Exercise 4.1

* I created a MusicOrganizer object and stored 3 audio files on it using the addFile method. I checked the number of files returned using the numberOfFiles method, which returned 3. I then used the listFile method to print the files.

Exercise 4.2

* Nothing happens when I removeFile(0). It does not give me an error. I did expect for it to give me an error.

Exercise 4.3

* I created a MusicOrganizer and added two file names to it. Then I called listFile(0) and listFile(1) to show the 2 files. Then I called removevFile(0) and then listFile(0). The second file that I added (which was originally printed when I called listFile(1)) was printed. This is what I expected. When I removed the first file name from the collection, the file at position 1 moved to position 0, therefore, when I called listFile(0) the second time, it returned the second file that I entered.

Exercise 4.4

* Private ArrayList<Book> library;

Exercise 4.5

* cs101 = new ArrayList<Student>();

Exercise 4.6

* private ArrayList<MusicTrack> tracks;

Exercise 4.7

* library = new ArrayList<Book>();
* library = new ArrayList<>();
* cs101 = new ArrayList<Student>();
* cs101 = new ArrayList<>();
* tracks = new ArrayList<MusicTrack>();
* tracks = new ArrayList<>();

Exercise 4.8

* If a collection stores 10 objects, the value that would be returned from a call to its size method is 10.

Exercise 4.9

* items.get(4);

Exercise 4.10

* The index of the last item stored in a collection of 15 objects is 14.

Exercise 4.11

* files.add(favoriteTrack);

Exercise 4.12

* dates.remove(2);

Exercise 4.13

* index 5

Exercise 4.14

* I added a method called checkIndex to the MusicOrganizer class that takes a single integer parameter and checks whether it is a valid index of the current state of the collection. If the parameter is not valid, it prints an error message saying what the valid range is. If the index is valid, it prints nothing.
* My method does work when I check an index if the collection is empty. It just prints the error message no matter what index I input as the parameter.

Exercise 4.15

* I wrote an alternative version of checkIndex called validIndex. It returns boolean values. I tested my method on the object bench with both valid and invalid parameters. It returns true when the parameter is valid and false when the parameter is not valid. I also tested the empty case. It returns false no matter what index I put as the parameter.

Exercise 4.16

* I rewrote the listFile and removeFile methods in MusicOrganizer so that they use the validIndex method to check their parameter.

Exercise 4.17

* When you use file names that do not exist, it prints “There was a problem playing: *file\_name*”, and there is a NullPointerException error.

Exercise 4.18

* public void listAllFiles()

Exercise 4.19

* No, you cannot write the body of the listAllFiles method like that because you do not know how many objects are stored in the ArrayList.

Exercise 4.20

* I implemented the listAllFiles method into my version of the music-organizer project.

Exercise 4.21

* I created a MusicOrganizer and stored a few files names in it. Then I used the listAllFiles method to print them out, and the method did work correctly.

Exercise 4.22

* I created an ArrayList<String> in the Code Pad. I then dragged the object icon to the object bench. I tried calling some of its methods, such as add, remove, size, isEmpty, and get. I then tried calling the add method from the Code Pad, which also worked.

Exercise 4.23

* I used the debugger to help me understand how the statements in the body of the loop in listAllFiles are repeated.

Exercise 4.24

* I completed a version of listAllFiles that includes the index of each file name in the listing.

![A screenshot of a cell phone

Description generated with very high confidence]()

Exercise 4.25

* I added the listMatching method to my version of the project. I checked the method, and it does only list the matching files.
* I also tried with a search string that matches none of the file names. Nothing is printed in this case.

Exercise 4.26

![A screenshot of a social media post

Description generated with very high confidence]()

Exercise 4.27

![A screenshot of a cell phone

Description generated with very high confidence]()

Exercise 4.28

* for (Track trackname : tracks)

Exercise 4.29

* boolean found = false;

while (! found)

{

if (the keys are in the next place)

{

found = true;

}

}

Exercise 4.30

* public void multiplesOfFive()

{

int number = 10;

while (number <= 95)

{

System.out.println(number);

number += 5;

}

}

Exercise 4.31

* public void addNum()

{

int i = 1;

int sum = 0;

while (i <= 10)

{

sum = sum + i;

i ++;

}

System.out.println(sum);

}

Exercise 4.32

* public int sum(int a, int b)

{

int num = a;

int sum = 0;

while (num <= b)

{

sum = sum + num;

num ++;

}

return sum;

}

Exercise 4.33

* public boolean isPrime(int n)

{

int divide = 2;

boolean prime = true;

while (divide <= (n-1) && prime)

{

if (n % divide == 0)

{

prime = false;

return prime;

}

}

if (prime)

{

return prime;

}

}

Exercise 4.34

* No, the value returned by size does not vary from one check to the next.

![A screenshot of a cell phone

Description generated with very high confidence]()

4.35

![A screenshot of a cell phone

Description generated with very high confidence]()

4.36

![A screenshot of a social media post

Description generated with very high confidence]()

4.37

* I added a genre field to the Track class and provided accessor and mutator methods for it. I allowed the user to set the value of this field.

4.38

* I added stopPlaying() to playTrack() once the index had been validated.

4.39

![A screenshot of a cell phone

Description generated with very high confidence]()

4.56

![A screenshot of a cell phone

Description generated with very high confidence]()

4.57

![A screenshot of a cell phone

Description generated with very high confidence]()

4.58

![A screenshot of a cell phone

Description generated with very high confidence]()

4.59

![A screenshot of a cell phone

Description generated with very high confidence]()

4.60

![A screenshot of a cell phone

Description generated with very high confidence]()

![A screenshot of a cell phone

Description generated with very high confidence]()

![A screenshot of a cell phone

Description generated with very high confidence]()