**Your First Class**

**Introduction**

When you program in an object-oriented language, you are programming with objects. An object describes a set of data in a program, and it can represent anything—a text message, a list of contacts, a bicycle—anything that needs to be represented. In this activity, you will write a simple Media Library application. You will create objects that represent your favorite songs, movies, and books.

Part I: Create a Song **Class**

1. Create a new project folder in VS. Call the project folder **MediaLib** and create a new class called MediaLib.

Your Media Library will eventually contain lists of your favorite media items, such as names of songs, movies, and books. Each of these categories will be a class in your program.

1. In VS, create a **Song.java** file. By convention, classes have names that begin with an uppercase letter, so be sure to capitalize properly. Click **Ok**.
2. Edit your Song class by defining two instance fields to represent the data for your Song class. The first is a rating on a scale of 1 to 10. Create a second data item for your Song class—the title of the song. This should be a String data type and also be private.
3. Add a “no-arg” constructor to your Song class that defines rating and title…both should be set to the correct starting value at this point.
4. Compile your code and fix any bugs you may have.

Part II: Make a Song Object

Creating a new object creates a place in memory where information about the object is stored. In this part of the activity, you will learn how to create a new object and how to access the information associated with the object.

1. Continuing in MediaLib, create a Song using the new keyword to instantiate, or create, a new object from the Song class:

|  |  |
| --- | --- |
| 1 | Song song1 = new Song(); |

The song1 variable is an object reference. An object reference refers to an object in memory; it points to a place in memory where the data for an object is stored. In this case, song1 points to where title and rating are stored as a Song object.

1. Use the following to print out the song1 variable:

|  |  |
| --- | --- |
| 1 | System.out.println(song1); |

What is shown?

When you use println to show the contents of the song1 variable, it shows the data type (Song) the @ sign, and then the reference or the address in memory of that object.

To show the actual data that the song1 variable points to is called de-referencingan object*.* You will de-reference an object in the next section.

Part III: Create Methods

De-referencing an object means to access the information, or data, contained in the object. A common way to de-reference an object and *access* its data is to use an object method called an accessor. A common way to de-reference an object and *set or change* its data is to use a method called a mutator. In this part of the activity, you will discover how to get and set data in an object.

1. Write an accessor method for your Song class called getTitle().

|  |  |
| --- | --- |
| 1  2  3 | public String getTitle() {  return title;  } |

* 1. The above method is public, meaning other objects, including your MediaLib object can use or access this method. If you had written private, other classes would not be able to access this method.
  2. After public, you specified String. This means that the getTitle() method will return or pass back some String value. Not all methods return something.
  3. Then there is the contents or the body of your method. Here, the method simply returns the title of your song.

1. Now write a mutator method for your Song class called setTitle().

|  |  |
| --- | --- |
| 1  2  3 | public void setTitle(String t) {  title = t;  } |

1. The above method is also public, meaning other objects, including your MediaLib object can use or access this method.
2. After public, you specified void. This method returns nothing. Notice it does not have a return statement like your accessor does.
3. After the method name setTitle, you specified (String t). This means a string value is passed to your method and assigned to the variable t. Think of t as the input to your method.
4. Finally, you assigned the value of t to title.
5. With accessor and mutator methods defined, you can get and set the title of a song. In the main method of MediaLib, add:

|  |  |
| --- | --- |
| 1  2 | song1.setTitle("Johnny B. Goode");  System.out.println(song1.getTitle()); |
|  |  |

1. In the same way you just created an accessor and a mutator for your title instance field, create an accessor and a mutator for the rating instance field. Show the rating of your song using System.out.println(…). Remember that rating is an int and not a String.

This may seem like a lot of work just to manage a bit of data, but it keeps data secure and consistent. Suppose, for example, you want to keep all ratings between 1 and 10, or between 1 and 5. Your mutator can make sure that happens.

Part IV: Create Movies and Books

Practice what you have learned by creating two new classes to keep track of movies and books. Refer to the code that is already written for the song class as an example.

1. In the same way you created a Song class, the instance fields, and the mutators and accessors, create a Movie class with the same instance fields, mutators, and accessors.
2. Repeat the process to create a Book class with the same instance fields and mutators and accessors.
3. Create a Movie and a Book object in the main method of MediaLib and show the data with: System.out.println(…);

Part V: One Class, Many Constructors

You have already created the Song class, but now you need to create many song

objects. In this part of the activity, you will learn how to create constructors that

makes it easier to instantiate objects in a class

1. Open your MediaLib project in VS, and open the Song class in the editor.
2. When you created the title and rating instance fields, you declared them. Using variables from your media library project, show an example of assignment dyslexia as described in the online resource.
3. Declare a new instance field called price in your Song class. Initialize it to 0.0 in the Song constructor. Check to be sure you made this a private instance field.
4. Provide an accessor and a mutator method for price.
   1. The accessor must return a data type that is the same as price, that is, a double.
   2. The mutator must also specify a double data type for its parameter.
5. In the main method (in your MediaLib class), use your newly written mutator and accessor methods to:
   1. Show the price of a song in the console.
   2. Assign a price for a song.
6. Back in your Song class, declare a boolean instance field called favorite.
7. Not all accessors and mutators need to use “get” or “set” in the method name. This is especially true when getting and setting boolean values.
8. Use the following mutator as a mutator of the favorite variable.

|  |  |
| --- | --- |
| 1. 1 2. 2 3. 3 | 1. public void addToFavorites() {    1. favorite = true; 2. } |

1. Eventually, you will create a lot of songs. With your current code, it may seem tedious to create many songs and set all of the titles, ratings, and prices. An easier way to create them would be to use one constructor and no accessors or mutators. It is possible to do this with your code—you just need a new constructor.
2. In general, a class can have more than one constructor as long as those constructors are different from one another. For example, the following is how you might create or instantiate a Song object using a constructor that has a title and a price:

Song song1 = new Song("Respect", 1.29);

The constructor would look like:

|  |  |
| --- | --- |
| 1  2  3  4 | public Song(String title, double price) {  this.title = title;  this.price = price;  } |

The declaration for the constructor, public Song(String title, double price), has two variables passed to it. There is a special name for this type of variable—parameter. The first parameter is title and the second parameter is price.

In the body of the constructor, the this.title syntax refers to the instance field title. With the this.title = title syntax, you are setting the title instance field to the value that was passed in using the title parameter. Using the same name for parameters and instance fields is a common practice in Java constructors. The this. syntax identifies which title variable to reference.

1. Copy and paste the code for the new constructor into your Song class. Compile and fix any bugs.
2. Using the new constructor as a template, create a third Song constructor that has a rating as its third parameter. Be sure to assign this.rating to the new parameter in this third constructor. Compile and check for bugs.
3. Using the constructor that has three parameters, create new songs so that you have at least seven or eight songs in your library. The titles don’t really matter at this point, but use a price of 1.29 or .99 for different songs and vary the ratings.

Part VI: Variables and Calculations

Java uses the standard mathematical operators to do calculations:

|  |  |
| --- | --- |
| + | Addition |
| - | Subtraction |
| \* | Multiplication |
| / | Division |

You will use these mathematical operators to do some calculations in this section.

1. First, calculate the total cost of all your songs. To do this, declare an accumulator variable. Name this double variable totalCost.
2. In addition to total cost, keep track of the number of songs you have. Create a variable called numSongs.
3. Every time you create a song, add the price to totalCost. Then add 1 to numSongs.
4. In a similar way, create a totalRatings variable and add all ratings of songs to it.

Part VI: Calculation Errors

You have the total cost of all songs and the total number of songs. In this part of the activity, create a new way to find out the average cost of a song in your media library.

1. You can now calculate the average cost:
   1. Create a well-named variable to store the average cost.
   2. Divide the total cost by the number of songs you have and store the result in your new variable.
   3. Using System.out.println(…), display the total cost and the average cost of all songs.
2. The calculated cost may not be what you expected. For example, add this code fragment somewhere in your main method:

|  |  |
| --- | --- |
| 1  2  3  4  5 | // testing a calculation:  double testVal = 109.41;  double testResult = testVal / 10;  System.out.println("Testing Result:");  System.out.println(testResult); |

The result of this calculation is 10.940999999999999. This may seem strange, since 109.41 divided by 10 should be 10.941. Computers store all data in a binary language of 1’s and 0’s (which represents the state of electricity flowing or not flowing). Certain numbers in our decimal system cannot be represented accurately in the binary number system on a computer. Doing calculations with decimals causes round-off errors. For now, you can ignore this round-off error.

1. In the same way you calculated an average cost, calculate an average rating for all of your songs and display it. Check the calculation manually to see whether your average is correct. It may not be!
2. To correctly calculate the average, the variable that stores the average rating (for example, aveRating) should be a double, totalRating should be an int, and numSongs should be an int.
3. Use a cast to get the correct result and check your calculations manually.

Part VII: The Modulus Operator

In addition to the standard math operators +, -, \*, and /, Java has a modulus operator represented by the % symbol. In this part of the activity, you will learn how to use the modulus operator to convert number of minutes into hours and minutes.

1. Create an integer duration variable for your Movie class and mutator and accessor methods for it.
2. Create a new method for your Movie class that displays the duration of the movie in a user-friendly format in hours and minutes.
3. Test your new method using durations such as 97, 134, or 199.
4. Improve the output: Display all output on one line using another method of the System.out class called print(…) that does not print a new line. Use syntax similar to:

|  |  |
| --- | --- |
| 1 | System.out.print("Hours"); |