enter T to list all items with a specific title.
Enter X to quit.

D
Enter the title:
Howl's Moving Castle
Enter the author/artist/director:
Hayao Miyazaki
Enter the genre:
Adventure
What type of media? Enter B for book, M for movie, S for song, P for podcast:
M
Enter the playing time (minutes):
119
Enter the lead actor:
N/A
Enter the release year YYYY:
2004
New item has been added.
```

Sample output to find media items.


```
Main Menu:
Enter A to view all media items.
Enter C to know if an item exists in the library.
Enter D to add a new item.
Enter F to find all titles by an author.
Enter R to remove a media item.
Enter S to view a sorted list of authors
Enter T to list all items with a specific title.
Enter X to quit.

F
Enter the author:
Hayao Miyazaki
Movie: Howl's Moving Castle, Hayao Miyazaki, Adventure, 119, N/A, 2004
```

Sample output to view media items.



```
Main Menu:
Enter A to view all media items.
Enter C to know if an item exists in the library.
Enter D to add a new item.
Enter F to find all titles by an author.
Enter R to remove a media item.
Enter S to view a sorted list of authors
Enter T to list all items with a specific title.
Enter X to quit.

A
Your media:
1 Song: Money, Cardi B, hip hop, 3.03, Tiger Woods, Atlantic
2 Movie: 2001; A Space Odyssey, Kubrick, sci fi, 142, Keir Dullea, 1968
3 Book: Snow Crash, Stephenson, sci fi, 480, 3.5
4 Book: Goosebumps, R.L.Stine, fiction, 128, 3.5
5 Song: Humble, Kendrick Lamar, hip hop, 2.57, Damn, Top Dawg
6 Song: Goosebumps, Travis Scott, trap, 4.03, Birds in the Trap Sing McKnight, Epic
7 Movie: Black Panther, Coogler, fantasy, 134, Chadwick Boseman, 2018
8 Podcast: Bit Flip, Simon Adler, science, Back in 2003, Belgium was holding a national election. One of their first
9 Movie: The Shining, Kubrick, horror, 146, Jack Nicholson, 1980
10 Movie: Slumdog Millionaire, Danny Boyle, drama, 123, Dev Patel, 2009
11 Book: The Three-Body Problem, Liu Cixin, sci fi, 302, 3.5
12 Book: Oryx and Crake, Margaret Atwood, dystopian fiction, 350, 3.5
13 Movie: Avengers: End Game, Anthony Russo, Joe Russo, Action, 180, Robert Downey Jr., 2019
14 Book: Ranger's Apprentice, John Flanagan, Adventure, 230, 12.0
15 Song: Spidey-Bells, Chris Pine, Single, 3.0, A Very Spidey Christmas, Into the Spider-verse
16 Movie: Howl's Moving Castle, Hayao Miyazaki, Adventure, 119, N/A, 2004
```

There are more features in the menu that you can check out.

These are the program files with complete code that are provided to you.

1. **`LibraryMain.java`**
2. **`MediaItem.java`**
3. **`MediaListTest.java`**

The following is a brief description of the classes with complete code that are ***given to you***:

1. **`LibraryMain.java`**
   This class contains the main method. We include a method that creates several **`MediaItems`** if you want to do some testing. Adding more media items will **not** affect the autograder, so you are more than welcome to do so. The only modification that you can make to this file is to uncomment line 19, which is a call to the **`createListForTesting`** method. This method populates the **`mediaList`** with data. This will allow you to run the app from the main method for testing. However, you must remember to test your code on an empty **`mediaList`** as well!
2. **`MediaItem.java`**
   This class is the superclass of all other media types. It stores the title, author and genre of the media item. **Do not make any modifications to this file.**
3. **`MediaListTest.java`**
   This is the test file, which acts as a testing framework for your code. You may add tests if you like. Do **not** modify the tests we give you since these are the tests the autograder will run (and maybe more).

The classes you will be developing are listed below. All of the tasks you must complete are designated by "`TODO`:" in these files.
We recommend developing the classes (files) in the following order:

1. **`Book.java`**
2. **`Song.java`**
3. **`Movie.java`**
4. **`Podcast.java`**
5. **`MediaList.java`**

This project will not compile when you first get it. This is due to the missing constructor calls in the subclasses. That is why we recommend you develop the subclasses first and compile them.

The following is a brief description of the different classes ***you need to implement:***

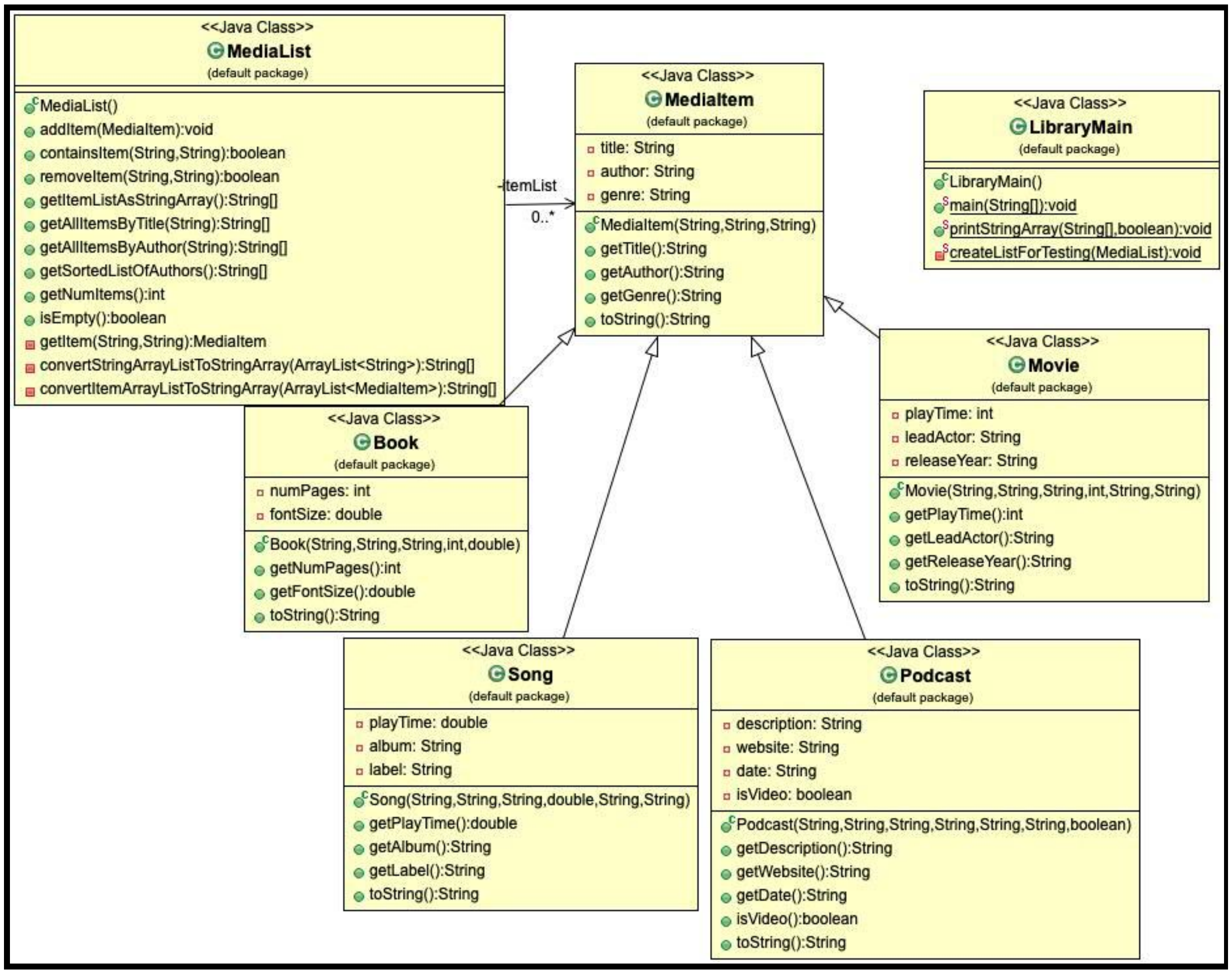
1. **`Book.java, Song.java`**, **`Movie.java`**, **`Podcast.java`**
   These classes represent various media types in the library. They extend the **`MediaItem`** class. They are in different stages of development.

2. **`MediaList.java`**
   This class encapsulates the data and functionality that allows the program to add and remove from a list of **`MediaItems`**. It also provides methods that allow users to perform some search features and other functions that access the data.

   Developing the methods in the **`MediaList`** class will provide you with experience using a data structure- an array or **`ArrayList`**. Programming mostly involves working with data. The operations on the data structure you will implement are a good preparation for the next course after this one, which is the study and use of various kinds of "Data Structures".
   Some other notes about developing the code in **`MediaList.java`** :
   1. You may use an array or an ArrayList to implement the data structure that holds the **`MediaItem`** objects. We suggest using the **`ArrayList`** as the code is simpler; however, the choice is yours.
   2. You must declare at least one instance variable that serves to hold the **`MediaItems`** in the list. You may also declare other instance variables you wish. Remember that if a variable is used in only one method, it should be declared in that method and not as an instance variable.
   3. All instance variables should be initialized in the constructor. Does the constructor take any parameters? To find the parameters, look in **`LibraryMain`** where it is called.
   4. You may define any private methods you wish to be called from other methods. This is not required, but strongly suggested as they are helpful in implementing some of the public methods. This is good coding practice.
   5. All **`String`** comparisons should be case-insensitive.

6. Several of the public methods in **`MediaList`** return an array. If you use an **`ArrayList`**, this will require you to convert from **`ArrayList`** to **`array`**.

Here is an overview **UML diagram of the completed project.**



Export and Submit

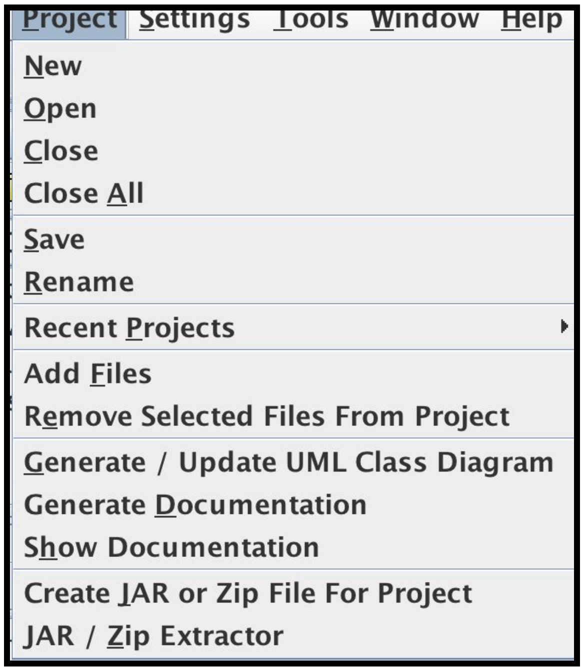
Step 1: Export the File

When you have completed this project, you should export a file containing the entire Java project. There are two ways to do this.

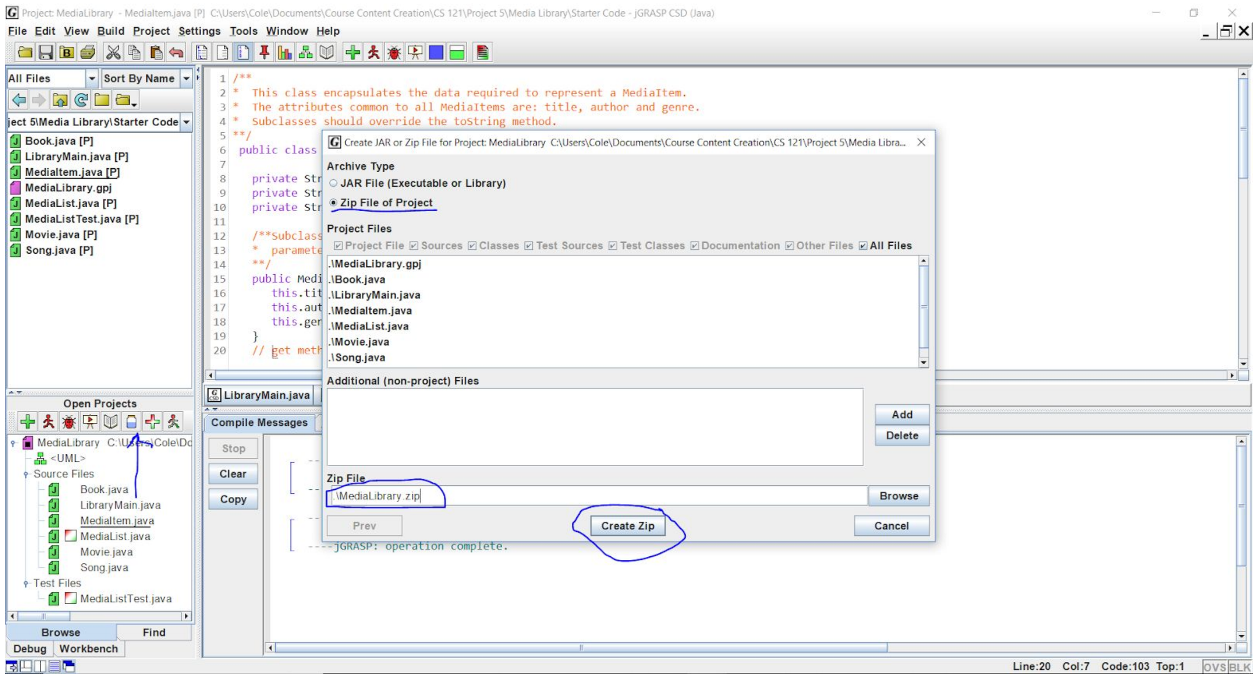
1- Click on the "**`jar`**" icon and the **`.zip`** file is created. See below for a visual guide.

OR:

2- Click on the *Project* menu in the package explorer. Then choose "*Project -> Create JAR or Zip file for Project*" from the menu.



Check it is the correct project name (**`MediaLibrary.zip`**) in the "*Zip file*" box (as in the figure below). Click on "*Create Zip*".


Project: MediaLibrary - MediaItem.java [P] C:\Users\Cole\Documents\Course Content Creation\CS 121\Project 5\Media Library\Starter Code - jGRASP CSD (Java)

```
/**
*  This class encapsulates the data required to represent a MediaItem.
*  The attributes common to all MediaItems are: title, author and genre.
*  Subclasses should override the toString method.
**/
public class
```

Create JAR or Zip File for Project: MediaLibrary C:\Users\Cole\Documents\Course Content Creation\CS 121\Project 5\Media Libra...

Archive Type
- JAR File (Executable or Library)
- [✓ Zip File of Project]

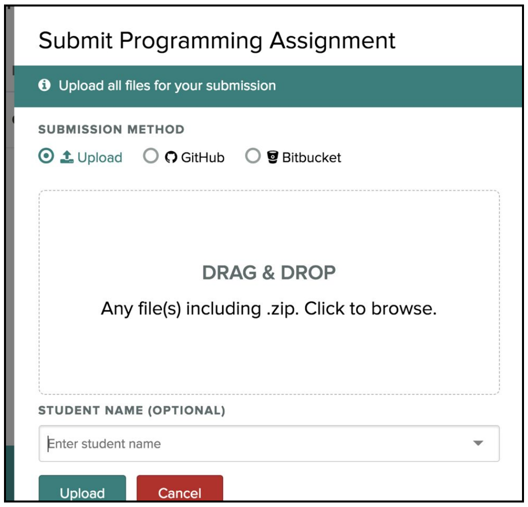
Project Files: ☑ Project File ☑ Sources ☑ Classes ☑ Test Sources ☑ Test Classes ☑ Documentation ☑ Other Files ☑ All Files

.\MediaLibrary.gpj
.\Book.java
.\LibraryMain.java
.\MediaItem.java
.\MediaList.java
.\Movie.java
.\Song.java

Zip File: .\MediaLibrary.zip

----jGRASP: operation complete.

Step 2: Submit the file in Gradescope
Now log into Gradescope, select the assignment, and submit the zip file for grading.


Submit Programming Assignment

Upload all files for your submission

SUBMISSION METHOD

[✓ Upload] GitHub Bitbucket

DRAG & DROP

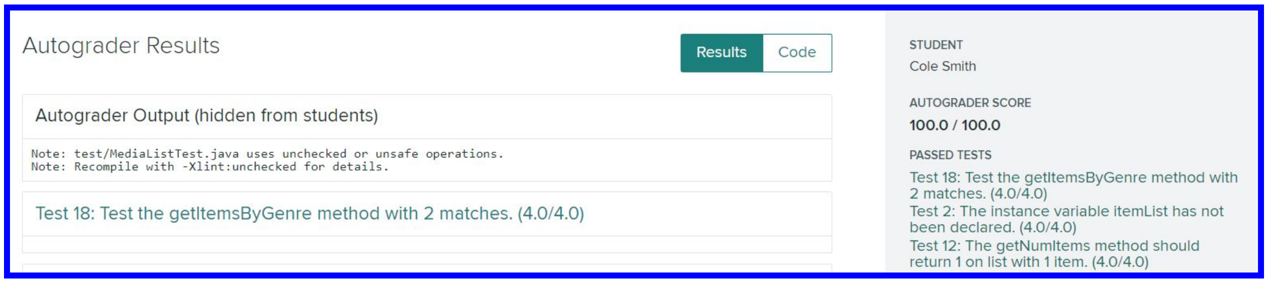
Any file(s) including .zip. Click to browse.

STUDENT NAME (OPTIONAL)

Enter student name

Compilation errors will be provided by the autograder for this assignment. You must read assignments carefully -- if your submission is not passing a test, it is almost certainly because your submission doesn't match the requirements of the assignment.

If all tests pass your screen shows 40/40 in Gradescope:
CHECK BELOW & REPLACE AS NECESSARY


Autograder Results

Autograder Output (hidden from students)

```
Note: test/MediaListTest.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

Test 18: Test the getItemsByGenre method with 2 matches. (4.0/4.0)

STUDENT
Cole Smith

AUTOGRADER SCORE
100.0 / 100.0

PASSED TESTS
Test 18: Test the getItemsByGenre method with 2 matches. (4.0/4.0)
Test 2: The instance variable itemList has not been declared. (4.0/4.0)
Test 12: The getNumItems method should return 1 on list with 1 item. (4.0/4.0)

Remember, you can re-submit the assignment as many times as you want, until the deadline. If it turns out you missed something and your code doesn't pass 100% of the tests, you can keep working until it does. Attend office hours for help or post your questions in Piazza.

Style Guide:

Apart from the score of the autograder, your project will be manually graded for following these style guidelines (10 points):

1. All unnecessary lines of code, including TODO lines have been removed.
2. Proper indenting must be used.
3. Optimal use of blank lines and spaces for operators.
4. Use of camelCase for variable/parameter and method names.
5. Instance variables declared early (not within code), and initialized where appropriate.
6. Descriptive variable and method names.

See zyBooks sections 2.8 and 2.14 for the style guidelines we require you to follow.

Tips

- Use your debugger! It is very useful in finding exactly where your program stops doing what you want it to do.
- Start early! Projects are starting to get a little longer, so if you get stuck you need to make sure you have enough time to seek help.
- Seek help when you get stuck. We have office hours and forums specifically for you to ask questions when you need assistance. Use public posts as much as possible so we don't have to answer the same question multiple times, only use a private post if you need us to see your code or have questions specific to you.
- Look at examples in both your textbook and previous labs. While you shouldn't be copying code, looking at examples can help clear up semantic or simple questions should they come up.
- Submit to gradescope at least once, even if you aren't completely done. There is a huge difference between a 50% and a 0%. That said, aim for 100%.

See this link for FAQs for projects.