Bestudeer en ***maak de opgaven*** 4.1 t/m 4.23, 4.25, 4.27 t/m 4.32, 4.34 t/m 4.42, 4.46 t/m 4.59, 4.61 t/m 4.77.

4.1

Ja klopt

4.2

Dan maakt hij die gewoon leeg ook al is hij er niet.

4.3

Zet hij 1 naar 0 schuift dus op. Niet verwacht.

4.4

private ArrayList<Book> library

4.5

ArrayList<Student> cs101

4.6

Private ArrayList<MusicTrack> tracks;

4.7

variable name = new Arraylist<>

4.8

10

4.9

Items.get(5);

4.10

14 want begint bij 0

4.11

Files.add(favouriteTrack);

4.12

Dates.remove(2);

4.13

Index 5 gaat 1 af en erboven heft ie geen last van

4.14

public void checkIndex(int check){

if ((check >= 0) && (check <= files.size()-1)){

}

else{

System.out.println("Valid range must be between 0 and -1");

}

}

4.15

public boolean checkIndex2(int check2){

if ((check2 >= 0) && (check2 <= files.size()-1)){

return true;

}

else{

return false;

}

4.16

public void removeFile(int index)

{

if(checkindex2(index))

{

files.remove(index);

}

}

public void listFile(int index)

{

if(checkindex2(index)){

System.out.println(files.get(index));

}

}

4.17

Dan zegt ie there was a problem

4.18

Public String listAllFiles();

4.19

Geen idee je weet niet hoe lang de array is.

4.20

Ok

4.21

Werkt.

4.22

Illegal start of expression..

4.23

Werkt niet.

4.25

public void listMatching(String searchString)

{

for(String filename : files){

if(filename.contains(searchString)){

// a match

System.out.println(filename);

}

}

}

4.28

ArrayList<Track> tracks = new ArrayLIist<Track>;

For( )

4.29

Boolean found = true;

While(missing){

If

{

missing = false;

}

Boolean found = false;

While(missing){

If {

Missing = true;

}

}

4.30

int index = 10;

while(index <= 95){

System.out.println(index);

index = index + 5;

}

4.31

public void sum(){

int index = 1;

int sum = 0;

while (index <= 10){

sum = sum + index;

index ++;

}

System.out.println(sum);

}

4.32

pubic int sum (int A, int B){

int index = A;

int final = 0;

while (index <= B){

final += index;

initial++;

}

}

4.34

Ok

4.35 36

private int playCount;

public void resetPlayCount()

{

playCount = 0;

}

public void incrementPlayCount()

{

playCount += 1;

}

public int getPlayCount()

{

return playCount;

}

4.37

private void setDetails(String artist, String title, String filename)

{

this.artist = artist;

this.title = title;

this.filename = filename;

}

4.38

player.stop(); toegevoegd aan de playTrack methode, bovenaan de if-statement

4.39

public void removeCertainTracks(String titleToRemove)

{

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

while(it.hasNext())

{

Track t = it.next();

String title = t.getTitle();

if(title.equals(titleToRemove))

{

it.remove();

}

}

}

4.40

public Club()

{

// Initialise any fields here ...

members = new ArrayList<Membership>();

}

4.41

public int numberOfMembers()

{

return members.size();

}

4.42

/\*\*

\* Add a new member to the club's list of members.

\* @param member The member object to be added.

\*/

public void join(Membership member)

{

members.add(member);

}

4.46

Ok

4.47

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

4.48

public void closeAuction()

{

for(Lot lot : lots)

{

Bid highestBid = lot.getHighestBid();

if(highestBid == null)

{

System.out.println(lot.toString() + " - This lot has not been sold.");

}

else

{

System.out.println(lot.toString() + " - This lot has been sold to " + highestBid.getBidder().getName());

}

}

}

4.49

public ArrayList<Lot> getUnsold()

{

ArrayList<Lot> unsoldLots = new ArrayList<Lot>();

for(Lot lot : lots)

{

if(lot.getHighestBid() == null)

{

unsoldLots.add(lot);

}

}

return unsoldLots;

}

4.50

Dat er niet uitkomt wat je verwacht

4.51

public Lot getLot(int lotNumber)

{

// lot exists?

if ((lotNumber >= 1) && (lotNumber < nextLotNumber))

{

for (Lot lot : lots)

{

if (lotNumber == lot.getNumber())

{

return lot;

}

}

}

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

return null;

}

4.52

public Lot removeLot(int number)

{

int counter = 0;

for (Lot lot : lots)

{

if (number == lot.getNumber())

{

lots.remove(counter);

System.out.println("Lot number " + number + " has been removed.");

return lot;

}

counter++;

}

//not found

System.out.println("Something went wrong.");

return null;

}

4.54

/\*\*

\* Determine the number of members who joined in the

\* given month.

\* @param month The month we are interested in.

\* @return The number of members who joined in that month.

\*/

public int joinedInMonth(int month)

{

int membersJoined = members.size();

int counter = 0;

if(month < 1 || month > 12)

{

System.out.println("Invalid month!");

}

else

{

for(Membership membership : members)

{

if(membership.getMonth() == month)

{

counter++;

}

}

}

return counter;

}

4.55

public ArrayList<Membership> purge(int month, int year)

{

ArrayList<Membership> collection = new ArrayList<Membership>();

if(month < 1 || month > 12)

{

System.out.println("Invalid month!");

return collection;

}

else

{

for(Membership membership : members)

{

if(membership.getMonth() == month && membership.getYear() == year)

{

System.out.println(membership.getName());

//String name = membership.getName();

//collection.add(name);

}

}

return collection;

}

}

4.56

public void printProductDetails()

{

for(Product product : stock)

{

System.out.println(product);

}

}

4.57

public Product findProduct(int id)

{

for(Product product : stock)

{

if(product.getID() == id)

{

return product;

}

}

return null;

}

4.58

public int numberInStock(int id)

{

return findProduct(id).getQuantity();

}

4.59

public void delivery(int id, int amount)

{

for(Product product : stock)

{

if(product.getID() == id)

{

product.increaseQuantity(amount);

}

}

}

4.61

Alles geeft ie 0.

4.62

Person[] people;

4.63

boolean[] vacant;

4.64

4x

4.65

int[] counts;

boolean[] occupied;

4.66

double[] readings = new double[60];

String[] urls = new String[90];

TicketMachine[] machines = new TicketMachine[5];

4.67

20

4.68

Geen() maar

double[] prices = new double[50];

4.69

4.70

public void printHourlyCountsTWO()

{

System.out.println("Hr: Count");

int hour = 0;

while(hour < hourCounts.length)

{

System.out.println(hour + ": " + hourCounts[hour]);

hour++;

}

}

4.71

public void printGreater(double [] marks, double mean)

{

for(int index = 0; index < marks.length; index++)

{

if(marks[index] > mean)

{

System.out.println(marks[index]);

}

}

}

4.72

public LogAnalyzer(LogfileReader r)

{

hourCounts = new int[24];

reader = r;

}

4.73

public int numberOfAccesses()

{

int total = 0;

int counter = 0;

// Add the value in each element of hourCounts

// to total.

while(counter < hourCounts.length)

{

total += hourCounts[counter];

counter++;

}

return total;

}

4.74

Ok

4.75 public void printBusiestHour()

{

int highestNumber = 0;

int hour = 1;

int totalHours = 0;

for(int i = 0; i < hourCounts.length; i++)

{

if(highestNumber < hourCounts[i])

{

highestNumber = hourCounts[i];

hour++;

}

totalHours++;

}

System.out.println("The busiest hour is: " + (totalHours - hour) + " and the number of accesses at this hour is :" + highestNumber);

}

4.76

public void printQuietestHour()

{

int lowestNumber = 999999999; //STUPIDLY HIGH!

int hour = 1;

for(int i = 0; i < hourCounts.length; i++)

{

if(lowestNumber > hourCounts[i])

{

lowestNumber = hourCounts[i];

hour++;

}

}

System.out.println("The quitest hour is: " + (hour+2) + " and the number of accesses at this hour is: " + lowestNumber);

}

4.77

De eerste