**Chapter 4**

Ex. 4.1 – Done

Ex 4.2 - If I try to remove a file before adding one no error occurs. After glancing at the code, it seems that I wouldn’t get an error and that is because of how the method it written. It merely checks to make sure it is a valid element and then clears its contents so even if there is nothing there it works as if there is.

Ex 4.3 – Depending how java works the deleted item works it will either re-assign subscripts to the following elements if I remove a item that comes before other so my item 1 will become item 0 when the original item 0 8 is removed. The other scenario is that it leaves a hole and we get null or an error. After testing it the first case appears true.

Ex. 4.4 – private ArrayList< Book > library = new ArrayList< String >();

Ex. 4.5 – private ArrayList< Student > cs101 = new ArrayList< Student >();

Ex. 4.6 – private ArrayList< MusicTrack > tracks = new ArrayList< MusicTrack >();

Ex. 4.7 – library.add(“1984”);

Cs101.add(“ Jacob Knott” );

Track.add(“ Stairway to Heaven” );

Ex. 4.8 – If a collection stores 10 items the returned value from the method size would be 9.

Ex. 4.9 – object.get( 5 )

Ex. 4.10 – The index attached to the last object in an Arraylist with 15 objects would be 14.

Ex. 4.11 – addFavorite(favoriteTrack, files)

Ex. 4.12 – dates.remove(3)

Ex. 4.13– If an object is stored with an index value of 6 and then objects with indexes 0 and 9 are removed the object that was 6th will become linked to the 5th index because everything will move to fill in to the empty space. However since 9 is greater than six its removal will have no effect on the object.

Ex. 4.14– Done and saved.

Ex. 4.15– Done and saved.

Ex. 4.16 – Done and saved.

Ex. 4.17 – I am having trouble with this one. It just keeps listing altereations of this error message

“Exception in thread "Thread-9" java.lang.NullPointerException

at MusicPlayer$1.run(MusicPlayer.java:62)”the thread number changes (6, 7, 10, 9). I am also trying to see what happens if I move the audio file out so it is so buried. To find its current location I have this mess

“This PC/Documents/Jake's Stuff/Winter 2017/Java/source\_code\_projects/projects/chapter04/audio/BlindBlake-EarlyMorningBlues.mp3”and have tried

Jake's Stuff/Winter 2017/Java/source\_code\_projects/projects/chapter04/audio/BlindBlake-EarlyMorningBlues.mp3

Documents/Jake's Stuff/Winter 2017/Java/source\_code\_projects/projects/chapter04/audio/BlindBlake-EarlyMorningBlues.mp3

Java/source\_code\_projects/projects/chapter04/audio/BlindBlake-EarlyMorningBlues.mp3I have even copied and saved it to my desktop screen to try and reduce likelihood of human error but neither of my attempts work and it flashes the same message but with thread numbers 11 and 12. My attempts were

“This PC/Desktop/BlindBlake-EarlyMorningBlues.mp3”and

“This PC/Desktop/BlindBlake-EarlyMorningBlues.mp3”I must be making some kind of error that I don’t know about and have been unable to find anything truly useful online the closest I got was this stackoverflow <https://stackoverflow.com/questions/33748882/sound-will-not-play-from-jar-but-only-when-i-run-it-from-netbeans-javazoom-pl>.

Ex. 4.18 – the header will likely look like “public void listAllFiles()”. It wouldn’t have a return type as I would print the information to the screen.

Ex. 4.19– We could write the listAll method like this but we would have to write the print command for every index. If there were 3 there would be 3 and if 10 then 10 etc. although doing this what we “Know” ridiculous. I’m assuming this is a Segway to the for loop, (it was invaluable in C++).

Ex. 4.20– Done and saved

Ex. 4.21– Done

Ex. 4.22– Done

Ex. 4.23– Done

Ex. 4.24– Done and saved

Ex. 4.25– Done and saved

Ex. 4.26– Done and saved

Ex. 4.27– REVISIT SOON

Ex. 4.28– public void tracks(){

for(Track track : tracks) {

}

Ex. 4.29– Boolean found = false;

While( found ){

If(the keys are in next place){

found = false;

}

}

Ex. 4.30– int index = 10;

While(index <= 95){

System.out.println(index);

Index = index \* 5;

}

Ex. 4.31– int total = 0;

Int x = 1

while (x < =10){

total += x;

}

Ex. 4.32– public void sum(int a, int b){

int sum = 0;

while (a <=b){

result += a;

a++;

}

}

Ex. 4.33– private static Boolean isPrime(int n){

if (n < 2) return false;

if (n == 2) return true;

if (n % 2 == 0) return false;

for (int x = 3; x \* x <= n; x += 2)

if (n % x == 0) return false;

return true;

}

Ex. 4.34– The value returned from size should not vary during the findFirst method.

Ex. 4.35– Done and saved

Ex. 4.36 – Done and saved

Ex. 4.37 – Done and saved

Ex. 4.38 – By calling the stopPlaying method first in any method that starts another track the playing track should be stopped and the next should begin.

Ex. 4.39 – Done and saved CHECK FUNCTIONALITY REDO WITH ITERATOR TOOL

Ex. 4.40 – Done and Saved

Ex. 4.41 – Done and Saved

Ex. 4.42 – Done and Saved

Ex. 4.43 – Done and Saved

Ex. 4.44 – My first thought would be to randomize the index values but this cannot be done as an index value is assigned to a specific track unless on is removed or added. If I want to hold to my initial though of shuffling it a list containing all tracks in the organizer could be created and randomized, then it could play them all. As to getting them to only play an equal amount a while or statement could be used with a counter to only play a track when its playCount is less than the counter and when no more tracks can be played it increments the counter. That would have it play music forever.

Ex. 4.45 – Done and Saved

Ex. 4.46 – Done

Ex. 4.47 – Done and Saved

Ex. 4.48 – Done and saved

Ex. 4.49 –

Ex. 4.50 –

Ex. 4.51 –

Ex. 4.52 –

Ex. 4.53 –

Ex. 4.54 –

Ex. 4.55 –

Ex. 4.56 –

Ex. 4.57 –

Ex. 4.58 –

Ex. 4.59 –

Ex . 4.60

Ex . 4.61

Ex . 4.62

Ex . 4.63

Ex . 4.64

Ex . 4.65

Ex . 4.66

Ex . 4.67

Ex . 4.68

Ex . 4.69

Ex . 4.70

Ex . 4.71

Ex . 4.72

Ex . 4.73

Ex . 4.74

Ex . 4.75

Ex . 4.76

Ex . 4.77

Ex . 4.78

Ex . 4.79

Ex . 4.80

Ex . 4.81

Ex . 4.82

Ex . 4.83

Ex . 4.84

Ex . 4.85

Ex . 4.86

Ex . 4.87

Ex . 4.88

Ex . 4.89

Ex . 4.90