Eric Wallace

Chapter 4 Exercises

4.1- Did it, entered 4 song titles

4.2- There was no error. I was expecting that it would error when trying to delete.

4.3- The item at listFile(1) moved up, no I thought it would have had an error or done nothing but looks like it adjusts to the next file in order to regain the memory storage.

4.4- private ArrayList<book> library;

4.5- private ArrayList<Student> cs101;

4.6- private ArrayList<tracks> MusicTrack;

4.7-

library = new ArrayList<>();

cs101 = new ArrayList<>();

tracks = new ArrayList<>();

4.8- 10

4.9-

public int getItems(4)

{

return items;

}

4.10- 14

4.11-

public void addFile(String file)

{

files.add(file);

}

4.12-

public void removeDate(int index)

{

if(index >= 0 && < dates.size()) {

dates.remove(index);

}

}

then enter an index of 2 for the third object.

4.13- 5

4.14- Did it.

4.15- Did it.

4.17- It throws an error and will not play.

4.18- public void listAllFiles()

4.19- There would need to be as many println statements as there were indexes following the pattern.

4.20- Did it.

4.21- Printed all three I entered.

4.22- Did it.

4.23- Did it.

4.25- Nothing prints with a string that doesn't match.

4.27- Dit it.

4.28- for (Track track : tracks)

4.29-

boolean found = false;

while(!found) {

if(the keys are in the next place) {

found = true;

}

}

4.30-

public void multiplesOfFive()

{

int multiple = 10;

while (multiple <= 95) {

System.out.println(multiple);

multiple = multiple + 5;

}

}

4.31-

int sum = 0;

int number = 1;

while(number <= 10) {

sum += number;

}

System.out.println("The sum is " + sum);

4.32-

public int sum( int a, int b)

{

int sum = 0;

int number = a;

while (number <= b) {

sum = sum + number;

number += 1;

}

return sum;

}

4.33-

4.34- It doesn't vary.

4.35- Did it.

4.36- Done.

4.37- Done, i added genre field.

4.39-

public void removeTitles(String title)

{

Iterator<Track> it = tracks.iterator();

while(it.hasNext()) {

Track t = it.next();

if(t.getTitle().contains(title)) {

it.remove();

}

}

}

4.40- Did it.

4.41- Did it.

4.43-

public void randomPlay()

{

if(tracks.size() > 0) {

Random rand = new Random();

int index = rand.nextInt(tracks.size());

playTrack(index);

}

}

4.44- Play them randomly.

4.45-

public void randomPlayAll()

{

ArrayList<Track> leftToPlay = new ArrayList<Track>(tracks);

Collections.shuffle(leftToPlay);

for(Track t : leftToPlay) {

player.playSample(t.getFilename());

}

}

public void randomPlayAll()

{

Random rand = new Random();

ArrayList<Track> leftToPlay = new ArrayList<Track>(tracks);

while(leftToPlay.size() > 0) {

int index = rand.nextInt(leftToPlay.size());

Track t = leftToPlay.remove(index);

player.playSample(t.getFilename());

}

}

4.46- Did it.

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());

Bid highestBid = lot.getHighestBid();

if(highestBid != null) {

System.out.println("Highest bidder: " + highestBidder().getName());

System.out.println("Bid: " + highestBid.getValue());

}

else {

System.out.println("Not Sold");

}

}

}

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);

}

}

}

4.50- There would be an error message.

4.51-

public Lot getLot(int number)

{

Lot lot = null;

if(lots.size() > 0) {

lot = lots(0);

int nextIndex = 1;

while(lot.getNumber() != number && nextIndex < lots.size()) {

lot = lots.get(nextIndex);

nextIndex++;

}

}

if (lot == null || lot.getNumber() != number) {

System.out.println("Lot number: " + number + " does not exits.");

return null;

}

}

4.52-

public Lot removeLot (int number)

{

Lot lot = getLot(number);

if(lot != null) {

lots.remove(lot);

}

return lot;

}

4.53-

Methods they share:

get(int index)

add(Element)

remove(int index)

booleancontains(Object)

voidclear()

intindexOf(Object)

Iterator <T> iterator()

voidset(in index, E element)

intsize()

ArrayLists have : ensureCapacity, forEach, sort, subList, trimToSize

LinkedLists have : addFirst and addLast, getFirst and getLast, descendingIterator, removeFirst and removeLast, push and pop.

4.54- Did it.

4.55- Did it.

4.56- Did it.

4.57- Did it.

4.58- Did it.

4.59- Did it.

4.84-

int i = 1;

do {

System.out.print(i + " ");

i++;

} while( i <= 10);

System.out.println();

It throws an exception.

4.85- Nothing happens when the collection is empty.

4.86- Did it

4.87- Switch statement is used to select one of many code blocks being executed.

switch(expression) { case a: // code block break; case b:// code block break; default:// code block}