Chapter 4
4.1: I added music and searched each file number
4.2: No feedback is received when deleting a file that isn't there and i'm not sure there needs to be since if something was there is isn't anymore regardless
4.3: The file that was added in second took the first index value when original file was removed

4.4: private ArrayList<book> library;
4.5:ArrayList<student> cs101;
4.6: private ArrayList<MusicTrack> tracks;
4.7: library = new ArrayList<book>(); or library = new ArrayList<>();
cs101 = new ArrayList<student>(); or cs101 = new ArrayList<>();
tracks = new ArrayList<MusicTrack>(); or tracks = new ArrayList<>();

4.8: 10
4.9: items.get(4);
4.10: 14
4.11: files.add(favoriteTrack);

4.12: dates.remove(2);
4.13: 5
4.14:
```
public void checkIndex(int index)
{
   if(index > 0 && index <= files.size()) {
      System.out.println("This is a valid index value");
   }
   else{
      System.out.println("This is not valid index value");
   }
}
```
Yes there is a line is printed to tell the user if it is valid or not
4.15:
```
public boolean validIndex(int index)
{
   if(index >= 0 && index <= files.size()) {
      return true;
   }
   else{
      return false;
   }
}
```

4.16:
```
public void listFile(int index)
{
    if(validIndex (index)) {
        String filename = files.get(index);
        System.out.println(filename);
    }
}

public void removeFile(int index)
{
    if(validIndex (index)) {
        files.remove(index);
    }
}
```

4.17: *listened to some blues*

4.18: public void listAllFiles()
4.19: However Many files were in the collection would be how many lines we would need so it depends on the collection

4.20:
```
public void listAllFiles()
{
    for(String filename : files) {
        System.out.println(filename);
    }
}
```

4.21: the listAllFiles method works
4.22:*code pad used*
4.23: *debugger used for further understanding*

4.24:
```
public void listAllFiles()
{
    int position = 0; //local int variable
    for(String filename : files) {
        System.out.println(position + ": " + filename);
        position++;
    }
}
```

4.25: Method added

4.26:
```
public void listmatching(String searchString)
{
    boolean notAMatch = true;
    for(String filename : files) {
        if(filename.contains(searchString)){
            System.out.println(filename);
            notAMatch = false;
        }
    }
    if(notAMatch) {
        System.out.println("No files found that matches: " + searchString);
    }
}
```

4.27:
```
public void playSample(String artist)
{
    int position = 0; //local int variable
    for(String filename : files) {
        if(filename.contains(artist)) {
            player.startPlaying(artist);
        }
        position++;
    }
}
```

4.28:
```
for(Track track : tracks)
```

4.29:
```
boolean found = false;
while(!false) {
        If (the keys are in the next place) {
        Missing = true;
        }
}
```

4.30:
```
public void multiplesOfFive()
{
    int multiple = 10;
    while(multiple <= 95){
        System.out.println(multiple);
        multiple = multiple + 5;

    }
}
```

4.31:
```
public void Sum()
{
   int sum = 0;
   int num = 1;
   while(num <= 10) {
   sum = num + sum;
   num++;
   }
   System.out.println(sum);
}
```

4.32:
```
public void sum(int a, int b)
{
   int Sum = 0;
   int num = a;
   while(num <= b) {
   Sum = num + Sum;
   num++;
   }
   System.out.println(Sum);
}
```

4.33:
the while loop divides every number between 2 and (n-1) and only returns true if n never has a remainder of 0 but will return false if when a number is divided by n evenly

```
public boolean isPrime(int n)
  {
     int division = 2;
     while(division < n){
        if(n % division == 0)
        {
           return false;
        }
        division++;
     }
     return true;
  }
```

4.34: the value by size does not vary
This can be changed by storing file.size() in a local variable before the loop

```
public int findFirst(String searchString)
{
    int index = 0;
    // Record that we will be searching until a match is found.
    boolean searching = true;
    //add a local variable to check and store before the loop
    int localSize = files.size();
    while(searching && index < localSize) {
        String filename = files.get(index);
        if(filename.contains(searchString)) {
            // A match. We can stop searching.
            searching = false;
        }
        else {
            // Move on.
            index++;
        }
    }
    if(searching) {
        // We didn't find it.
        return -1;
    }
    else {
        // Return where it was found.
        return index;
    }
}
```

4.35:

```
// new field added to the Track class
private int playCount;
//methods to reset and increment field
public void resetPlayCount()
    {
        playCount = 0;
    }
    public void increasePlayCount()
    {
        playCount++;
    }
```

4.36:

```
public void playTrack(int index)
{
    if(indexValid(index)) {
        Track track = tracks.get(index);
        player.startPlaying(track.getFilename());
        System.out.println("Now playing: " + track.getArtist() + " - " + track.getTitle());
    }
    tracks.increasePlayCount();
}
```

4.37:

```
//new field
private String genre;
// accessor and mutator for field
public String getGenre()
   {
      return genre;
   }

//Or

public String getDetails()
   {
      return genre + ": " + artist + ": " + title + "  (file: " + filename + ")";
   }

//And
public void addGenre(String Genre)
   {
      Genre = genre;
   }
```

4.38:

```
   public void playTrack(int index)
   {
      if(indexValid(index)) {
          Track track = tracks.get(index);
          player.stop.(); //stops the previous music before starting the next
          player.startPlaying(track.getFilename());
          System.out.println("Now playing: " + track.getArtist() + " - " + track.getTitle());
      }
    }
```

4.39:
```
public void deleteTitle(String title)
{
    Iterator<Track> it = tracks.iterator();
        while(it.hasNext()){
        Track t = it.next();
        if(title.contains(title)) {
                it.remove();
        }
    }
}
```

4.40:
```
import java.util.ArrayList;


 // Define any necessary fields here ...
private ArrayList<Membership> members;

// Initialise any fields here ...
members = new ArrayList<>();
```

4.41:
```
public int numberOfMembers()
{
     return members.size();
}
```

4.42:
```
public void join(Membership member)
{
    members.add(member);
}
```

4.43:

4.44:

4.45:

4.46: created auction and a couple of lots, bids, persons

4.47:
```
boolean successful = selectedLot.bidFor(new Bid(bidder, value));
```

4.48:

```
public void close()
{
    for (Lot lot : lots) {
        System.out.println(lot.getNumber() + ": " + lot.getDescription());
        //if there is a bid
        Bid highestBid = lot.getHighestBid();
        if (highestBid != null)  {
            System.out.println("Lot: " + lot.getNumber() + ", higest bidder was " +
            highestBid.getValue());
            System.out.println("Bid was from " + highestBid.getBidder().getName());
        }
        else
        {
            System.out.println("Lot: " + lot.getNumber() + " has no bids");
        }
    }
}
```

4.49:

```
public ArrayList<Lot> getUnsold(){
    ArrayList<Lot> unsoldLots = new ArrayList<Lot>();
      for(Lot lot : lots) {
       Bid highestBid = lot.getHighestBid();
       if(highestBid == null) {
          unsoldLots.add(lot);
       }
    }
       return unsoldLots;
 }
```

4.50: The getLot method is assuming that the location in the arraylist is at getLotNumber() -1.I would think the index numbers could be changes if the lots can be removed. The getLot method would also print out a message if there was an error.

4.51:

4.52:

4.53:

4.54:

4.55:

4.56:

4.57:

4.58:

4.59:

4.60: