  
  
**Assessment Cover Sheet**

|  |  |
| --- | --- |
| **Subject Code:** | CSIT314 |
| **Subject Name:** | Software Development Methodologies |
| **Submission Type:** | Deliverable 4 - Report |
| **Assignment Title:** | Deliverable 4 |
| **Student Name:** | Simran Patt, Ocean Dsouza, Muhammad Uzair Naeem, Saim Ali |
| **Student Number:** | 7009690, 6628485, 6538654, 6482491 |
| **Student Phone/Mobile No.** | 0545762036,0503605417,0563249747, 050519211 |
| **Student E-mail:** | scp106@uowmail.edu.au ,osd331@uowmail.edu.au,nm690@uowmail.edu.au, sa122@uowmail.edu.au |
| **Lecturer Name:** | Dr.Abdelghani Benharref |
| **Due Date:** | 10-06-2022 |
| **Date Submitted:** | 10-06-2022 |

|  |  |
| --- | --- |
| **PLAGIARISM:** The penalty for deliberate plagiarism is FAILURE in the subject. Plagiarism is cheating by using the written ideas or submitted work of someone else. UOWD has a strong policy against plagiarism.  The University of Wollongong in Dubai also endorses a policy of non-discriminatory language practice and presentation.  **PLEASE NOTE:**STUDENTS MUST RETAIN A COPY OF ANY WORK SUBMITTED | **DECLARATION:** I/We certify that this is entirely my/our own work, except where I/we have given fully-documented references to the work of others, and that the material contained in this document has not previously been submitted for assessment in any formal course of study. I/we understand the definition and consequences of plagiarism.  **Signature of Student:** |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Optional Marks:** | | **Comments:** | |

https://my.uowdubai.ac.ae/images/scissors.gif

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Lecturer Assignment Receipt**(To be filled in by student and retained by Lecturer upon return of assignment) | | | **Subject:** | **Assessment Title:** | | **Student Name:** | **Student Number:** | | **Due Date:** | **Date Submitted:** | | **Signature of Student:** | | |

https://my.uowdubai.ac.ae/images/scissors.gif

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Student Assignment Receipt** (To be filled in and retained by Student upon submission of assignment) | | | **Subject:** | **Assessment Title:** | | **Student Name:** | **Student Number:** | | **Due Date:** | **Date Submitted:** | | **Signature of Lecturer** | | |

*Table of Contents*

[Java Class Code 4](#_Toc105801044)

[Card Class 4](#_Toc105801045)

[Customer Class 5](#_Toc105801046)

[ElderlyCard Class 6](#_Toc105801047)

[PeopleOfDeterminationCard Class 7](#_Toc105801048)

[Staff Class 8](#_Toc105801049)

[StudentCard Class 9](#_Toc105801050)

[Trips 10](#_Toc105801051)

[Main Class 11](#_Toc105801052)

[Register Staff function (only Admin) 12](#_Toc105801053)

[Code 12](#_Toc105801054)

[Control Flow Graph (CFG) 14](#_Toc105801055)

[Def and Use Analysis 14](#_Toc105801056)

[Test Cases 14](#_Toc105801057)

[All nodes 15](#_Toc105801058)

[All conditions 15](#_Toc105801059)

[All paths 15](#_Toc105801060)

[All independent paths 15](#_Toc105801061)

[Registering New Card for customer and adding in customerDB 15](#_Toc105801062)

[Control Flow Graph (CFG) 20](#_Toc105801063)

[Def and Use Analysis 21](#_Toc105801064)

[Test Cases 21](#_Toc105801065)

[All nodes 21](#_Toc105801066)

[All conditions 29](#_Toc105801067)

[All paths 29](#_Toc105801068)

[Conditional coverage 37](#_Toc105801069)

[Delete User from Database 38](#_Toc105801070)

[Control Flow Graph (CFG) 40](#_Toc105801071)

[Def and Use Analysis 40](#_Toc105801072)

[Cases 41](#_Toc105801073)

[All nodes 41](#_Toc105801074)

[All conditions 43](#_Toc105801075)

[All paths 43](#_Toc105801076)

[All Independent Paths 45](#_Toc105801077)

[Team Distribution Sheet – Deliverable 03 47](#_Toc105801078)

# Java Class Code

## Card Class

package com.company;

public class Card

{

//generated Tag Number

private long tagNumber;

//This will be the hashed Tag number

private String Key;

//Card Balance

private int Balance;

//Card points

private int DailyTravelPoints;

//type of card

private String Type;

public Card(){};

public Card(long tagNumber, String key, int balance, int dailyTravelPoints, String type) {

this.tagNumber = tagNumber;

Key = key;

Balance = balance;

DailyTravelPoints = dailyTravelPoints;

Type = type;

}

//Getters

public long getTagNumber() { return tagNumber; }

public String getKey() { return Key; }

public int getBalance() {return Balance; }

public int getDailyTravelPoints() { return DailyTravelPoints; }

public String getType() {return Type;}

//Setters

public void setTagNumber(long tagNumber) { this.tagNumber = tagNumber;}

public void setKey(String key) {Key = key;}

public void setBalance(int balance) { Balance = balance;}

public void setDailyTravelPoints(int dailyTravelPoints) {DailyTravelPoints = dailyTravelPoints; }

public void setType(String type) {Type = type; }

//toString

public String Display() {

return "Card{" +

"tagNumber=" + tagNumber +

", Key='" + Key + '\'' +

", Balance=" + Balance +

", DailyTravelPoints=" + DailyTravelPoints +

", Type='" + Type + '\'' +

'}';

}

}

## Customer Class

package com.company;

import java.util.ArrayList;

import java.util.List;

public class Customer {

private String Name;

private String email;

private String EID;

private String Address;

private String Biometrics;

private int BackupPin;

private Card cards;

private List<Trips> l1;

//Parameterized constructor

//We will only fill The name,email,EID, Address list of cards and list of trips, As biometrics will be done by another function.

public Customer(String name, String email, String EID, String address, Card cards) {

Name = name;

this.email = email;

this.EID = EID;

Address = address;

this.cards = cards;

}

//Getters and Setters

public String getName() {return Name;}

public void setName(String name) { Name = name; }

public String getEmail() { return email; }

public void setEmail(String email) {this.email = email;}

public String getEID() {return EID;}

public void setEID(String EID) {this.EID = EID; }

public String getAddress() {return Address; }

public void setAddress(String address) { Address = address;}

public String getBiometrics() { return Biometrics;}

public void setBiometrics(String biometrics) {Biometrics = biometrics;}

public int getBackupPin() {return BackupPin;}

public void setBackupPin(int backupPin) {BackupPin = backupPin;}

public Card getCards() { return cards; }

public void setCards(Card cards) { this.cards = cards; }

public List<Trips> getL1() { return l1;}

public void setL1(List<Trips> l1) {this.l1 = l1;}

@Override

public String toString() {

return "Customer{" +

"Name='" + Name + '\'' +

", email='" + email + '\'' +

", EID='" + EID + '\'' +

", Address='" + Address + '\'' +

", Biometrics='" + Biometrics + '\'' +

", BackupPin=" + BackupPin +

", cards=" + cards +

", l1=" + l1 +

'}';

}

}

## ElderlyCard Class

package com.company;

public class ElderlyCard extends Card

{

private int Discount\_rate;

private String EID;

//Default and Parameterized constructors

public ElderlyCard(int discount\_rate, long EID) {

Discount\_rate = discount\_rate;

EID = EID; }

public ElderlyCard(long tagNumber, String key, int balance, int dailyTravelPoints, String type, int discount\_rate, String EID) {

super(tagNumber, key, balance, dailyTravelPoints, type);

Discount\_rate = discount\_rate;

EID = EID;}

//Getters

public int getDiscount\_rate() { return Discount\_rate; }

public String getEID() {return EID; }

//Setter

public void setDiscount\_rate(int discount\_rate) {Discount\_rate = discount\_rate; }

public void setEID(long EID) {EID = EID; }

//toString

@Override

public String Display() {

return "StudentCard{" +

"Discount\_rate=" + Discount\_rate +

", EID=" + EID +

'}';

}

}

## PeopleOfDeterminationCard Class

package com.company;

public class People\_of\_Determination\_Card extends Card

{

private int Discount\_rate;

private long PODno;

//Default and Parameterized constructors

public People\_of\_Determination\_Card(int discount\_rate, long PODno) {

Discount\_rate = discount\_rate;

PODno = PODno; }

public People\_of\_Determination\_Card(long tagNumber, String key, int balance, int dailyTravelPoints, String type, int discount\_rate, long PODno) {

super(tagNumber, key, balance, dailyTravelPoints, type);

Discount\_rate = discount\_rate;

PODno = PODno; }

//Getters

public int getDiscount\_rate() { return Discount\_rate; }

public long getPODno() { return PODno; }

//Setter

public void setDiscount\_rate(int discount\_rate) { Discount\_rate = discount\_rate; }

public void setPODno(long PODno) { PODno = PODno; }

//toString

@Override

public String Display() {

return "StudentCard{" +

"Discount\_rate=" + Discount\_rate +

", PODno=" + PODno +

'}';

}

}

## Staff Class

package com.company;

import java.util.Date;

public class Staff

{

private String name;

private Date DOB;

private String nationality;

private String EID;

private long Contact\_Num;

private String ID;

//default constructor

public Staff(){};

//Parameterized constructors

public Staff(String name, Date DOB, String nationality, String EID, long contact\_Num, String ID)

{

this.name = name;

this.DOB = DOB;

this.nationality = nationality;

this.EID = EID;

Contact\_Num = contact\_Num;

this.ID = ID;

}

//getters

public String getName() {return name; }

public Date getDOB() {return DOB;}

public String getNationality() { return nationality; }

public String getEID() { return EID; }

public long getContact\_Num() { return Contact\_Num; }

public String getID() { return ID; }

//setters

public void setName(String name) {this.name = name; }

public void setDOB(Date DOB) { this.DOB = DOB; }

public void setNationality(String nationality) { this.nationality = nationality;}

public void setEID(String EID) { this.EID = EID; }

public void setContact\_Num(long contact\_Num) { Contact\_Num = contact\_Num; }

public void setID(String ID) { this.ID = ID; }

public String Display() {

return "Staff{" +

"name='" + name + '\'' +

", DOB=" + DOB +

", nationality='" + nationality + '\'' +

", EID='" + EID + '\'' +

", Contact\_Num=" + Contact\_Num +

", ID=" + ID +

'}';

}

}

## StudentCard Class

package com.company;

public class StudentCard extends Card

{

private int Discount\_rate;

private long StudentID;

//Default and Parameterized constructors

public StudentCard(int discount\_rate, long studentID) {

Discount\_rate = discount\_rate;

StudentID = studentID; }

public StudentCard(long tagNumber, String key, int balance, int dailyTravelPoints, String type, int discount\_rate, long studentID) {

super(tagNumber, key, balance, dailyTravelPoints, type);

Discount\_rate = discount\_rate;

StudentID = studentID; }

//Getters

public int getDiscount\_rate() { return Discount\_rate }

public long getStudentID() { return StudentID; }

//Setter

public void setDiscount\_rate(int discount\_rate) { Discount\_rate = discount\_rate }

public void setStudentID(long studentID) { StudentID = studentID; }

//toString

@Override

public String Display() {

return "StudentCard{" +

"Discount\_rate=" + Discount\_rate +

", StudentID=" + StudentID +

'}';

}

}

## Trips

private String Trip\_Cost;

//default constructor

public Trips (){};

//Parameterized constructor

public Trips(String tagIn, String tagOut, String location\_Tagin, String location\_Tagout, String trip\_Cost) {

TagIn = tagIn;

TagOut = tagOut;

Location\_Tagin = location\_Tagin;

Location\_Tagout = location\_Tagout;

Trip\_Cost = trip\_Cost; }

//Getters

public String getTagIn() { return TagIn; }

public String getTagOut() { return TagOut; }

public String getLocation\_Tagin() { return Location\_Tagin; }

public String getLocation\_Tagout() { return Location\_Tagout; }

public String getTrip\_Cost() { return Trip\_Cost; }

//Setters

public void setTagIn(String tagIn) { TagIn = tagIn; }

public void setTagOut(String tagOut) { TagOut = tagOut; }

public void setLocation\_Tagin(String location\_Tagin) { Location\_Tagin = location\_Tagin; }

public void setLocation\_Tagout(String location\_Tagout){ Location\_Tagout = location\_Tagout; }

public void setTrip\_Cost(String trip\_Cost) { Trip\_Cost = trip\_Cost; }

//Tostring

@Override

public String toString() {

return "Trips{" +

"TagIn='" + TagIn + '\'' +

", TagOut='" + TagOut + '\'' +

", Location\_Tagin='" + Location\_Tagin + '\'' +

", Location\_Tagout='" + Location\_Tagout + '\'' +

", Trip\_Cost='" + Trip\_Cost + '\'' +

'}';

}

}

## Main Class

package com.company;

import javax.crypto.KeyGenerator;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.\*;

import java.util.regex.Pattern;

public class Main {

**//Code for 3 functions are below.**

public static void main(String[] args) throws ParseException, NoSuchAlgorithmException {

//This is staff DB

ArrayList<Staff> staffdb=new ArrayList<>();

//Registering Staff to the Database

RegisterStaff(staffdb);

//This is the Customer Database

ArrayList<Customer> cusdb=new ArrayList<>();

//Registering card to customer database

RegisterNewCard(cusdb);

}

}

# Register Staff function (only Admin)

## Code

public static void RegisterStaff(ArrayList<Staff> StaffDB) throws ParseException {

//Header

System.out.println("\*\*\* Welcome Admin! \*\*\*");

System.out.println("Fill in the following fields to register a Staff member");

//declaration of variables

String name;

Date DOB;

String temp;

String nationality;

String EID;

String temp2;

long Contact\_Num;

String ID;

SimpleDateFormat formatter = new SimpleDateFormat("dd-MM-yyyy", Locale.ENGLISH);

//Reading Admin input from keyboard

Scanner sc = new Scanner(System.in);

try {

System.out.print("Name: ");

name = sc.next();

//checking if name field is empty , equals " ", if the length of the name input is greater than one,

//and if it does not match Regex pattern

if (name.isEmpty() || name.equals(" ") || name.length() <= 1 || Pattern.matches("[a-zA-Z]\*",name)==false)

{

System.out.println("Name field filled incorrectly!");

return;

}

//checking if date is empty, or is qual to " " and if it is in a specified format

System.out.print("Date of Birth (dd-mm-yyyy): ");

temp = sc.next();

if (name.isEmpty() || name.equals(" ") || Pattern.matches("^([0]?[1-9]|[1|2][0-9]|[3][0|1])[.\\/-]([0]?[1-9]|[1][0-2])[.\\/-]([0-9]{4}|[0-9]{2})$",temp)==false) {

System.out.println("Date field filled incorrectly!");

return;

}

DOB = formatter.parse(temp);

//checking if Nationality field is empty or equals " " and if if nationality is all text

System.out.print("Nationality: ");

nationality = sc.next();

if (nationality.isEmpty() || nationality.equals(" ")||Pattern.matches("[a-zA-Z]\*",nationality)==false)

{

System.out.println("nationality field filled incorrectly!");

return;

}

//checking if EID field is empty or equals " " or if its length is greather than 15 as uae EID is 15 digits long

System.out.print("Emirates ID(784-\*\*\*\*-\*\*\*\*\*\*\*-\*): ");

EID = sc.next();

if (EID.isEmpty() || EID.equals(" ") ||Pattern.matches("^784-[0-9]{4}-[0-9]{7}-[0-9]{1}$",EID)==false) {

System.out.println("Emirates ID field filled incorrectly!");

return;

}

//Checking if contact number is empty, or equals to " " or if it is in a specified format

System.out.print("Contact Number (+971 \*\*\*\*\*\*\*\*\*): ");

temp2 = sc.next();

if (temp2.isEmpty() || temp2.equals(" ") ||Pattern.matches("[0-9]{9}",temp2)==false) {

System.out.println("Contact Number field filled incorrectly!");

return;

}

Contact\_Num = Long.parseLong(temp2);

//Here we will generate the ID number for the staff

ID="00"+Integer.toString(StaffDB.size()+1);

//Since all the feilds have been validated and information is correct we will add to database

//Note: In this case the array list is the Database

//Firstly we will create object of type Staff

Staff s1=new Staff( name, DOB, nationality, EID, Contact\_Num, ID);

//We are adding the staff to the Database

StaffDB.add(s1);

System.out.println(s1.Display()+'\n'+"Sucessfully added to database !");

} catch (Exception f) {

System.out.println("Errors have occured try again!");

}

}

//Generate Random number will be done for Registering a new card

public static long GenerateTagnumber(){

//Declaration of variables

int number;

String temp;

long TagNumber;

//Randdom will allow us to generate sudo-random numbers

Random r1=new Random();

//String array to store the 10 random numbers in format string

String [] randArr=new String[10];

//For loop we will generate a random number in each iteration then store it in the string array

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

number = r1.nextInt(10);

randArr[i] = Integer.toString(number);

}

//Concatinate all the values of the array into one string

temp=randArr[0]+randArr[1]+randArr[2]+randArr[3]+randArr[4]+randArr[5]+randArr[6]+randArr[7]+randArr[8]+randArr[9];

//Convert the above string into Long

TagNumber=Long.parseLong(temp);

return TagNumber;

}

//Converting Tag number to Sha256 hash

public static String Keygen(String input) throws NoSuchAlgorithmException {

//hash the Tag number using Sha256 using MessageDigest

MessageDigest md = MessageDigest.getInstance("SHA-256");

md.update(input.getBytes());

//String the hash in a byte Array

byte[] digest = md.digest();

//Converting the Byte Array to string uisng stringBuffer

StringBuffer sb = new StringBuffer();

//For each byte in the Array convert to String

for (byte b : digest){

sb.append(String.format("%02x", b & 0xff));

}

//return String or hashed value

return sb.toString();

}

## Control Flow Graph (CFG)

## Def and Use Analysis

## Test Cases

### All nodes

### All conditions

### All paths

### All independent paths

# Registering New Card for customer and adding in customerDB

public static void RegisterNewCard(ArrayList<Customer> CustomerDB) throws NoSuchAlgorithmException {

//Declaring variables to create object of type Card

long tagNumber;

String key;

int balance;

int dailyTravelPoints;

long studID;

long PODno;

String type;

//Declaring variables to create object of type Customer

String cname;

String EID;

String Email;

String Adress;

//Scanner to take user input

Scanner sc=new Scanner(System.in);

System.out.println("\*\*\* Welcome Staff / Admin! \*\*\*");

System.out.println("Fill in the following field's to register a new card");

//Allocating Values to declarations

tagNumber=GenerateTagnumber();

key=Keygen(String.valueOf(tagNumber));

balance=20;//Strandard Balnace that comes with all Nol cards

dailyTravelPoints=0;//Initially travle points will be 0 user has to earn them

//Here we will take input from user about what type of card he needs to create

System.out.print("Please specify the Type of the Card (Student,Elderly,People of determination or Normal): ");

type= sc.next();

//Here we will validate the input to check if it is empty or " " or if it does not fall under the four categories Student,Elderly,POD and Normal.

if (type.isEmpty()||type.equals(" "))

{

System.out.println("Type field filled incorrectly!");

}

//If the Card type is student

if (type.equalsIgnoreCase("student"))

{

//Take the customers name and Validate it to check if empty or " " and by REGEX pattern

System.out.print("Customer Name: ");

cname= sc.next();

if (cname.isEmpty() || cname.equals(" ") || cname.length() <= 1 || Pattern.matches("[a-zA-Z]\*",cname)==false)

{

System.out.println("Customer Name field filled incorrectly!");

return;

}

//Take the Customers EID and Validate it to check if empty, " " and by REGEX pattern

System.out.print("Customer EID: ");

EID= sc.next();

if (EID.isEmpty() || EID.equals(" ") ||Pattern.matches("^784-[0-9]{4}-[0-9]{7}-[0-9]{1}$",EID)==false) {

System.out.println("Emirates ID field filled incorrectly!");

return;

}

//Take the Customers Email and Validate it to check if empty, " " and by REGEX pattern

System.out.print("Customer Email: ");

Email= sc.next();

if (Email.isEmpty() || Email.equals(" ") ||Pattern.matches("[a-z0-9]+@[a-z]+\\.[a-z]{2,3}",Email)==false) {

System.out.println("Email field filled incorrectly!");

return;

}

//Take the Customers Adress and Validate it to check if empty, " " and by REGEX pattern

System.out.print("Customer Address: ");

Adress= sc.next();

if (Adress.isEmpty() || Adress.equals(" ") ||Pattern.matches("[a-zA-Z]\*",Adress)==false) {

System.out.println("Address field filled incorrectly!");

return;

}

//Take the Student ID and Validate it to check if empty, " " and by REGEX pattern

System.out.print("Customer StudentID: ");

studID= sc.nextLong();

if (studID<=0 ||Pattern.matches("[0-9]{5}",String.valueOf(studID))==false) {

System.out.println("Student ID field filled incorrectly!");

return;

}

//Create object of type student card

StudentCard s1=new StudentCard(tagNumber, key, balance, dailyTravelPoints, type,50,studID);

//Creat obejct of type Customer

Customer c1=new Customer(cname,Email,EID,Adress,s1);

//Display customer

System.out.println(c1.toString());

//Adding customer to the "Database"

CustomerDB.add(c1);

System.out.println("Customer added to Database sucessfully !");

}

else if(type.equalsIgnoreCase("People of determination"))

{ //Take the customers name and Validate it to check if empty or " " and by REGEX pattern

System.out.print("Customer Name: ");

cname= sc.next();

if (cname.isEmpty() || cname.equals(" ") || cname.length() <= 1 || Pattern.matches("[a-zA-Z]\*",cname)==false)

{

System.out.println("Customer Name field filled incorrectly!");

return;

}

//Take the Customers EID and Validate it to check if empty, " " and by REGEX pattern

System.out.print("Customer EID: ");

EID= sc.next();

if (EID.isEmpty() || EID.equals(" ") ||Pattern.matches("^784-[0-9]{4}-[0-9]{7}-[0-9]{1}$",EID)==false) {

System.out.println("Emirates ID field filled incorrectly!");

return;

}

//Take the Customers Email and Validate it to check if empty, " " and by REGEX pattern

System.out.print("Customer Email: ");

Email= sc.next();

if (Email.isEmpty() || Email.equals(" ") ||Pattern.matches("[a-z0-9]+@[a-z]+\\.[a-z]{2,3}",Email)==false) {

System.out.println("Email field filled incorrectly!");

return;

}

//Take the Customers Adress and Validate it to check if empty, " " and by REGEX pattern

System.out.print("Customer Address: ");

Adress= sc.next();

if (Adress.isEmpty() || Adress.equals(" ") ||Pattern.matches("[a-zA-Z]\*",Email)==false) {

System.out.println("Address field filled incorrectly!");

return;

}

//Take the Customers People of Determination Number and Validate it to check if empty, " " and by REGEX pattern

System.out.print("Customer PODno: ");

PODno= sc.nextLong();

if (PODno<=0 || Pattern.matches("[0-9]{5}",String.valueOf(PODno))==false) {

System.out.println("People of determination No. field filled incorrectly!");

return;

}

//Creating object of type People of determination card

People\_of\_Determination\_Card p1=new People\_of\_Determination\_Card(tagNumber, key, balance, dailyTravelPoints, type,50,PODno);

//Creat obejct of type Customer

Customer c1=new Customer(cname,Email,EID,Adress,p1);

//Display customer

System.out.println(c1.toString());

//Adding customer to the "Database"

CustomerDB.add(c1);

System.out.println("Customer added to Database sucessfully !");

}

else if(type.equalsIgnoreCase("Elderly"))

{

//Take the customers name and Validate it to check if empty or " " and by REGEX pattern

System.out.print("Customer Name: ");

cname= sc.next();

if (cname.isEmpty() || cname.equals(" ") || cname.length() <= 1 || Pattern.matches("[a-zA-Z]\*",cname)==false)

{

System.out.println("Customer Name field filled incorrectly!");

return;

}

//Take the Customers EID and Validate it to check if empty, " " and by REGEX pattern

System.out.print("Customer EID: ");

EID= sc.next();

if (EID.isEmpty() || EID.equals(" ") ||Pattern.matches("^784-[0-9]{4}-[0-9]{7}-[0-9]{1}$",EID)==false) {

System.out.println("Emirates ID field filled incorrectly!");

return;

}

//Take the Customers Email and Validate it to check if empty, " " and by REGEX pattern

System.out.print("Customer Email: ");

Email= sc.next();

if (Email.isEmpty() || Email.equals(" ") ||Pattern.matches("[a-z0-9]+@[a-z]+\\.[a-z]{2,3}",Email)==false) {

System.out.println("Email field filled incorrectly!");

return;

}

//Take the Customers Adress and Validate it to check if empty, " " and by REGEX pattern

System.out.print("Customer Address: ");

Adress= sc.next();

if (Adress.isEmpty() || Adress.equals(" ") ||Pattern.matches("[a-zA-Z]\*",Adress)==false) {

System.out.println("Address field filled incorrectly!");

return;

}

//Creating object of type ElderlyCard

ElderlyCard e1=new ElderlyCard(tagNumber, key, balance, dailyTravelPoints, type,50,EID);

//Creat obejct of type Customer

Customer c1=new Customer(cname,Email,EID,Adress,e1);

//Display customer

System.out.println(c1.toString());

//Adding customer to the "Database"

CustomerDB.add(c1);

System.out.println("Customer added to Database sucessfully !");

}

else if(type.equalsIgnoreCase("Normal"))

{//Take the customers name and Validate it to check if empty or " " and by REGEX pattern

System.out.print("Customer Name: ");

cname= sc.next();

if (cname.isEmpty() || cname.equals(" ") || cname.length() <= 1 || Pattern.matches("[a-zA-Z]\*",cname)==false)

{

System.out.println("Customer Name field filled incorrectly!");

return;

}

//Take the Customers EID and Validate it to check if empty, " " and by REGEX pattern

System.out.print("Customer EID: ");

EID= sc.next();

if (EID.isEmpty() || EID.equals(" ") ||Pattern.matches("^784-[0-9]{4}-[0-9]{7}-[0-9]{1}$",EID)==false) {

System.out.println("Emirates ID field filled incorrectly!");

return;

}

//Take the Customers Email and Validate it to check if empty, " " and by REGEX pattern

System.out.print("Customer Email: ");

Email= sc.next();

if (Email.isEmpty() || Email.equals(" ") ||Pattern.matches("[a-z0-9]+@[a-z]+\\.[a-z]{2,3}",Email)==false) {

System.out.println("Email field filled incorrectly!");

return;

}

//Take the Customers Adress and Validate it to check if empty, " " and by REGEX pattern

System.out.print("Customer Address: ");

Adress= sc.next();

if (Adress.isEmpty() || Adress.equals(" ") ||Pattern.matches("[a-zA-Z]\*",Adress)==false) {

System.out.println("Address field filled incorrectly!");

return;

}

//Creating object regular card

Card ca1=new Card(tagNumber, key, balance, dailyTravelPoints, type);

//Create object of type Customer

Customer c1=new Customer(cname,Email,EID,Adress,ca1);

//Display customer

System.out.println(c1.toString());

//Adding customer to the "Database"

CustomerDB.add(c1);

System.out.println("Customer added to Database sucessfully !");

}

else{

{

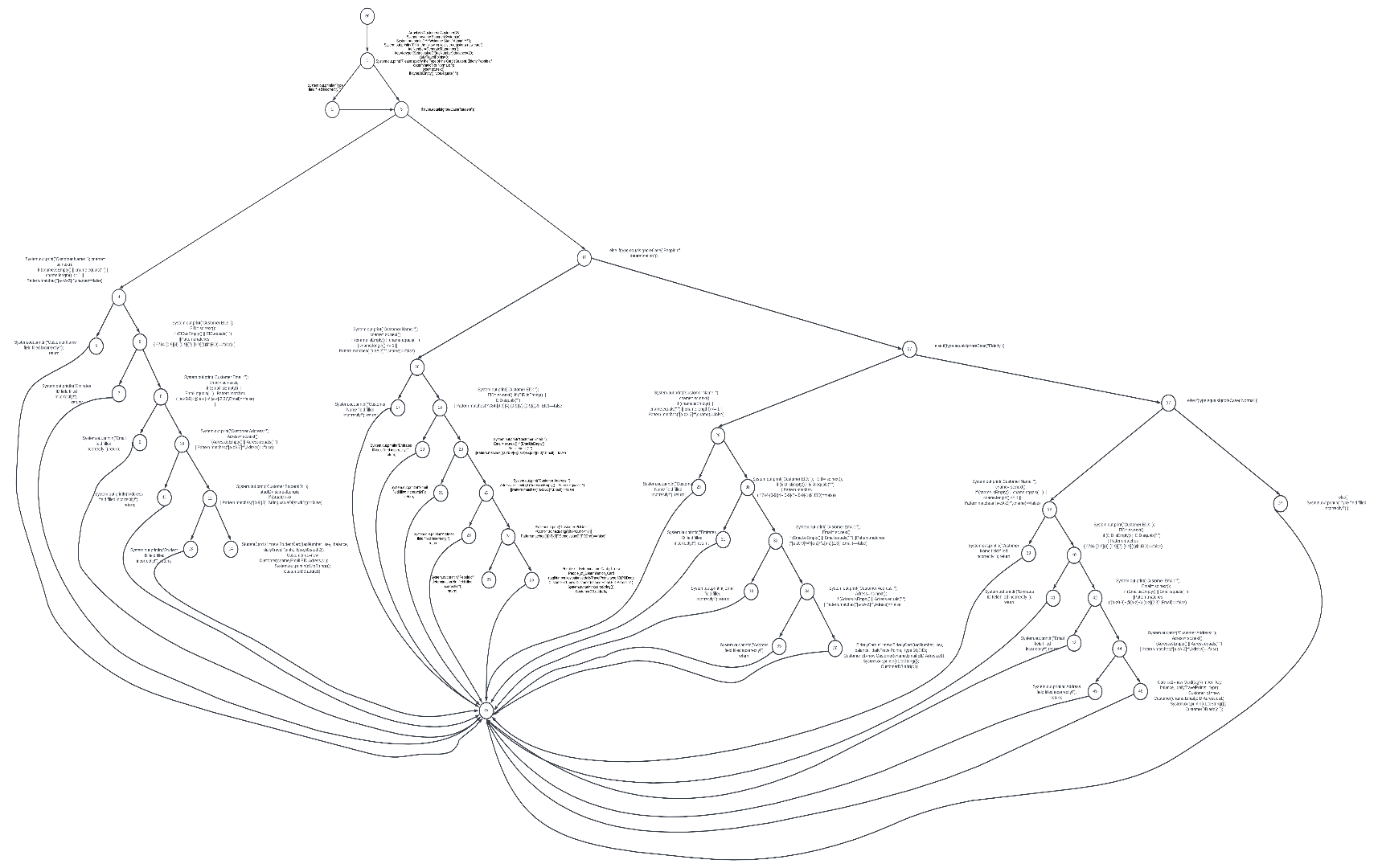
System.out.println("Type field filled incorrectly!");

}

}

}

## Control Flow Graph (CFG)



## Def and Use Analysis

|  |  |
| --- | --- |
| **Definition** | **Usage** |
| tagNumber | du| du| du| du |
| key | du| du| du| du |
| balance | du| du| du| du |
| dailyTravelPoints | du| du| du| du |
| studID | duuu |
| PODno | duuu |
| type | duuuu | duuuu | duuuu |duuuu |
| cname | duuuuu| duuuuu| duuuuu| duuuuu |
| EID | duuuu| duuuu| duuuuu| duuuu |
| Email | duuuu| duuuu| duuuu| duuuu |
| Adress | duuuu| duuuu| duuuu| duuuu |
| sc | duuuuuu| duuuuuu| duuuuu| duuuuu |

## Test Cases

### All nodes

#### Student

Type= “ ”

p1: en, 1, 2, ex

E.O: “Type field filled incorrectly!”

O.O: “Type field filled incorrectly!”

V: Pass

Type= “RandomText”

p1: en, 1, 2, 3, 15, 27, 37, 47 ex

E.O: “Type field filled incorrectly!”

O.O: “Type field filled incorrectly!”

V: Pass

Type= “student”

Customer Name= 12345

p1: en, 1, 2, 3, 4, 5 ex

E.O: “Customer name field filled incorrectly!”

O.O: “Customer name field filled incorrectly!”

V: Pass

Type= “student”

Customer Name= Ocean

Emirates ID= “”

p1: en, 1, 2, 3, 4, 6, 7 ex

E.O: “Emirated ID field filled incorrectly!”

O.O: “Emirated ID field filled incorrectly!”

V: Pass

Type= “student”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=1234567

p1: en, 1, 2, 3, 4, 6, 8, 9 ex

E.O: “Email field filled incorrectly!”

O.O: “Email field filled incorrectly!”

V: Pass

Type= “student”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=” ocean dsouza @yahoo.com”

Customer address=123

p1: en, 1, 2, 3, 4, 6, 8, 10, 11 ex

E.O: “Address field filled incorrectly!”

O.O: “Address field filled incorrectly!”

V: Pass

Type= “student”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=” ocean dsouza @yahoo.com”

Customer address=”Nasserya, near Viva”

Student ID=OneTwothreeThree

p1: en, 1, 2, 3, 4, 6, 8, 10, 12, 13 ex

E.O: “Student ID field filled incorrectly!”

O.O: “StudentID field filled incorrectly!”

V: Pass

#### People of Determination

Type= “people of determination”

Customer Name= 12345

p1: en, 1, 2, 3, 15, 16, 17 ex

E.O: “Customer name field filled incorrectly!”

O.O: “Customer name field filled incorrectly!”

V: Pass

Type= “people of determination”

Customer Name= Ocean

Emirates ID= “12345678”

p1: en, 1, 2, 3, 15, 16, 18, 19 ex

E.O: “Emirates ID field filled incorrectly!”

O.O: “Emirates ID field filled incorrectly!”

V: Pass

Type= “people of determination”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=1234567

p1: en, 1, 2, 3, 15, 16, 18, 20, 21 ex

E.O: “Email field filled incorrectly!”

O.O: “Email field filled incorrectly!”

V: Pass

Type= “people of determination”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=” ocean dsouza @yahoo.com”

Customer address=” ”

p1: en, 1, 2, 3, 15, 16, 18, 20, 22, 23 ex

E.O: “Address field filled incorrectly!”

O.O: “Address field filled incorrectly!”

V: Pass

Type= “people of determination”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=” ocean dsouza @yahoo.com”

Customer address=”Nasserya, near Viva”

People of determination No.=1234567890

p1: en, 1, 2, 3, 15, 16, 18, 20, 22, 24, 25 ex

E.O: “People of determination no. field filled incorrectly!”

O.O: “People of determination no. field filled incorrectly!”

V: Pass

#### Elderly

Type= “Elderly”

Customer Name= @12345

p1: en, 1, 2, 3, 15, 27, 28, 29 ex

E.O: “Type field filled incorrectly!”

O.O: “Type field filled incorrectly!”

V: Pass

Type= “Elderly”

Customer Name= Ocean

Emirates ID= “ ”

p1: en, 1, 2, 3, 15, 27, 28, 30, 31 ex

E.O: “Emirates ID field filled incorrectly!”

O.O: “Emirates ID field filled incorrectly!”

V: Pass

Type= “Elderly”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email= A@b@c@domain.com

p1: en, 1, 2, 3, 15, 27, 28, 30, 32, 33 ex

E.O: “Email field filled incorrectly!”

O.O: “Email field filled incorrectly!”

V: Pass

Type= “Elderly”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=” ocean dsouza @yahoo.com”

Customer address=” ”

p1: en, 1, 2, 3, 15, 27, 28, 30, 32, 34, 35 ex

E.O: “Address field filled incorrectly!”

O.O: “Address field filled incorrectly!”

V: Pass

#### Normal

Type= “Normal”

Customer Name= O5e4n

p1: en, 1, 2, 3, 15, 27, 37, 38, 39ex

E.O: “Type field filled incorrectly!”

O.O: “Type field filled incorrectly!”

V: Pass

Type= “Normal”

Customer Name= Ocean

Emirates ID= “784200076918356”

p1: en, 1, 2, 3, 15, 27, 37, 38, 40, 41 ex

E.O: “Emirates ID field filled incorrectly!”

O.O: “Emirates ID field filled incorrectly!”

V: Pass

Type= “Normal”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email= “a”b(c)d,e:f;gi[j\k]l@domain.com”

p1: en, 1, 2, 3, 15, 27, 37, 38, 40, 42, 43 ex

E.O: “Email field filled incorrectly!”

O.O: “Email field filled incorrectly!”

V: Pass

Type= “Normal”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=” ocean dsouza @yahoo.com”

Customer address=12345

p1: en, 1, 2, 3, 15, 27, 37, 38, 40, 42, 44, 45 ex

E.O: “Address field filled incorrectly!”

O.O: “Address field filled incorrectly!”

V: Pass

### All conditions

### All paths

#### Student

Type= “ ”

p1: en, 1, 2, ex

E.O: “Type field filled incorrectly!”

O.O: “Type field filled incorrectly!”

V: Pass

Type= “Test123”

p1: en, 1, 2, 3, 15, 27, 37, 47 ex

E.O: “Type field filled incorrectly!”

O.O: “Type field filled incorrectly!”

V: Pass

Type= “student”

Customer Name= “ ”

p1: en, 1, 2, 3, 4, 5 ex

E.O: “Customer name field filled incorrectly!”

O.O: “Customer name field filled incorrectly!”

V: Pass

Type= “student”

Customer Name= Ocean

Emirates ID= “123456”

p1: en, 1, 2, 3, 4, 6, 7 ex

E.O: “Emirates ID field filled incorrectly!”

O.O: “Emirates ID field filled incorrectly!”

V: Pass

Type= “student”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=A@b@c@gmail.com

p1: en, 1, 2, 3, 4, 6, 8, 9 ex

E.O: “Email field filled incorrectly!”

O.O: “Email field filled incorrectly!”

V: Pass

Type= “student”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=” ocean dsouza @yahoo.com”

Customer address=123

p1: en, 1, 2, 3, 4, 6, 8, 10, 11 ex

E.O: “Address field filled incorrectly!”

O.O: “Address field filled incorrectly!”

V: Pass

Type= “student”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=” ocean dsouza @yahoo.com”

Customer address=”Al Maysaloon”

Student ID=123-456-789

p1: en, 1, 2, 3, 4, 6, 8, 10, 12, 13 ex

E.O: “Student ID field filled incorrectly!”

O.O: “StudentID field filled incorrectly!”

V: Pass

ype= “student”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=” ocean dsouza @yahoo.com”

Customer address=”Al Maysaloon”

Student ID=12345

p1: en, 1, 2, 3, 4, 6, 8, 10, 12, 14 ex

E.O: “Customer added to Database sucessfully !”

O.O: “Customer added to Database sucessfully!”

V: Pass

#### People of Determination

Type= “people of determination”

Customer Name= J@ne

p1: en, 1, 2, 3, 15, 16, 17 ex

E.O: “Customer name field filled incorrectly!”

O.O: “Customer name field filled incorrectly!”

V: Pass

Type= “people of determination”

Customer Name= Saim

Emirates ID= “A1b2c3”

p1: en, 1, 2, 3, 15, 16, 18, 19 ex

E.O: “Emirates ID field filled incorrectly!”

O.O: “Emirates ID field filled incorrectly!”

V: Pass

Type= “people of determination”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=” ”

p1: en, 1, 2, 3, 15, 16, 18, 20, 21 ex

E.O: “Email field filled incorrectly!”

O.O: “Email field filled incorrectly!”

V: Pass

Type= “people of determination”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=” ocean dsouza @yahoo.com”

Customer address=”12345 ”

p1: en, 1, 2, 3, 15, 16, 18, 20, 22, 23 ex

E.O: “Address field filled incorrectly!”

O.O: “Address field filled incorrectly!”

V: Pass

Type= “people of determination”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=” ocean dsouza @yahoo.com”

Customer address=”Nasserya, near Viva”

People of determination No.=(empty)

p1: en, 1, 2, 3, 15, 16, 18, 20, 22, 24, 25 ex

E.O: “People of determination no. field filled incorrectly!”

O.O: “People of determination no. field filled incorrectly!”

V: Pass

Type= “people of determination”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=” ocean dsouza @yahoo.com”

Customer address=”Nasserya, near Viva”

People of determination No.=12345

p1: en, 1, 2, 3, 15, 16, 18, 19, 20, 22, 24, 26 ex

E.O: “Customer added to Database sucessfully !”

O.O: “Customer added to Database sucessfully!”

V: Pass

#### Elderly

Type= “Elderly”

Customer Name= @12345

p1: en, 1, 2, 3, 15, 27, 28, 29 ex

E.O: “Type field filled incorrectly!”

O.O: “Type field filled incorrectly!”

V: Pass

Type= “Elderly”

Customer Name= Ocean

Emirates ID= “ ”

p1: en, 1, 2, 3, 15, 27, 28, 30, 31 ex

E.O: “Emirates ID field filled incorrectly!”

O.O: “Emirates ID field filled incorrectly!”

V: Pass

Type= “Elderly”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email= A@b@c@domain.com

p1: en, 1, 2, 3, 15, 27, 28, 30, 32, 33 ex

E.O: “Email field filled incorrectly!”

O.O: “Email field filled incorrectly!”

V: Pass

Type= “Elderly”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=” ocean dsouza @yahoo.com”

Customer address=” ”

p1: en, 1, 2, 3, 15, 27, 28, 30, 32, 34, 35 ex

E.O: “Email field filled incorrectly!”

O.O: “Email field filled incorrectly!”

V: Pass

Type= “Elderly”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=” ocean dsouza @yahoo.com”

Customer address=” f”

p1: en, 1, 2, 3, 15, 27, 28, 30, 32, 34, 35 ex

E.O: “Address field filled incorrectly!”

O.O: “Address field filled incorrectly!”

V: Pass

Type= “Elderly”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=” ocean dsouza @yahoo.com”

Customer address=” Maysaloon Diyar awkaf”

p1: en, 1, 2, 3, 15, 27, 28, 30, 32, 34, 35 ex

E.O: “Customer added to Database sucessfully !”

O.O: “Customer added to Database sucessfully!”

V: Pass

#### Normal

Type= “Normal”

Customer Name= O5e4n

p1: en, 1, 2, 3, 15, 27, 37, 38, 39 ex

E.O: “Type field filled incorrectly!”

O.O: “Type field filled incorrectly!”

V: Pass

Type= “Normal”

Customer Name= Ocean

Emirates ID= “784200076918356”

p1: en, 1, 2, 3, 15, 27, 37, 38, 40, 41 ex

E.O: “Emirates ID field filled incorrectly!”

O.O: “Emirates ID field filled incorrectly!”

V: Pass

Type= “Normal”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email= “a”b(c)d,e:f;gi[j\k]l@domain.com”

p1: en, 1, 2, 3, 15, 27, 37, 38, 40, 42, 43 ex

E.O: “Email field filled incorrectly!”

O.O: “Email field filled incorrectly!”

V: Pass

Type= “Normal”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=” ocean dsouza @yahoo.com”

Customer address=12345

p1: en, 1, 2, 3, 15, 27, 37, 38, 40, 42, 44, 45 ex

E.O: “Address field filled incorrectly!”

O.O: “Address field filled incorrectly!”

V: Pass

Type= “Normal”

Customer Name= Ocean

Emirates ID= “784-2000-7691835-6”

Customer Email=” ocean dsouza @yahoo.com”

Customer address=”Maysaloon diyar awkaf building”

p1: en, 1, 2, 3, 15, 27, 37, 38, 40, 42, 44, 45 ex

E.O: “Customer added to Database sucessfully !”

O.O: “Customer added to Database sucessfully!”

V: Pass

### Conditional coverage

# Delete User from Database

public static void DeleteUser (ArrayList<Staff> StaffDB,ArrayList<Customer>CustomerDB)

{

//Customer tagnumber

long Tagnumber;

//Emirates ID

String EID;

//Employee ID variable

String ID;

//Scanner to take user input

Scanner sc=new Scanner(System.in);

String choice= sc.next();

System.out.println("\*\*\* Welcome Admin! \*\*\*");

System.out.println("Please select which user you would like to remove");

System.out.println("1. Staff 2. Customer");

// If the choice of the user is Staff

if (choice.equalsIgnoreCase("Staff"))

{

//We store his Employee ID

System.out.println("EmployeeID: ");

ID=sc.next();

//We store his EID

System.out.println("Emirated ID: ");

EID= sc.next();

//For Each staff in staffDB we loop and check

for (int i= 0;i<StaffDB.size();i++)

{

//We check if the EmpID and EID is matching by retreiving details using getters.

if(StaffDB.get(i).getID()==ID && StaffDB.get(i).getEID()==EID)

{

//When found remove from Arraylist and return

StaffDB.remove(i);

System.out.println("Employee Deleted sucessfully !");

return;

}

}

//If not found Show error message

System.out.println("Employee Not found !");

}

// If the choice of the user is Customer

else if (choice.equalsIgnoreCase("Customer"))

{

//We store his Tag number

System.out.println("Tag Number: ");

Tagnumber=sc.nextLong();

//We store his EID

System.out.println("Emirated ID: ");

EID= sc.next();

//For each staff in CustomerDB we loop and check

for (int i= 0;i<CustomerDB.size();i++)

{

//If the input Tag number is equal tro customers card and if his EID matches

if(CustomerDB.get(i).getCards().getTagNumber()==Tagnumber && CustomerDB.get(i).getEID()==EID)

{

//We then delete Him from the Arraylist

CustomerDB.remove(i);

System.out.println("Customer Deleted sucessfully !");

return;

}

}

//If not Found Show error message

System.out.println("Customer Not found !");

}//I input of choice is not Customer or staff it is invalid option

else

{

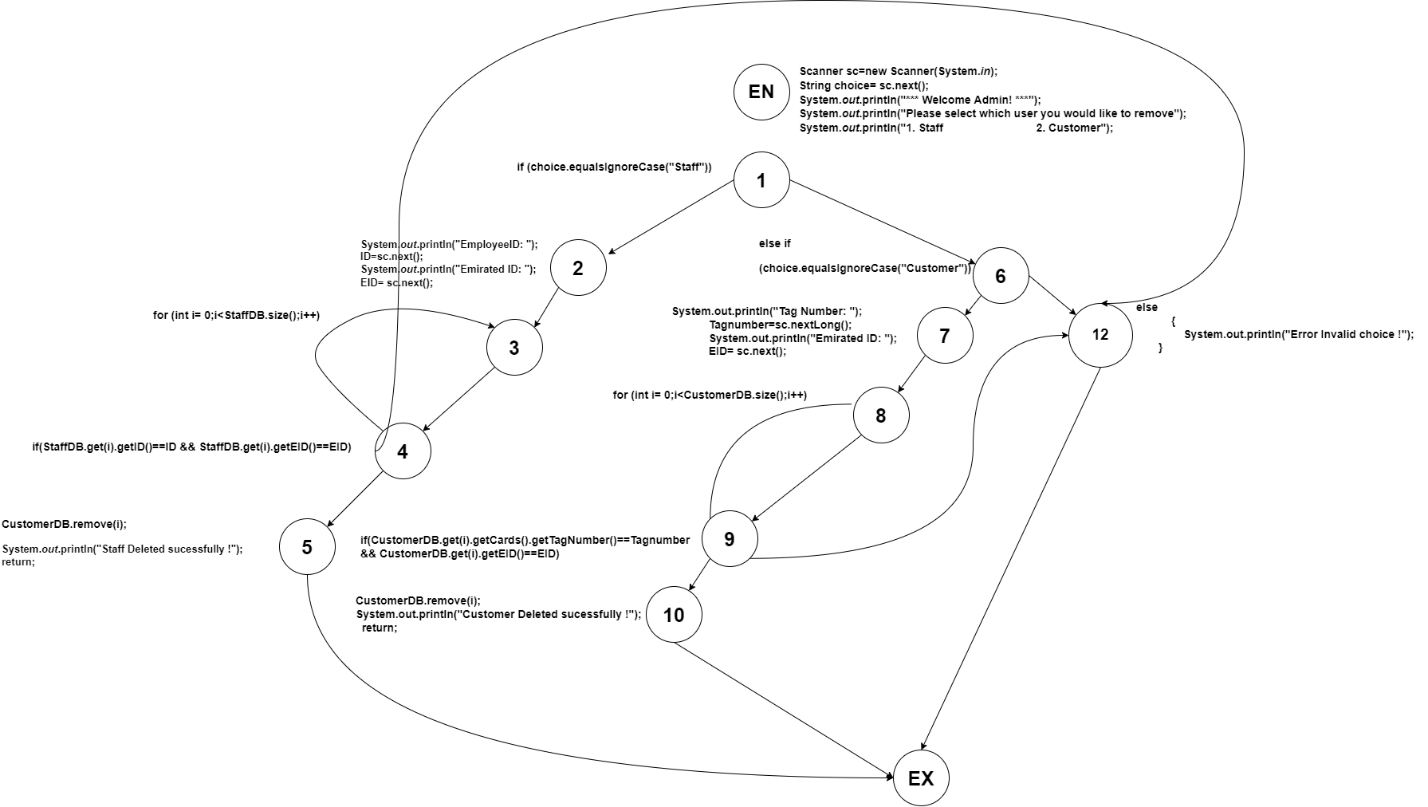
//Show error message

System.out.println("Error Invalid choice !");

}

}

## Control Flow Graph (CFG)



## Def and Use Analysis

|  |  |
| --- | --- |
| **Definition** | **Usage** |
| tagNumber | d(u)\* |
| EID | d(u)\*| d(u)\* |
| ID | d(u)\* |
| sc | duu| duu |
| choice | du| du |

## Cases

### All nodes

t1: 4

p1: en, 1,6,12, ex

E.O: “Error Invalid choice !”

O.O: “Error Invalid choice !””

V: Pass

T2: staff, Onetwothree, 784-2001-7654245

P2: en, 1, 2, (3, 4,), 12 ex

E.O: “Error Invalid choice !”

O.O: “Error Invalid choice !””

V: Pass

T3: staff, 1002, 78420017654245

P3: en, 1, 2, (3, 4,), 12 ex

E.O: “Error Invalid choice !”

O.O: “Error Invalid choice !””

V: Pass

T4: customer, 1234, 784-2001-7654245

p4: en, 1, 6, 7, (8, 9), 12 ex

E.O: “Error Invalid choice !”

O.O: “Error Invalid choice !””

V: Pass

T5: customer, 1234567890, “ ”

p5: en, 1, 6, 7, (8, 9), 12 ex

E.O: “Error Invalid choice !”

O.O: “Error Invalid choice !””

V: Pass

### All conditions

### All paths

t1: Null

p1: en, 1,6,12, ex

E.O: “Error Invalid choice !”

O.O: “Error Invalid choice !””

V: Pass

T2: staff, 123, 784-2001-7654245

P2: en, 1, 2, (3, 4,), 12 ex

E.O: “Error Invalid choice !”

O.O: “Error Invalid choice !””

V: Pass

T3: staff, 1002, “ ”

P3: en, 1, 2, (3, 4,), 12 ex

E.O: “Error Invalid choice !”

O.O: “Error Invalid choice !””

V: Pass

T4: staff, 1002, 784-2001-7654245

P3: en, 1, 2, (3, 4,), 5 ex

E.O: “Staff Deleted successfully”

O.O: “Staff Deleted successfully”

V: Pass

T5: customer, 1004, “784-2001-7654245 ”

p5: en, 1, 6, 7, (8, 9), 12 ex

E.O: “Error Invalid choice !”

O.O: “Error Invalid choice !””

V: Pass

T5: customer, 1234567890, null

p5: en, 1, 6, 7, (8, 9), 12 ex

E.O: “Error Invalid choice !”

O.O: “Error Invalid choice !””

V: Pass

T5: customer, 1234567890, “784-2001-7654245 ”

p5: en, 1, 6, 7, (8, 9), 12 ex

E.O: “Customer Deleted successfully”

O.O: “Customer Deleted successfully”

V: Pass

### All Independent Paths

*Step 1: McCabe Value*

V(G) = number of “if” + 1: 4 + 1 = 5

*Step 2: Longest path*

T1: customer, 633643219 , 784-2001-7654386

P1: en, 1, 6, 7, (8, 9), 10 ex

E.O: “Customer Deleted successfully”

O.O: “Customer Deleted successfully”

V: Pass

#### Step 3: Remaining paths

t1: staff,0024, 784-2001-7654245

p1: en, 1, 2,(3,4),5 ex

E.O: “Staff Deleted successfully”

O.O: “Staff Deleted successfully”

V: Pass

T2: staff, we24, 784-2001-765tsfst

P2: en, 1, 2, (3, 4,), 12 ex

E.O: “Employee Not found”

O.O: “Employee Not found”

V: Pass

T3: customer, 633fdfdfye , 784-2001-765453gs

P3: en, 1, 6, 7, (8, 9), 12 ex

E.O: “Error Invalid choice !”

O.O: “Error Invalid choice !””

V: Pass

T4: 3

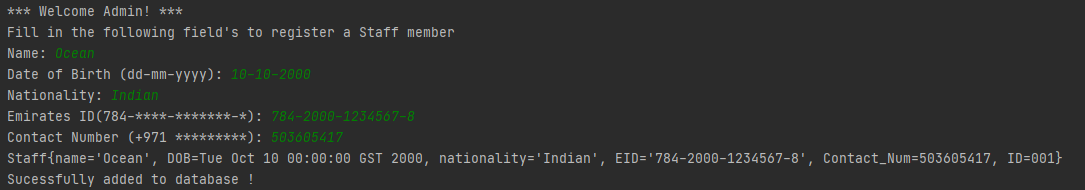
p4: en, 1,6,12, ex

E.O: “Error Invalid choice !”

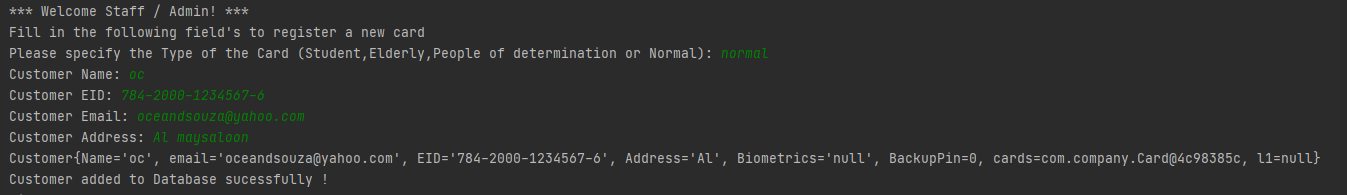
O.O: “Error Invalid choice !””

V: Pass

# Implementation Screenshots



Successfully Creating a new Staff member.



Successfully Registered a new card.

# Team Distribution Sheet – Deliverable 03

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Student name** | **Student ID** | **Items covered** | **Contribution on each item** | **Signature** |
| Saim Ali | 6482491 | * **CFG –** Register Staff, Read Card, Generate Tag Number, | 100% |  |
| Uzair Naeem | 6538654 | * **CFG –** Register Staff, Read Card, Generate Tag Number | 100% |  |
| Ocean Dsouza | 6628485 | * **Java Codes for functionalities** * **CFG** – Delete User from DB | 100% |  |
| Simran Patt | 7009690 | * **Java Codes for functionalities** * **CFG –** Register Staff, Read Card, Generate Tag Number | 100% |  |