L06
Due by the end of Friday, March 3rd

In this week's lab, you will practice writing a Java class with methods and testing them.

Today's Activities

Activity 1: The "regular" stuff.

Create a new project called `L06`, with the program name of `MusicTrackDriver`. Save it under the folder structure that you have created.

Now download `MusicTrackDriver.java` from Brightspace and replace the file you created above with it. Alternately, you can copy the content of the file you download to replace the content of the new file.

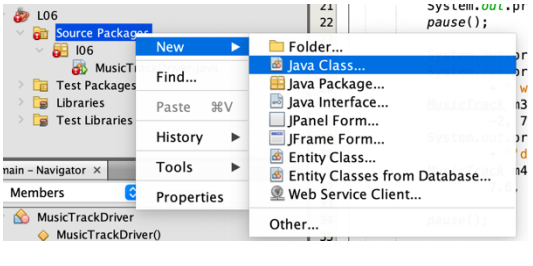
Once you do that, then you will see a bunch of errors. Just ignore them for now.

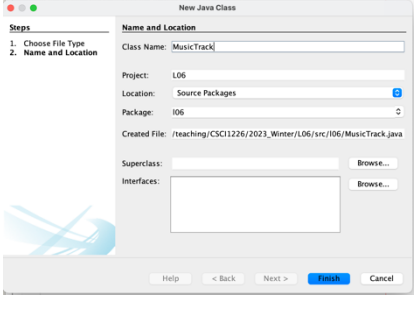
```
1  package l06;
2
3  /**
4   * This class tests methods in MusicTrack.<br><br>
5   *
6   * <strong>DO NOT CHANGE ANYTHING IN THIS CLASS!!</estrong
7   *
8   * @author yasushi akiyama
9   */
10 public class MusicTrackDriver {
11
12     /**
13      * @param args the command line arguments
14      */
15     public static void main(String[] args) {
16         System.out.println("=== Testing the default constructor ===");
17         MusicTrack m1 = new MusicTrack("Michael Brecker", "Midnight Mood",
18                 6.38, 2560174);
19         MusicTrack m2 = new MusicTrack("Beatles", "Come Together",
20                 4.32, 579091111);
21         System.out.println("(If you don't see any errors above, good!)");
22         pause();
23
24         System.out.println("\n--- now those with errors ---");
25         System.out.println("\nYou should see the error message for 'length'"
26                 + " with -2.0");
27         MusicTrack m3 = new MusicTrack("Queen", "Somebody To Love",
28                 -2, 729515997);
```

After this, **YOU MUST NOT CHANGE ANYTHING IN THIS FILE**. If you do, chances are your new program will not work as expected. You will only change and submit the new class that you will be creating in the rest of the activities.

Activity 2: Create a new class called `MusicTrack`.

This is explained in class, so if you do not know how to do it, you should review the corresponding lecture slides. The below screenshots are provided as reminders.





You should now modify the author tag to make sure that your name and A# appear appropriately. Also add javadoc comments to explain what this class is for (you need to figure out by reading this document).

Activity 3: Add private members of instance variables.

The next step is to declare (private) instance variables for this class. They are:

- `artist` of the type `String`, that holds the name of the artist.
- `title` of the type `String`, that holds the title of the music track.
- `length` of the type `double`, that represents the length of the track in minutes (e.g., 3 minutes and 30 second of track means `length` of 3.5)
- `downloadCount` of the type `int`, that represents the number of times this music track is downloaded.

As discussed in class, the instance variable declarations will go inside the class but outside of `main()`.

Activity 4: Add a stub for the (primary) constructor of the class.

You will now add a constructor to the class. In this constructor, you will have to initialize all the instance variables that you created above.

All the constructors will start with the word `public`, followed by the class name (i.e., `MusicTrack`), which is followed by zero or more parameters in ( ). Constructors are special kinds of methods, that do not have return types like typical methods, and they are invoked when an object of this class is created.

Since you are initializing the instance variables, you will need to have the matching parameters for them (`String artist`, `String title`, `double length`, `int downloadCount`), **in that order**.

Once you complete the above steps correctly (and save the file!), you should now notice some of the errors are gone in `MusicTrackDriver` but different errors appeared.

```
    System.out.println("\n\n=== Testing getters ===");
    System.out.println("Should print Michael Brecker: " + m1.getArtist());
    System.out.println("Should print Midnight Mood: " + m1.getTitle());
    System.out.println("Should print 6.38: " + m1.getLength());
    System.out.println("Should print 2560174: " + m1.getDownloadCount());
    pause();

    System.out.println("\n--- now those with errors ---\n"
            + "(They should have been set to zeros "
            + "in the default constructor)\n");

    System.out.println("Should print 0.0: " + m3.getLength());
    System.out.println("Should print 0: " + m4.getDownloadCount());
```

If you have done everything correctly, you "could" now run the program, but will crash after printing "`=== Testing getters ===`" (as shown in the screenshot below).

```
run:
=== Testing the default constructor ===
(If you don't see any errors above, good!)

... Press Enter to continue

--- now those with errors ---

You should see the error message for 'length' with -2.0 below

You should see the error message for 'length' with -5.0 below

You should see the error message for 'length' with -7.0 below

You should see the error message for 'downloadCount' with -58

You should see the error message for 'downloadCount' with -240

... Press Enter to continue


=== Testing getters ===
Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - Erroneous
tree type: <any>
        at l06.MusicTrackDriver.main(MusicTrackDriver.java:49)
/Users/yasushia/Library/Caches/NetBeans/12.4/executor-snippets/run.xml:111: The following er
ror occurred while executing this line:
/Users/yasushia/Library/Caches/NetBeans/12.4/executor-snippets/run.xml:68: Java returned: 1
```

Activity 5: Add getters (a.k.a. accessors) for all the instance variables.

This will be extremely easy! As seen in class, a `getter` is public, has the return type that matches the instance variable's data type, has no parameters, and has a name that looks like `getVariableName` (but with `VariableName` replaced with an instance variable name that this `getter` is written for). It (usually) has one line that simply returns that instance variable.

For example, the `getter` for `artist` should look like:

```
public String getArtist() {
    return artist;
}//end getArtist()
```

Add 3 more `getters` (i.e., a total of 4) for the rest of the instance variables. Once you do that, then the errors for the getters in the driver should now be gone. If you still see errors, then you have done something wrong.

```
System.out.println("\n\n=== Testing getters ===");
System.out.println("Should print Michael Brecker: " + m1.getArtist());
System.out.println("Should print Midnight Mood: " + m1.getTitle());
System.out.println("Should print 6.38: " + m1.getLength());
System.out.println("Should print 2560174: " + m1.getDownloadCount());
pause();

System.out.println("\n--- now those with errors ---\n"
        + "(They should have been set to zeros "
        + "in the default constructor)\n");

System.out.println("Should print 0.0: " + m3.getLength());
System.out.println("Should print 0: " + m4.getDownloadCount());

pause();
```

You can again run the program, and this time, you will go further than the last time:

```
run:
=== Testing the default constructor ===
(If you don't see any errors above, good!)

... Press Enter to continue

--- now those with errors ---

You should see the error message for 'length' with -2.0 below

You should see the error message for 'length' with -5.0 below

You should see the error message for 'length' with -7.0 below

You should see the error message for 'downloadCount' with -58

You should see the error message for 'downloadCount' with -240

... Press Enter to continue


=== Testing getters ===
Should print Michael Brecker: null
Should print Midnight Mood: null
Should print 6.38: 0.0
Should print 2560174: 0

... Press Enter to continue

--- now those with errors ---
(They should have been set to zeros in the default constructor)

Should print 0.0: 0.0
Should print 0: 0

... Press Enter to continue


=== Testing setters ===
Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - Erroneous sym type: l06.MusicTrack.setTitle
        at l06.MusicTrackDriver.main(MusicTrackDriver.java:65)
/Users/yasushia/Library/Caches/NetBeans/12.4/executor-snippets/run.xml:111: The following error occurred while executing this line:
/Users/yasushia/Library/Caches/NetBeans/12.4/executor-snippets/run.xml:68: Java returned: 1
BUILD FAILED (total time: 5 seconds)
```

The output is still wrong so we will need to fix it in the next activity.

Activity 6: Write the "body" of the constructor.

As mentioned above, you will need to initialize all the instance variables using the parameters of the constructor. That is, the value of `artist` is initialized using the value of the parameter for `artist`, the value of `title` is initialized using the value of the parameter for `title`, and so on. If you cannot remember how to do this, you will need to go back and check the lecture slides.

The first 2 (`String`) variables (`artist` and `title`) are straightforward, as each initialization code is only one-line long with an assignment operator (=).

But for the 2 other variables, you will need to validate the values of the parameters. For this, you will check if a parameter value is negative or not. If it is, then you will first print out an error message:

```
ERROR: Invalid length -2.0, setting it to 0
```

Where `-2.0` above is the parameter value. You will then assign 0 (zero) to that variable.

If the parameter value is zero or larger, then you can just use that value to initialize the variable.

Again you have seen a very similar example(s) to this in class, so if you cannot remember, make sure to review the lecture/slides.

Once you complete it, you can now run the program and see the different (correct) output as shown below:

```
run:
=== Testing the default constructor ===
(If you don't see any errors above, good!)

... Press Enter to continue

--- now those with errors ---

You should see the error message for 'length' with -2.0 below
ERROR: Invalid length -2.0, setting it to 0

You should see the error message for 'length' with -5.0 below
ERROR: Invalid length -5.0, setting it to 0

You should see the error message for 'length' with -7.0 below
ERROR: Invalid length -7.0, setting it to 0

You should see the error message for 'downloadCount' with -58
ERROR: Invalid downloadCount -58, setting it to 0

You should see the error message for 'downloadCount' with -240
ERROR: Invalid downloadCount -240, setting it to 0

... Press Enter to continue


=== Testing getters ===
Should print Michael Brecker: Michael Brecker
Should print Midnight Mood: Midnight Mood
Should print 6.38: 6.38
Should print 2560174: 2560174

... Press Enter to continue

--- now those with errors ---
(They should have been set to zeros in the default constructor)

Should print 0.0: 0.0
Should print 0: 0

... Press Enter to continue
```

Activity 7: Add setters (a.k.a. mutators).

The final task is to add `setters` to your class. Mostly, this is simple, especially for the 2 `String` instance variables in our class, as they do not involve any value checking. A setter for them is `public`, has `void` as its return type, and its name is `setVarilableName` with `VariableName` replaced by the actual instance variable name. For example, the setter for `artist` is:

```
public void setArtist(String artist) {
    this.artist = artist;
}//end setArtist()
```

Note that NetBeans differentiates 2 distinct `artists` in the above code by using different colours. The one in green is the instance variable `artist`, and the one in black is the parameter `artist`. So this is the correct way to use the `this` keyword, and the above code assigns the value of the parameter `artist` to the instance variable `artist`.

For the 2 instance variables that you validated the values for in the constructor, you will need a similar code that prints an error message when the value is invalid. But in setters, **you do not assign it to zero**. That is, if an invalid value is detected, then you only print out the error but do nothing else (i.e., **do not change the existing values!**). Also note that the error message is slightly different from that in the constructor.

If the value is valid, then use it to assign the value to the instance variable.

That's all the activities, and if you've done all the steps correctly, you should see the below output:

```
=== Testing the default constructor ===
(If you don't see any errors above, good!)

... Press Enter to continue

--- now those with errors ---

You should see the error message for 'length' with -2.0 below
ERROR: Invalid length -2.0, setting it to 0

You should see the error message for 'length' with -5.0 below
ERROR: Invalid length -5.0, setting it to 0

You should see the error message for 'length' with -7.0 below
ERROR: Invalid length -7.0, setting it to 0

You should see the error message for 'downloadCount' with -58
ERROR: Invalid downloadCount -58, setting it to 0

You should see the error message for 'downloadCount' with -240
ERROR: Invalid downloadCount -240, setting it to 0

... Press Enter to continue


=== Testing getters ===
Should print Michael Brecker: Michael Brecker
Should print Midnight Mood: Midnight Mood
Should print 6.38: 6.38
Should print 2560174: 2560174

... Press Enter to continue

--- now those with errors ---
(They should have been set to zeros in the default constructor)

Should print 0.0: 0.0
Should print 0: 0

... Press Enter to continue
```

```
=== Testing setters ===
Should (literally!!) print Something: Something
Should print 3.03: 3.03
Should print 235959475: 235959475

... Press Enter to continue

--- now those with errors ---
Should see the error for length (-1.0) but no change
ERROR: Invalid length -1.0, the value is unchanged

Should see the error for length (-20.0) but no change
ERROR: Invalid length -20.0, the value is unchanged

Should still print 3.03: 3.03

Should see the error for downloadCount (-100) but no change
ERROR: Invalid downloadCount -100, the value is unchanged

Should see the error for downloadCount (-50) but no change
ERROR: Invalid downloadCount -50, the value is unchanged

Should print 235959475: 235959475

... Press Enter to continue
```

Make sure to add javadocs for ALL the methods you have written including the constructor!

Submit only `MuiscTrack.java`.

You will be graded on the following:

1. You are identified by both name and A-number as an `@author` of the program, you added the program description in the javadoc comment for the class `MusicTrack`, and the program compiles and runs (no other points will be given if it doesn't compile or the program crashes without printing any output).
2. The program has 4 instance variables,
3. … which are defined `private`, and each has the correct data type.
4. It has a constructor with proper javadoc,
5. … which takes 4 parameters in the correct order,
6. … … and initializes the instance variables correctly using the parameters,
7. … … … with proper value validations,
8. … … … which display error messages and assign zeros when invalid values are found.
9. It has getters (accessors) with proper javadocs,
10. … and they work as expected.
11. It has setters (mutators) with proper javadocs,
12. … and they work as expected,
13. … with proper value validations,
14. … which display error messages (different one from those in the constructor) but do not change the values when invalid values are found.