# Train\_ticket\_reservation\_system\_part\_02

first name : Buddhin Saroj Student last name : Samarasinghe

Contents

1. Code explanation method by method with outputs 01
   * addPasenger( )
   * viewTrainQueue( )
   * deletePassenger ( )
   * saveData ( )
   * loadData ( )
   * simulation ( )
   * Quit ( )
2. Test plan… 20

***TrainStation (Main Class)***

## addPassenger ( ) method

private void addPassenger() {

if (PassengerQueue.isFull()){

System.out.println(" Passenger Queue is full! ");

}else if (waitingRoom.size()!=0) { Stage gui = new Stage();

AnchorPane guiAnchorpane = new AnchorPane(); guiAnchorpane.setStyle("-fx-background-color: #17202A "); GridPane firstPane = new GridPane();

firstPane.setStyle("-fx-background-color: #17202A "); Label waitingroom = new Label("Waiting Room"); waitingroom.setStyle("-fx-text-fill: #F7F9F9 ");

Label passengerQueue = new Label("Train Queue"); passengerQueue.setStyle("-fx-text-fill: #F7F9F9 "); GridPane secondPane = new GridPane(); secondPane.setStyle("-fx-background-color: #17202A "); Button[] btn = new Button[42]; //button array

Button btn\_add = new Button("Add"); //create add button and set styles btn\_add.setDisable(true);

btn\_add.setPrefHeight(30); btn\_add.setPrefWidth(100.0);

btn\_add.setStyle("-fx-background-color:#196F3D"); btn\_add.setLayoutX(150); btn\_add.setLayoutY(900);

Button btn\_delete = new Button("Delete"); //ceate delete button and set styles btn\_delete.setDisable(true);

btn\_delete.setPrefHeight(30.0);

btn\_delete.setPrefWidth(100.0);

btn\_delete.setStyle("-fx-background-color: #c70000"); btn\_delete.setLayoutX(250); btn\_delete.setLayoutY(900);

Button btn\_ok = new Button("OK"); btn\_ok.setPrefHeight(30); btn\_ok.setPrefWidth(195.0);

btn\_ok.setStyle("-fx-background-color:#2874A6"); btn\_ok.setLayoutX(500);

btn\_ok.setLayoutY(900);

firstPane.setPrefSize(500, 800); firstPane.setLayoutX(150); firstPane.setLayoutY(75); firstPane.setHgap(15); firstPane.setVgap(15);

AtomicReference<Passenger> variable = new AtomicReference<>(waitingRoom.get(0)); AtomicReference<Button> variableBtn = new AtomicReference<>(btn[0]); //store clicked button int count = 0;

for (Passenger passenger : waitingRoom) { if (passenger == null) continue;{

btn[count] = new Button("Name : " + passenger.getName() + "\nSeat No : " + passenger.getSeat()); btn[count].setPrefSize(195, 180);

btn[count].setStyle("-fx-background-color: #F1C40F "); int finalCount = count;

btn[count].setOnAction(event -> {

if (variableBtn.get()!=null) variableBtn.get().setStyle("-fx-background-color: #F1C40F "); btn[finalCount].setStyle("-fx-background-color: #17A589 ");

variable.set(passenger);

variableBtn.set(btn[finalCount]); //store clicked button in variableBtn btn\_add.setDisable(false);

btn\_delete.setDisable(false);

});

count++;

}

}

int num = 0;

for (int a = 0; a < 1; a++) {

for (int b = 0; b < count; b++) {

firstPane.add(btn[num++], a, b);//allign the created checkboxes in GridPane.

}

}

Random rand = new Random(); int rand\_no = rand.nextInt(6)+1; secondPane.setPrefSize(500, 800); secondPane.setLayoutX(500); secondPane.setLayoutY(75); secondPane.setHgap(15); secondPane.setVgap(15);

Button[] pasBtn = new Button[6]; int count3 = 0;

while (count3 < rand\_no) { //create the check boxes pasBtn[count3] = new Button("Passenger : " + (count3 + 1)); pasBtn[count3].setPrefSize(195, 180); pasBtn[count3].setStyle("-fx-background-color: #B3B6B7 "); count3++;

}

int num3 = 0;

for (int x = 0; x < 6; x++) { for (int y = 0; y < 1; y++) {

if (num3 == rand\_no) break;

secondPane.add(pasBtn[num3++], y, x);//allign the created checkboxes in GridPane.

}

}

AtomicInteger passengerCount= new AtomicInteger(); btn\_ok.setOnAction(event -> {

gui.close();

});

btn\_add.setOnAction(event -> {

pasBtn[passengerCount.get()].setStyle("-fx-background-color: #1D8348 "); PassengerQueue.addToQueue(variable.get()); waitingRoom.remove(variable.get());

passengerCount.getAndIncrement(); variableBtn.get().setDisable(true);

variableBtn.get().setStyle("-fx-background-color: #1D8348 "); variableBtn.set(null);

btn\_add.setDisable(true); btn\_delete.setDisable(true);

if (passengerCount.get()==rand\_no){gui.close();}

});

btn\_delete.setOnAction(event -> {

pasBtn[passengerCount.get()].setStyle("-fx-background-color: #A93226 "); waitingRoom.remove(variable.get());

passengerCount.getAndIncrement(); variableBtn.get().setDisable(true);

variableBtn.get().setStyle("-fx-background-color: #A93226 "); variableBtn.set(null);

btn\_add.setDisable(true); btn\_delete.setDisable(true);

if (passengerCount.get()==rand\_no){gui.close();}

});

waitingroom.setLayoutY(30); waitingroom.setLayoutX(215); passengerQueue.setLayoutY(30); passengerQueue.setLayoutX(570);

guiAnchorpane.getChildren().addAll(firstPane, secondPane, waitingroom, passengerQueue, btn\_add, btn\_delete, btn\_ok);

Scene scene = new Scene(guiAnchorpane, 900, 950); gui.setScene(scene);

gui.setTitle("Waitiing Room"); gui.showAndWait();

}else {

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\* No Data to view! \*\*\*\*\*\*\*\*\*\*\*\*\*"); System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

}



## viewTrainQueue ( ) method

private static void viewTrainQueue() { Stage gui = new Stage();

AnchorPane guiAnchorpane = new AnchorPane(); guiAnchorpane.setStyle("-fx-background-color: #17202A "); GridPane firstPane = new GridPane();

firstPane.setStyle("-fx-background-color: #17202A "); Label pasengerQueue = new Label("Passenger Queue"); pasengerQueue.setStyle("-fx-text-fill: #F7F9F9 "); Button[] btn = new Button[42]; firstPane.setPrefSize(900, 800); firstPane.setLayoutX(50);

firstPane.setLayoutY(75); firstPane.setHgap(15); firstPane.setVgap(15);

PassengerQueue trainQueue = new PassengerQueue(); Passenger[] myArray ;

myArray = trainQueue.getQueueArray(); int count = 0;

while (count < 42) { //create the check boxes if(myArray[count] != null){

btn[count] = new Button("Name : "+myArray[count].getName()+"\nSeat No : "+myArray[count].getSeat());

btn[count].setStyle("-fx-background-color: #F1C40F ");

}

else{

btn[count] = new Button("Empty"); btn[count].setStyle("-fx-background-color: #c70000 ");

//btn[count].setDisable(true) } btn[count].setPrefSize(195, 180); count++;

}

int num = 0;

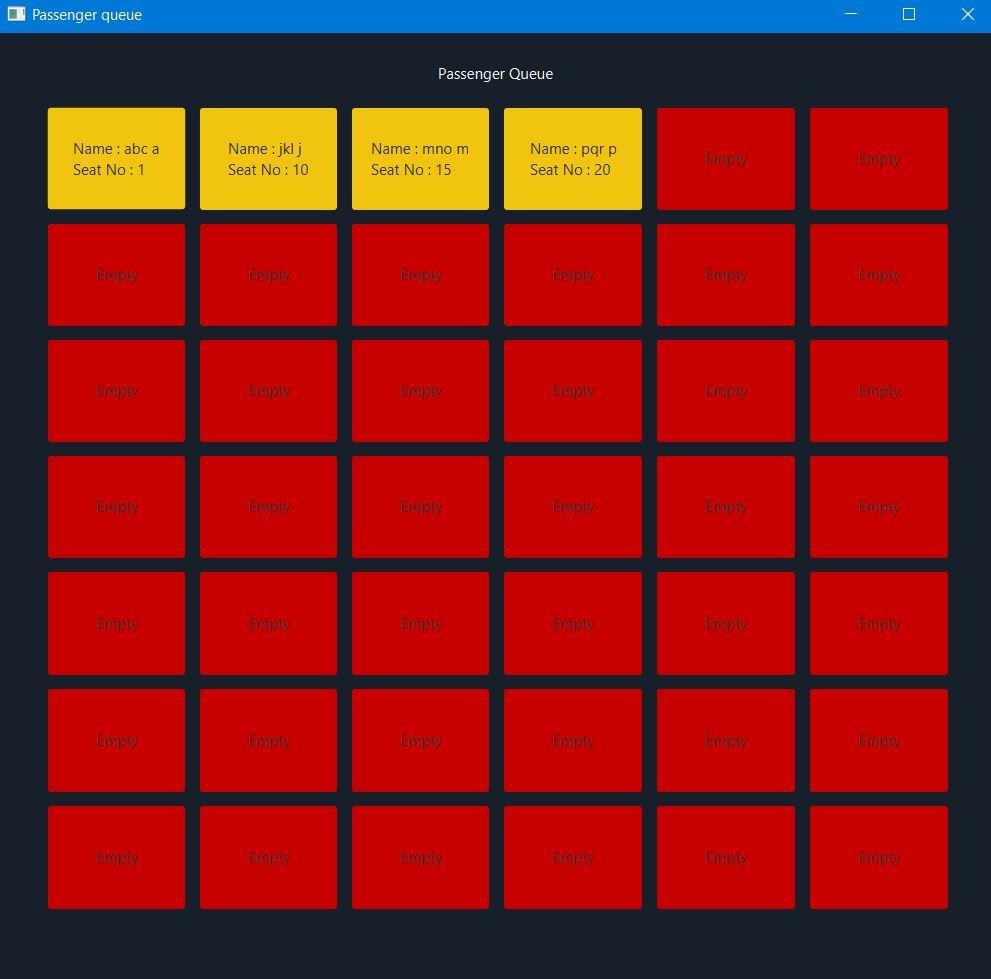
for (int i = 0; i < 7; i++) { for (int j = 0; j < 6; j++) {

firstPane.add(btn[num++], j, i);//allign the created checkboxes in GridPane.}} pasengerQueue.setLayoutY(30);

pasengerQueue.setLayoutX(440); guiAnchorpane.getChildren().addAll(firstPane,pasengerQueue); Scene scene = new Scene(guiAnchorpane, 1000, 950); gui.setScene(scene);

gui.setTitle("Passenger queue"); gui.showAndWait();

}



## deletePassenger ( ) method

private static void deletePassenger() {

PassengerQueue trainQueue = new PassengerQueue(); Scanner sc = new Scanner(System.in); System.out.println("Seat No : ");

int no = sc.nextInt(); try {

for (int i = 0; i < 42; i++) {

if (trainQueue.getQueueArray()[i].getSeat() == no) { trainQueue.remove(i);

break;

}

}

for (int x = 0; x < 41; x++) {

if (trainQueue.getQueueArray()[x] == null) { trainQueue.getQueueArray()[x] = trainQueue.getQueueArray()[x + 1]; trainQueue.getQueueArray()[x + 1] = null;

}

}

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Deleted succesfull!\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

} catch (Exception e) { System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\* No Seats Detected \*\*\*\*\*\*\*\*\*\*\*\*\*"); System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

}

## SaveData ( ) method

private static void saveData() {

MongoClient mongoClient = MongoClients.create("mongodb://LocalHost:27017"); MongoDatabase database = mongoClient.getDatabase("TrainBookingxx"); MongoCollection<Document> collection = database.getCollection("Ticketsxx"); BasicDBObject document = new BasicDBObject(); collection.deleteMany(document); //delete previous datas

for(Passenger passenger : PassengerQueue.getQueueArray()){ if (passenger != null){

Document record = new Document("Title","name")

.append("name",passenger.getName())

.append("seat",passenger.getSeat()); collection.insertOne(record);

}

}

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); System.out.println("\*\*\*\*\*\*\*\*\*\*\*\* Succesfully Stored! \*\*\*\*\*\*\*\*\*\*\*\*"); System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

## loadData ( ) method

private static void loadData() {

MongoClient mongoClient = MongoClients.create("mongodb://LocalHost:27017"); MongoDatabase database = mongoClient.getDatabase("TrainBookingxx"); MongoCollection<Document> collection = database.getCollection("Ticketsxx"); BasicDBObject document = new BasicDBObject();

FindIterable<Document> data = collection.find(); PassengerQueue passenger = new PassengerQueue(); for (Document record : data) {

String name = record.get("name").toString();

int seat = Integer.parseInt(record.get("seat").toString());

//System.out.println(Integer.toString(seat)); Passenger obj = new Passenger(); obj.setName(name,"");

obj.setSeat(seat); PassengerQueue.addToQueue(obj); for(Passenger passenger1 : waitingRoom){

if(passenger1.getSeat() == seat){ waitingRoom.remove(passenger1); break;

}

}

}

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); System.out.println("\*\*\*\*\*\*\*\*\*\*\*\* Loading Succesfull! \*\*\*\*\*\*\*\*\*\*\*\*"); System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

## simulation ( ) method

private void simulation() {

PassengerQueue trainQueue = new PassengerQueue(); Passenger[] myArray ;

myArray = trainQueue.getQueueArray(); for(int a = 0 ; a<42 ; a++){

if(myArray[a] != null){ temp.add(myArray[a]);}}

if(temp.size() != 0) {

Random rand = new Random(); int queueLength = 0;

for (int x = 0; x < 42; x++) {

int dice\_01 = rand.nextInt(6) + 1; int dice\_02 = rand.nextInt(6) + 1; int dice\_03 = rand.nextInt(6) + 1;

int dice\_total = dice\_01 + dice\_02 + dice\_03; if (myArray[x] != null) { myArray[x].setSecondsInQueue(dice\_total); totalTime += dice\_total; myArray[x].setTotalSeconds(totalTime); queueLength++;

if (maximumWaitingTime < dice\_total) { maximumWaitingTime = dice\_total;}

if (minimumWaitingTime > dice\_total) { minimumWaitingTime = dice\_total;}}} try {

if (queueLength > maxQueueLength) { maxQueueLength = queueLength; trainQueue.setMaxLength(maxQueueLength);} averageTime = totalTime / temp.size();

Stage gui = new Stage();

AnchorPane guiAnchorpane = new AnchorPane(); guiAnchorpane.setStyle("-fx-background-color: #17202A "); GridPane firstPane = new GridPane();

firstPane.setStyle("-fx-background-color: #17202A "); Label pasengerQueue = new Label("Train Board"); pasengerQueue.setStyle("-fx-text-fill: #F7F9F9 "); Button[] btn = new Button[42]; firstPane.setPrefSize(900, 800); firstPane.setLayoutX(50);

firstPane.setLayoutY(75); firstPane.setHgap(15); firstPane.setVgap(15); int count = 0;

for(int i = 0 ; i<temp.size() ; i++) { //create the check boxes

btn[count] = new Button("Name : " + temp.get(count).getName() + "\nSeat No : " + temp.get(count).getSeat() + "\nProcessing Time: "

+ temp.get(count).getSecondsInQueue()+"\nWaiting time: "+temp.get(count).getTotalSeconds());

//btn[count].setDisable(true);

btn[count].setStyle("-fx-background-color: #F1C40F "); btn[count].setPrefSize(195, 180);

count++;}

int buttons = 0; int num = 0;

for (int i = 0; i < 7; i++) { for (int j = 0; j < 6; j++) {

firstPane.add(btn[num++], i, j);//allign the created checkboxes in GridPane. buttons++;

if(buttons == temp.size()){ break;}}if(buttons == temp.size()) {

break;}}

Label myTxt = new Label("Maximum length of Queue : " + trainQueue.getMaxLength()+"\nMaximum Waiting time : " + maximumWaitingTime+"\nMinimum Waiting time : "

+ minimumWaitingTime +"\nAverage Waiting time : " + averageTime); myTxt.setStyle("-fx-text-fill: #F7F9F9 ");

myTxt.setLayoutX(910); myTxt.setLayoutY(300); pasengerQueue.setLayoutY(30); pasengerQueue.setLayoutX(440);

guiAnchorpane.getChildren().addAll(firstPane, pasengerQueue,myTxt); Scene scene = new Scene(guiAnchorpane, 1200, 950); gui.setScene(scene);

gui.setTitle("Train Board"); System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); System.out.println("\*\*\*Your program is in process,Wait " + maximumWaitingTime + " seconds\*\*\*"); System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); TimeUnit.SECONDS.sleep(maximumWaitingTime);

gui.showAndWait()

File txtFile = new File("trainBoard"); if (!txtFile.exists()) { txtFile.createNewFile();}

PrintWriter writer = new PrintWriter(txtFile); for (int i = 0; i < temp.size(); i++) {

writer.println("Name : " + temp.get(i).getName()); writer.println("Seat No : " + temp.get(i).getSeat());

writer.println("Processing time : " + temp.get(i).getSecondsInQueue()); writer.println("Waiting time : " + temp.get(i).getTotalSeconds()); writer.println("");} writer.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

writer.println("Maximum length of Queue : " + trainQueue.getMaxLength());

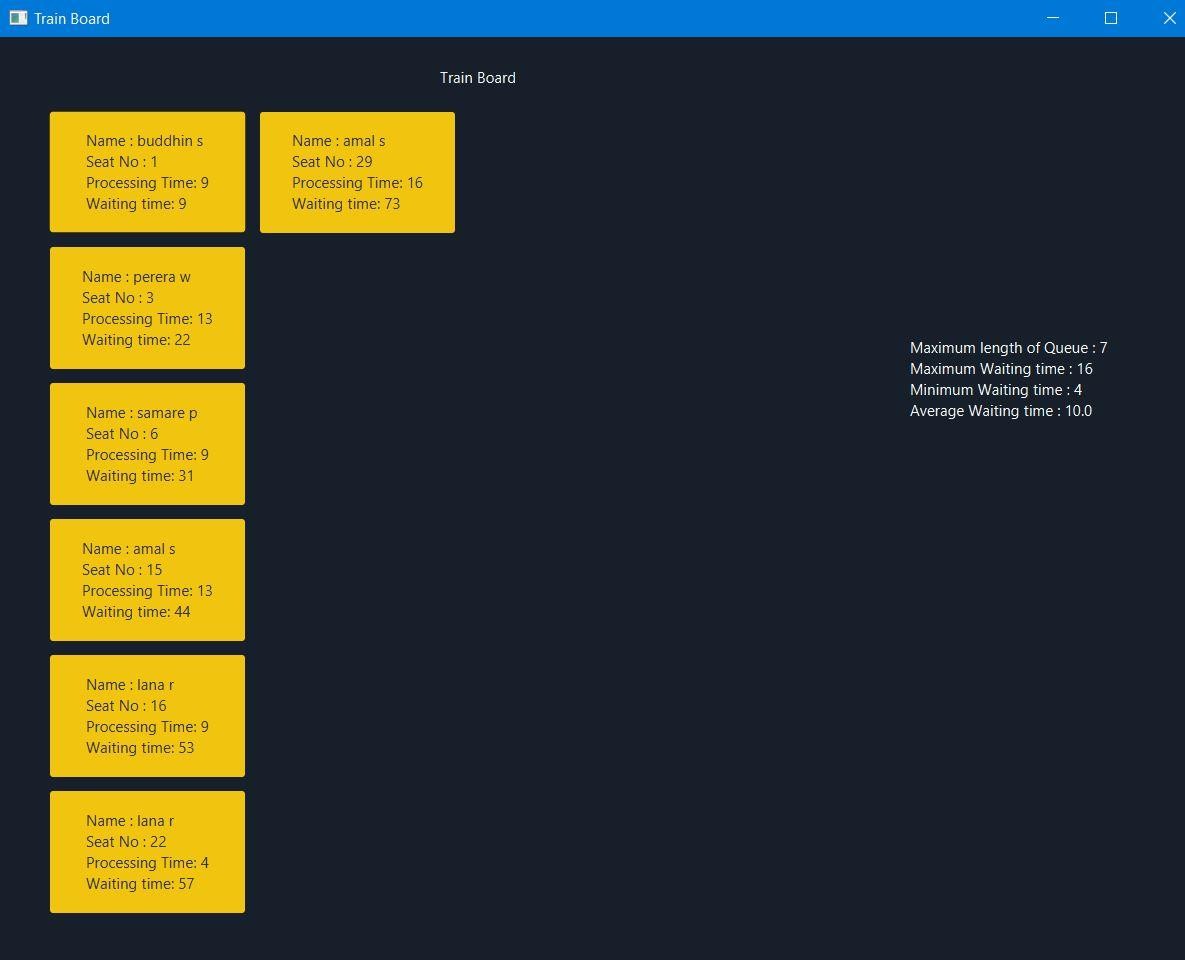
writer.println("Maximum Waiting time : " + maximumWaitingTime); writer.println("Minimum Waiting time : " + minimumWaitingTime); writer.println("Average Waiting time : " + averageTime); writer.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); writer.close();

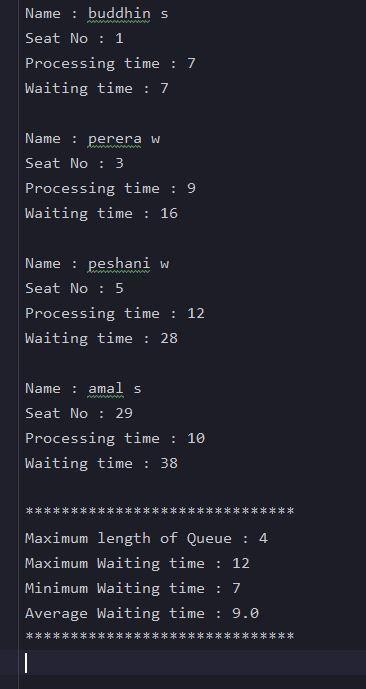
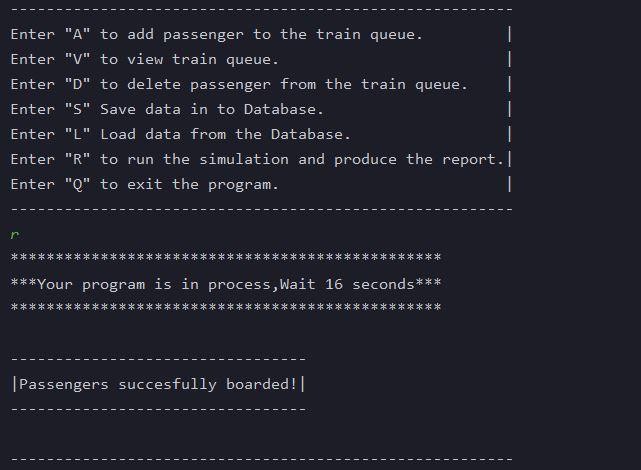
} catch (Exception e) {

System.out.println("An error occured,Please try again.");} for (int i = 0; i < 42; i++) {

myArray[i] = null;}

}else { System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*No Passengers to board\*\*\*\*\*\*\*\*\*\*\*\*\*"); System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");} trainQueue.setLast(0);}





# Passenger Class

package com.trainStation;

public class Passenger { private String name;

private int secondsInQueue; private int seat;

private int totalSeconds; public int getTotalSeconds() { return totalSeconds;}

public void setTotalSeconds(int totalSeconds) { this.totalSeconds = totalSeconds;

}

public int getSeat() { return seat;

}

public void setSeat(int seat) { this.seat = seat;

}

public String getName() { return name;

}

public void setName(String firstName,String surName) { this.name = firstName+" "+surName;}

public int getSecondsInQueue() { return secondsInQueue;

}

public void setSecondsInQueue(int secondsInQueue) { this.secondsInQueue = secondsInQueue;

}

}

# PassengerQueue Class

public class PassengerQueue {

private static Passenger [] queueArray = new Passenger[42]; private static int maxLength;

private static int last;

public static Passenger[] getQueueArray() { return queueArray;

}

public void setQueueArray() { this.queueArray = queueArray;

}

public void setLast(int last) { this.last = last;

}

public static void addToQueue(Passenger passenger){ queueArray[last]=passenger;

last = last + 1;

}

public static Passenger remove(int seatNo){ last = last - 1 ;

return queueArray[seatNo] = null;} public static boolean isFull(){ return false;}

public static void setMaxLength(int maxLength) { PassengerQueue.maxLength = maxLength;} public static int getMaxLength() {

return maxLength;

}}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Test input** | **Expected result** | **Actual result** | **Pass/Fail** |
| 01 | Option “A” or “a” | addPassenger ( ) | addPassenger ( ) | Pass |
| 02 | Option “V” or “v” | viewTrainQueue ( ) | viewTrainQueue ( ) | Pass |
| 03 | Option “D” or “d” | deletePassenger ( ) | deletePassenger ( ) | Pass |
| 04 | Option “S” or “s” | saveData ( ) | saveData ( ) | Pass |
| 05 | Option “L” or “l” | loadData ( ) | loadData ( ) | Pass |
| 06 | Option “R” or “r” | simulation ( ) | simulation ( ) | Pass |
| 07 | Option “Q” or “q” | quit ( ) | quit ( ) | Pass |
| 08 | Any numerical value or letters except A,V,D,S,L,R or Q | Invalid input | Invalid input | Pass |
| addPassenger ( ) method | | | | |
| 09 | If no data in database | “No data” | “No data” | Pass |
| 10 | If passenger Queue is full | “Passenger queue is full” | “Passenger queue is full” | Pass |
| deletePassenger ( ) method | | | | |
| 11 | Seat no < 42 and correct booked seat no | “Deleted succefull” | “Deleted succefull” | Pass |
| 12 | Seat no < 42 but selected incorrect seat no | No seats detected | No seats detected | Pass |
| 13 | Seat no > 42 | Please enter correct seat number | Please enter correct seat number | Pass |