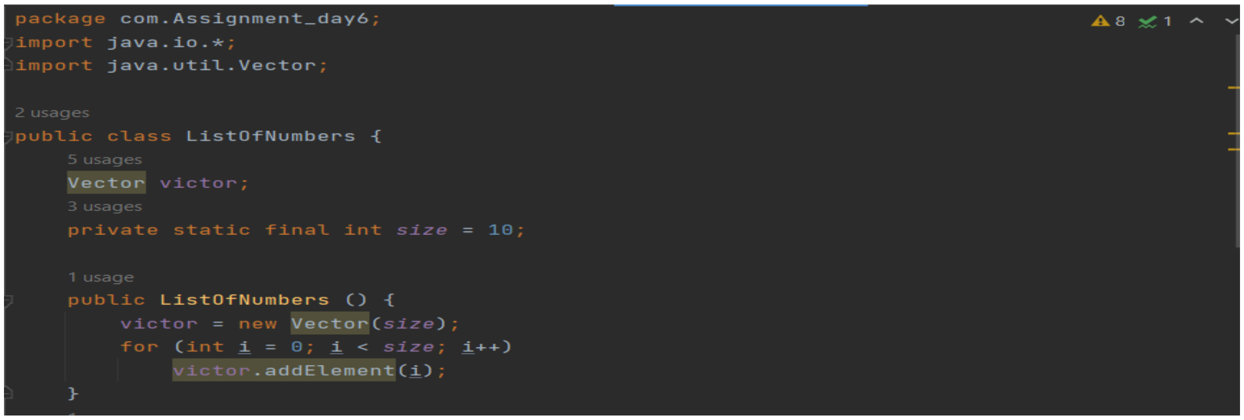
Day 6 – Assignment

Pratik K Kamble
Employee ID : 46263548
16 – 09 – 2022

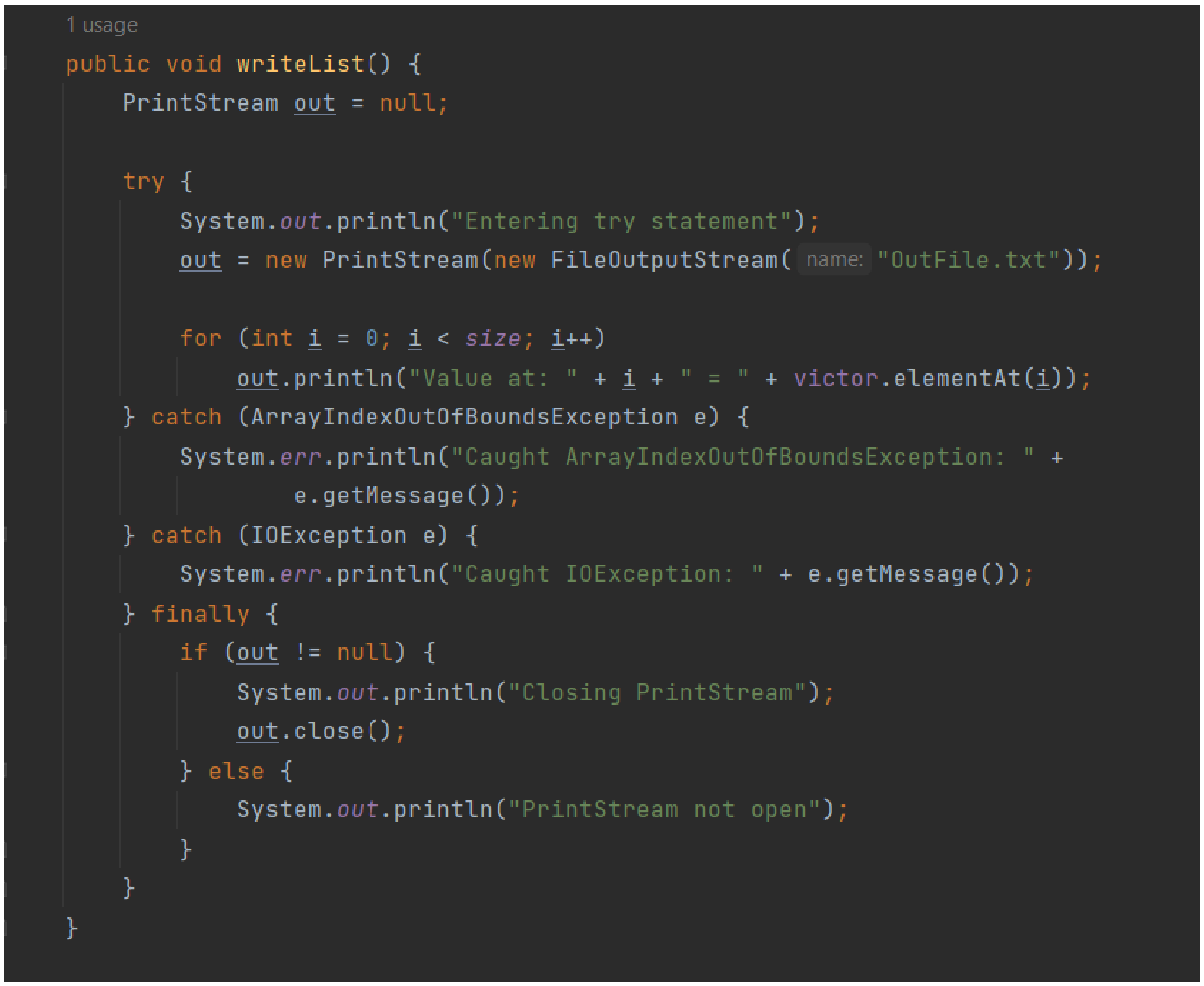
Q1: I am providing you the skeleton code. Write it as it is in your code editor. And follow the below instruction.
Add a **readList** method to **ListOfNumbers.java** (in a .java source file). This method should read in int values from a file, print each value, and append them to the end of the vector. You should catch all appropriate errors. You will also need a text file containing numbers to read in.

```
import java.io.*; import
java.util.Vector;

public class ListOfNumbers
{     private Vector victor;
    private static final int size = 10;

    public ListOfNumbers ()
{         victor = new
Vector(size);          for (int i = 0; i <
size; i++)
            victor.addElement(new Integer(i));
    }
    public void writeList()
{         PrintStream out = null;

        try {
            System.out.println("Entering try statement");
            out = new PrintStream(new FileOutputStream("OutFile.txt"));

            for (int i = 0; i < size; i++)
                out.println("Value at: " + i + " = " + victor.elementAt(i));
        } catch (ArrayIndexOutOfBoundsException e) {
            System.err.println("Caught ArrayIndexOutOfBoundsException: " +
 e.getMessage());
        } catch (IOException e) {
            System.err.println("Caught IOException: " + e.getMessage());
        } finally {
            if (out != null) {
                System.out.println("Closing
PrintStream");               out.close();
            } else {
                System.out.println("PrintStream not open");
            }
        }
    }
}
```

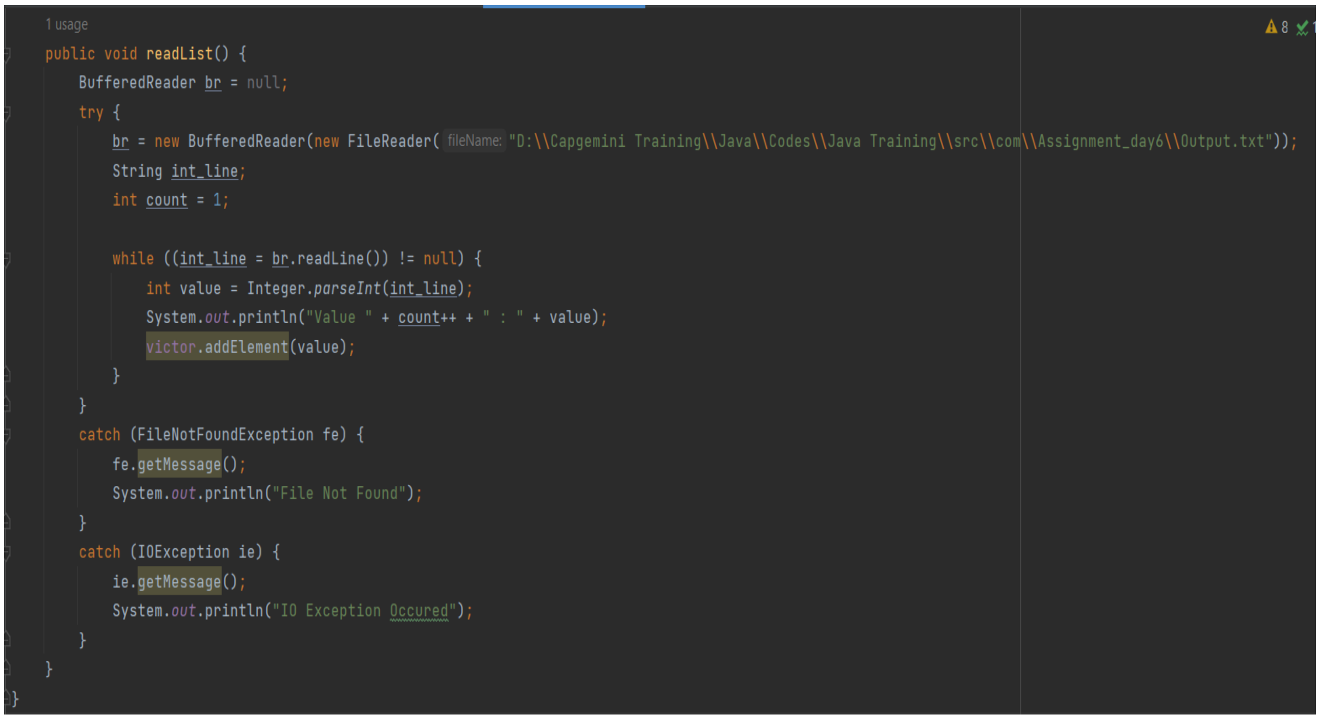
Solution –



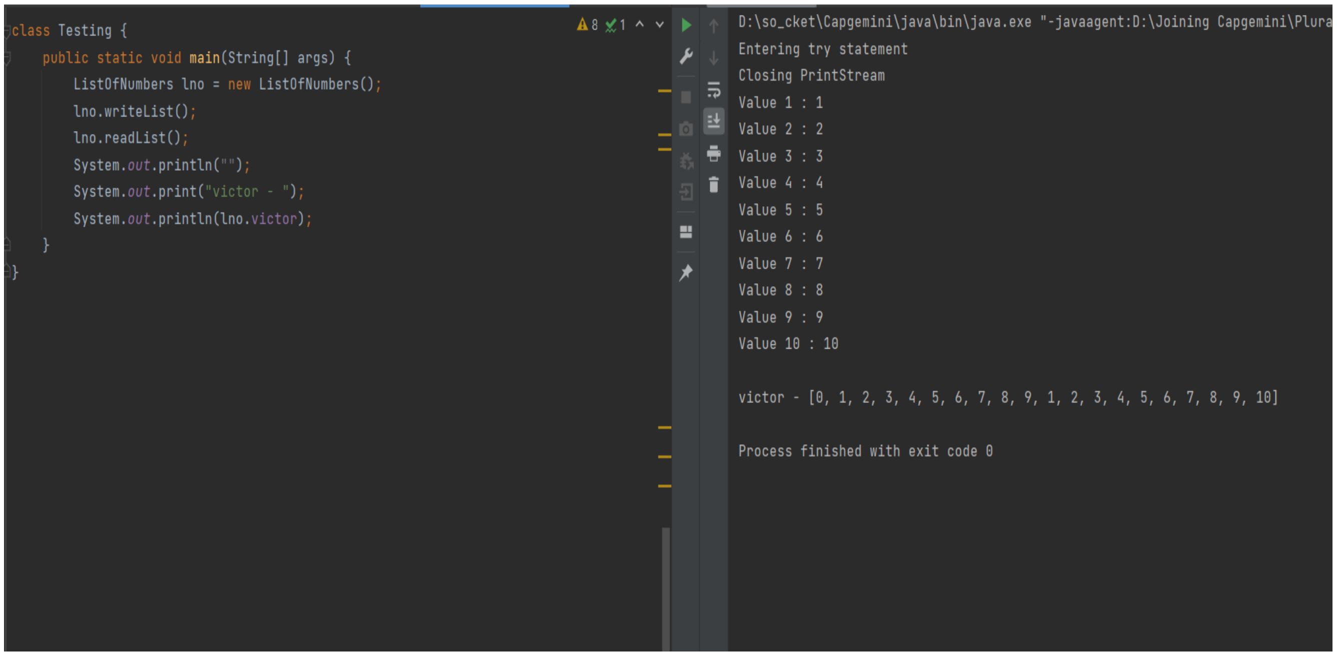
```
package com.Assignment_day6;
import java.io.*;
import java.util.Vector;

2 usages
public class ListOfNumbers {
    5 usages
    Vector victor;
    3 usages
    private static final int size = 10;

    1 usage
    public ListOfNumbers () {
        victor = new Vector(size);
        for (int i = 0; i < size; i++)
            victor.addElement(i);
    }
```



```java
1 usage
public void writeList() {
    PrintStream out = null;

    try {
        System.out.println("Entering try statement");
        out = new PrintStream(new FileOutputStream( name: "OutFile.txt"));

        for (int i = 0; i < size; i++)
            out.println("Value at: " + i + " = " + victor.elementAt(i));
    } catch (ArrayIndexOutOfBoundsException e) {
        System.err.println("Caught ArrayIndexOutOfBoundsException: " +
                e.getMessage());
    } catch (IOException e) {
        System.err.println("Caught IOException: " + e.getMessage());
    } finally {
        if (out != null) {
            System.out.println("Closing PrintStream");
            out.close();
        } else {
            System.out.println("PrintStream not open");
        }
    }
}
```



```java
1 usage
public void readList() {
    BufferedReader br = null;
    try {
        br = new BufferedReader(new FileReader( fileName: "D:\\Capgemini Training\\Java\\Codes\\Java Training\\src\\com\\Assignment_day6\\Output.txt"));
        String int_line;
        int count = 1;

        while ((int_line = br.readLine()) != null) {
            int value = Integer.parseInt(int_line);
            System.out.println("Value " + count++ + " : " + value);
            victor.addElement(value);
        }
    }
    catch (FileNotFoundException fe) {
        fe.getMessage();
        System.out.println("File Not Found");
    }
    catch (IOException ie) {
        ie.getMessage();
        System.out.println("IO Exception Occured");
    }
}
}
```



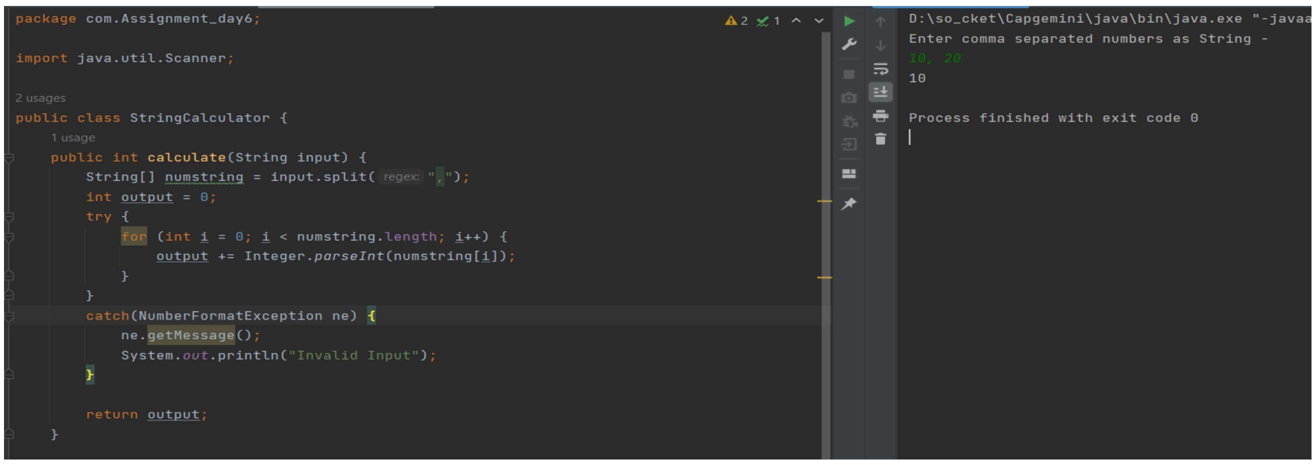
```java
class Testing {
    public static void main(String[] args) {
        ListOfNumbers lno = new ListOfNumbers();
        lno.writeList();
        lno.readList();
        System.out.println("");
        System.out.print("victor - ");
        System.out.println(lno.victor);
    }
}
```

```
D:\so_cket\Capgemini\java\bin\java.exe "-javaagent:D:\Joining Capgemini\Plura
Entering try statement
Closing PrintStream
Value 1 : 1
Value 2 : 2
Value 3 : 3
Value 4 : 4
Value 5 : 5
Value 6 : 6
Value 7 : 7
Value 8 : 8
Value 9 : 9
Value 10 : 10

victor - [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Process finished with exit code 0
```

Q2: Write a program for the StringCalculator. This program will take the comma separated numbers and print the sum of the numbers as output. You need to make sure that the numbers that are input in String form must be numeric, not the string. You need to handle the proper exception and show the proper error message to user.
Follow below table for more understanding.

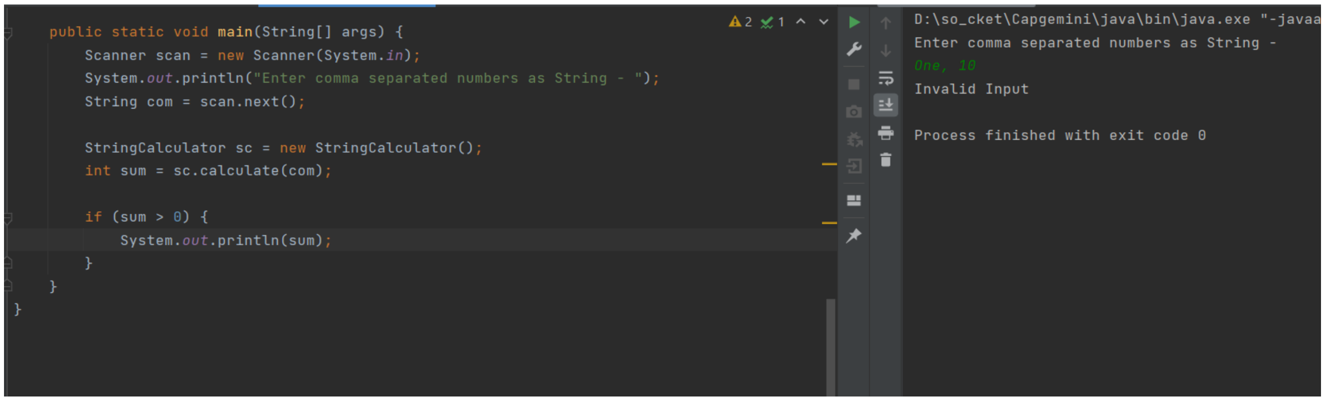
Class Name	StringCalculator
Method	public int calculate(String input){return 0;}
Description	Input will be comma separated numbers. And output will be the sum of all the numbers that are provided in input. For example Case 1: “10,20” Output : 30 Case 2: “10” Output: 10 Case 3: “” Output: 0 Case4:”One, 2” Output: Invalid Input

Solution –



```java
package com.Assignment_day6;

import java.util.Scanner;

2 usages
public class StringCalculator {
    1 usage
    public int calculate(String input) {
        String[] numstring = input.split( regex: ",");
        int output = 0;
        try {
            for (int i = 0; i < numstring.length; i++) {
                output += Integer.parseInt(numstring[i]);
            }
        }
        catch(NumberFormatException ne) {
            ne.getMessage();
            System.out.println("Invalid Input");
        }

        return output;
    }
```

```
D:\so_cket\Capgemini\java\bin\java.exe "-javaa
Enter comma separated numbers as String -
10, 20
10

Process finished with exit code 0
```

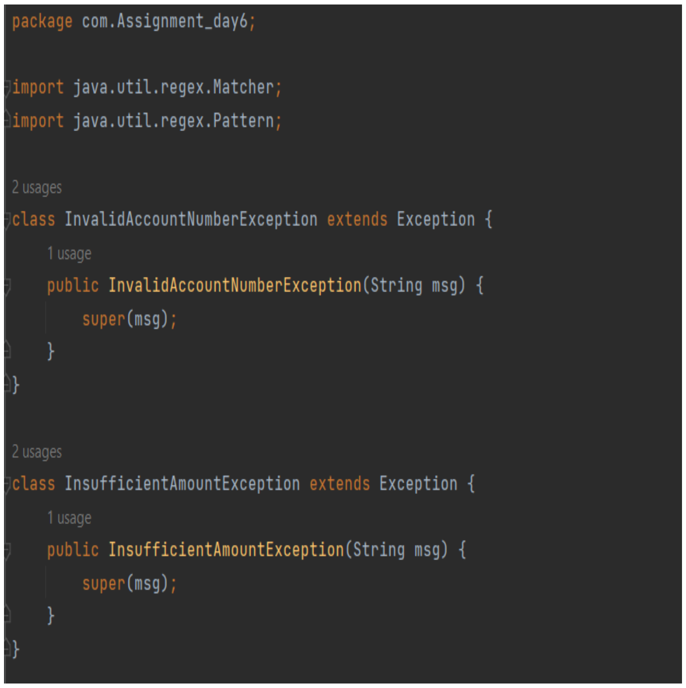


```java
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter comma separated numbers as String - ");
        String com = scan.next();

        StringCalculator sc = new StringCalculator();
        int sum = sc.calculate(com);

        if (sum > 0) {
            System.out.println(sum);
        }
    }
}
```

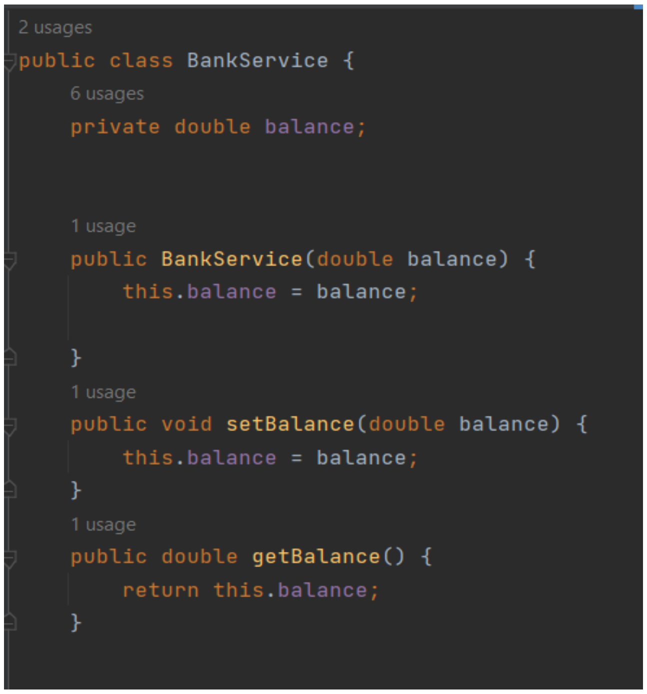
```
D:\so_cket\Capgemini\java\bin\java.exe "-javaa
Enter comma separated numbers as String -
One, 10
Invalid Input

Process finished with exit code 0
```

Q3: Create a BankService, which will have withdraw method. This method will receive two input as account number and amount to be withdrawn. When this method is called, it should prompt that it may throw InvalidAccountNumberException and InsufficientAmountException. Programer will have to handle these exception.
Assume that you will have some initial balance. So incase if amount that is withdrawn exceeds the balance then InsufficientBalanceException must be reported.
If account number is not valid then InvalidAccountNumberException must be reported.
Create a separate class to test the functionality of the BankService.

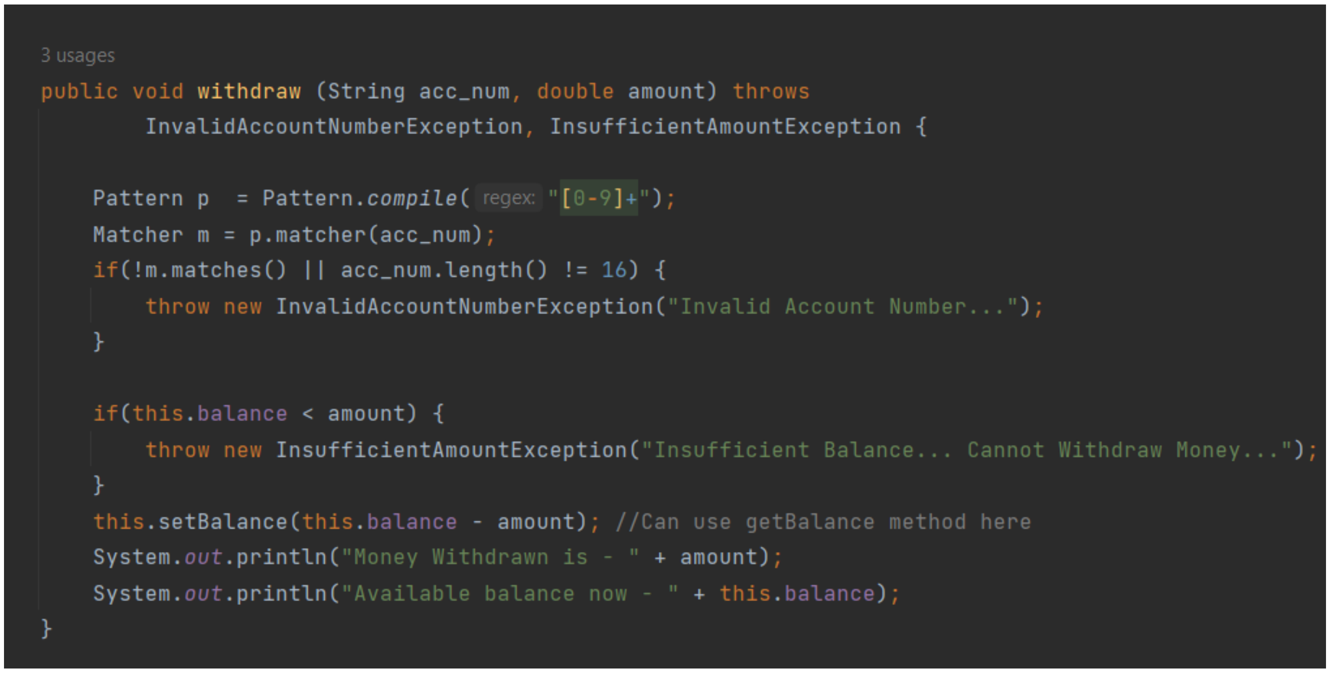
Solution –



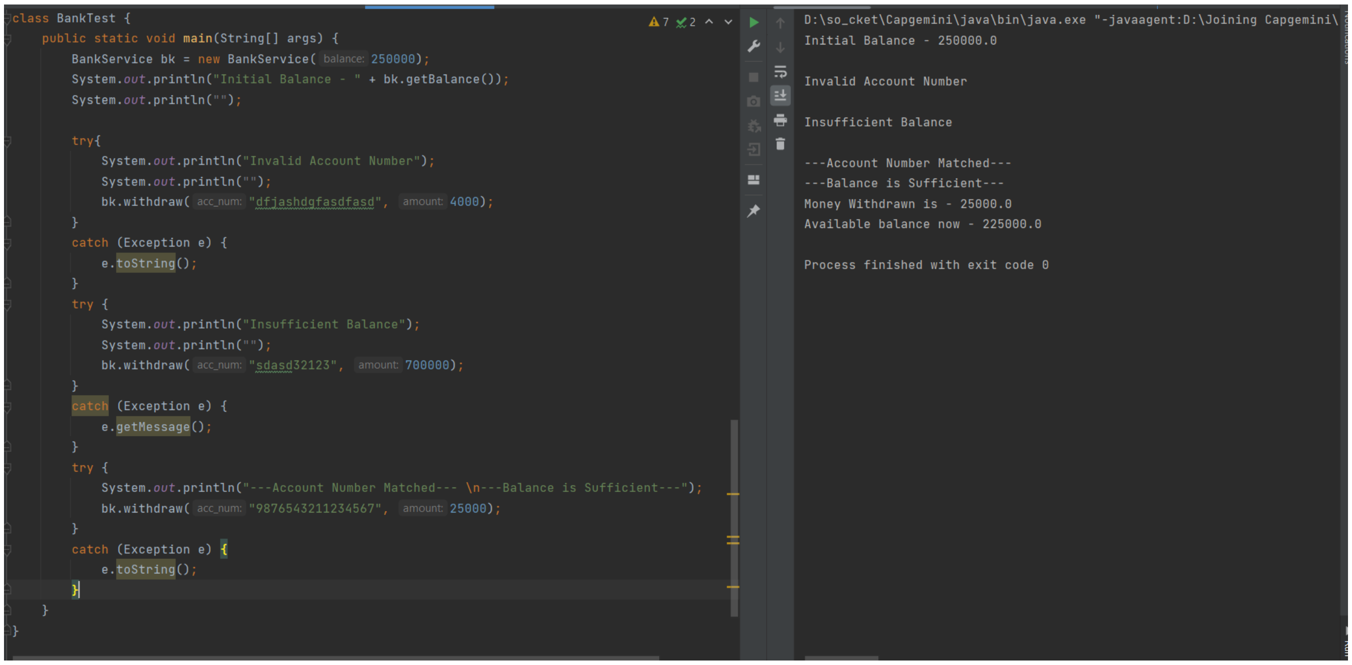
```java
package com.Assignment_day6;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

2 usages
class InvalidAccountNumberException extends Exception {
    1 usage
    public InvalidAccountNumberException(String msg) {
        super(msg);
    }
}

2 usages
class InsufficientAmountException extends Exception {
    1 usage
    public InsufficientAmountException(String msg) {
        super(msg);
    }
}
```



```java
2 usages
public class BankService {
    6 usages
    private double balance;


    1 usage
    public BankService(double balance) {
        this.balance = balance;

    }
    1 usage
    public void setBalance(double balance) {
        this.balance = balance;
    }
    1 usage
    public double getBalance() {
        return this.balance;
    }
```



```java
3 usages
public void withdraw (String acc_num, double amount) throws
        InvalidAccountNumberException, InsufficientAmountException {

    Pattern p  = Pattern.compile( regex: "[0-9]+");
    Matcher m = p.matcher(acc_num);
    if(!m.matches() || acc_num.length() != 16) {
        throw new InvalidAccountNumberException("Invalid Account Number...");
    }

    if(this.balance < amount) {
        throw new InsufficientAmountException("Insufficient Balance... Cannot Withdraw Money...");
    }
    this.setBalance(this.balance - amount); //Can use getBalance method here
    System.out.println("Money Withdrawn is - " + amount);
    System.out.println("Available balance now - " + this.balance);
}
```



```
class BankTest {
    public static void main(String[] args) {
        BankService bk = new BankService( balance: 250000);
        System.out.println("Initial Balance - " + bk.getBalance());
        System.out.println("");

        try{
            System.out.println("Invalid Account Number");
            System.out.println("");
            bk.withdraw( acc_num: "dfjashdgfasdfasd",  amount: 4000);
        }
        catch (Exception e) {
            e.toString();
        }
        try {
            System.out.println("Insufficient Balance");
            System.out.println("");
            bk.withdraw( acc_num: "sdasd32123",  amount: 700000);
        }
        catch (Exception e) {
            e.getMessage();
        }
        try {
            System.out.println("---Account Number Matched--- \n---Balance is Sufficient---");
            bk.withdraw( acc_num: "9876543211234567",  amount: 25000);
        }
        catch (Exception e) {
            e.toString();
        }
    }
}
```

```
D:\so_cket\Capgemini\java\bin\java.exe "-javaagent:D:\Joining Capgemini\
Initial Balance - 250000.0

Invalid Account Number

Insufficient Balance

---Account Number Matched---
---Balance is Sufficient---
Money Withdrawn is - 25000.0
Available balance now - 225000.0

Process finished with exit code 0
```

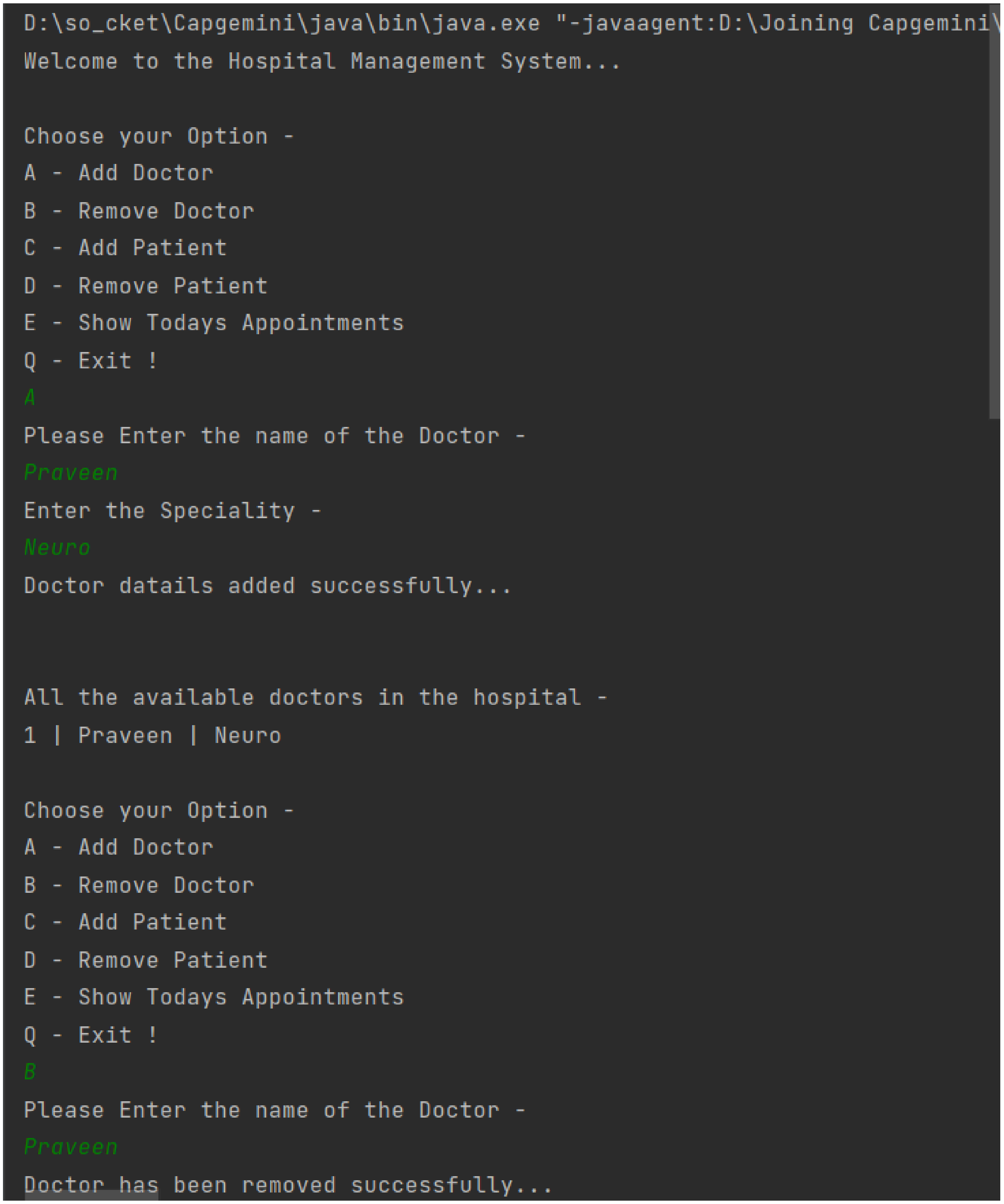
Q4: Refer the Exercise 3 from the OOPs Lab (Doctor Information System). There are multiple places where different type of issues can come. You need to Create appropriate Exceptions and handle them properly in the application. So this is the addition of your OOPs Lab, Question 3.
Follow below table for the different type of issues and the probable Exception that you need to manage.

When saving Doctor detail. Check before saving that doctor object should not be null. Otherwise appropriate Exception should be reported	Look for the NullPointerException
When accessing or trying to save the object out of array index.	Look for the ArrayIndexOutOfBoundsException
When accessing the doctor by id	DoctorNotFoundException: Need to create a custom Exception. Create it as RuntimeException. And understand the importance of RuntimeException
When accessing the patient by id	PatientNotFoundExcepiton: make it as checked exception.
Note:	You must handle the same exceptions for Patients as well. While saving patient detail.
Handle situation, when patient looks for doctor as per the speciality.	SpecialityNotFoundException: Create a custom exception, that must be handled, so that if patient look for any other specialty which is not supported.

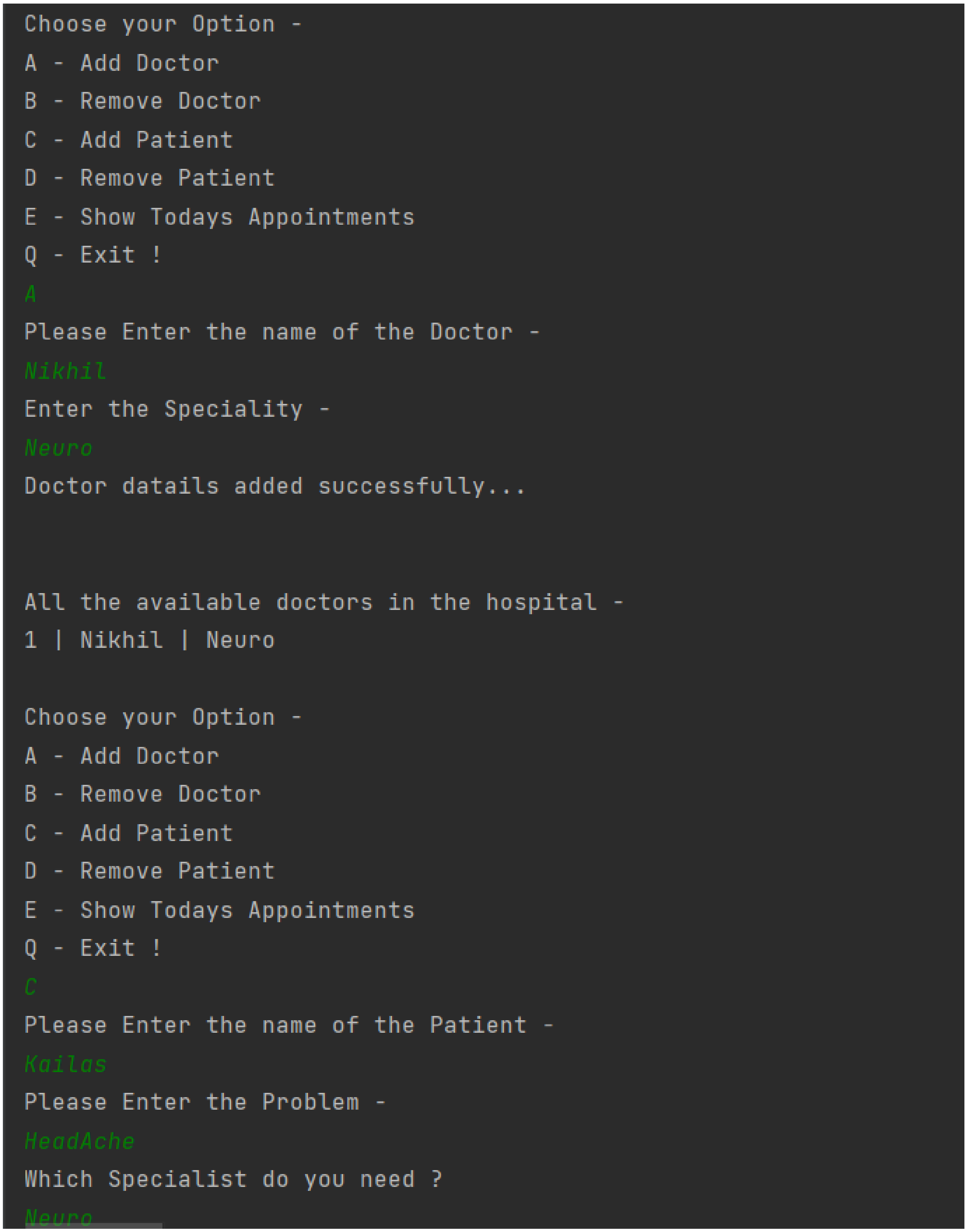
Solution –

```java
package com.Assignment_day6;

import java.util.Objects;
import java.util.Scanner;

class DoctorNotFoundException extends RuntimeException {
    public DoctorNotFoundException(String s) {
        super(s);
    }

    @Override
    public String toString() {
        return super.toString();
    }
}

class PatientNotFoundExcepiton extends Exception {
    public PatientNotFoundExcepiton(String s) {
        super(s);
    }

    @Override
    public String toString() {
        return super.toString();
    }
}
class Doctor{
    String name;
    String speciality;

    public Doctor(String name) {
        this.name = name;
    }
    public Doctor(String name, String speciality) {
        this.name = name;
        this.speciality = speciality;
    }

}

class Patient {
    String name;
    String problem;
    String reqspecial;

    public Patient(String name) {
        this.name = name;
    }
    public Patient(String name, String problem, String reqspecial) {
        this.name = name;
        this.problem = problem;
        this.reqspecial = reqspecial;
    }
}
public class HospitalManagement {
    Doctor[] doctors = new Doctor[10];
    Patient[] patients = new Patient[10];

    public void addDoctor(String name, String speciality){
        Doctor newd = new Doctor(name, speciality);
        for (int i = 0; i < doctors.length; i++) {
            if(this.doctors[i] == null) {
                this.doctors[i] = newd;
                break;
            }
        }
        System.out.println("Doctor datails added successfully...");
        System.out.println("");
    }

    public void addPatient(String name, String problem, String reqspecial) {
        Patient newp = new Patient(name, problem, reqspecial);
        for (int i = 0; i < this.patients.length; i++) {
            if(patients[i] == null) {
                this.patients[i] = newp;
                break;
            }
        }
        System.out.println("Patient datails added successfully...");
        System.out.println("");
    }

    public void removeDoctor(String name) throws DoctorNotFoundException{
        try {
            Doctor rd = new Doctor(name);
            for(int i = 0; i < this.doctors.length; i++) {
                if(Objects.equals(doctors[i].name, rd.name)) {
                    doctors[i] = null;
                    break;
                }
            }
            System.out.println("Doctor has been removed successfully...");
            System.out.println("");
        }
        catch (DoctorNotFoundException de) {
            de.toString();
        }
        catch (NullPointerException ne) {
            ne.toString();
        }
    }

    public void removePatient(String name) throws PatientNotFoundExcepiton,
NullPointerException {
        try {
            Patient rp = new Patient(name);
            for(int i = 0; i < this.patients.length; i++) {
                if(Objects.equals(patients[i].name, rp.name)) {
                    patients[i] = null;
                    break;
                }
            }
            System.out.println("Patient removed successfully...");
            System.out.println("");
        } catch (NullPointerException ne) {
            ne.toString();
        }
    }
    public void showDoctors() throws DoctorNotFoundException, NullPointerException
{
        try {
            System.out.println("All the available doctors in the hospital - ");
            int count = 1;
            for (Doctor doctor : doctors) {
                if (doctor == null) {
                } else {
                    System.out.println(count++ + " | " + doctor.name + " | " +
doctor.speciality);
                }
            }
            System.out.println("");
        }
        catch (DoctorNotFoundException de) {
            de.toString();
        }
        catch (NullPointerException ne) {
            ne.toString();
        }
    }

    public void showPatients() throws NullPointerException,
PatientNotFoundExcepiton {
        try {
            System.out.println("All the Patients having today's appointment - ");
            int count = 1;
            for (Patient patient : this.patients) {
                if (patient == null) {
                    continue;
                } else {
                    System.out.println(count++ + " | " + patient.name + " | " +
patient.problem);
                }
            }
            System.out.println("");
        } catch (NullPointerException ne) {
            ne.toString();
        }
    }

    public void viewAppointment() throws PatientNotFoundExcepiton,
NullPointerException, DoctorNotFoundException {
        int count = 1;
        try{
            for(int i = 0; i < doctors.length; i++) {
                if (this.doctors[i] != null){
                    for (int j = 0; j < patients.length; j++) {
                        if(this.patients[j] != null) {
                            if
(this.doctors[i].speciality.equals(this.patients[j].reqspecial) ) {
                                System.out.println(count++ + " | " +
this.doctors[i].name
                                        + " -|- " + this.patients[j].name + " -|- "
                                        + this.patients[j].problem);
                                throw new PatientNotFoundExcepiton("Patient Not
Found...");
                            }
                        }
                        else {
//                          System.out.println("Patient is not present in the
database...");
//                          break;
                        }
                    }
                    throw new DoctorNotFoundException("Doctor Not Found...");

                }
                else {
//                  System.out.println("Doctor is not present in the database...");
//                  break;
                }
            }
        }
        catch (PatientNotFoundExcepiton pe) {
            pe.toString();
        }
        catch (DoctorNotFoundException de) {
            de.toString();
        }
        catch (NullPointerException ne) {
            ne.toString();
        }
    }


    public static void main(String[] args) throws PatientNotFoundExcepiton {
        HospitalManagement hm = new HospitalManagement();

        System.out.println("Welcome to the Hospital Management System...");
        System.out.println("");
        Scanner scan = new Scanner(System.in);

        boolean quit = false;
        while(!quit) {
            System.out.println("Choose your Option - ");
            System.out.println("A - Add Doctor");
            System.out.println("B - Remove Doctor");
            System.out.println("C - Add Patient");
            System.out.println("D - Remove Patient");
            System.out.println("E - Show Todays Appointments");
            System.out.println("Q - Exit !");
            String choice = scan.next();
            switch (choice) {
                case "A" :
                    System.out.println("Please Enter the name of the Doctor - ");
                    String dname = scan.next();
                    System.out.println("Enter the Speciality - ");
                    String speciality = scan.next();
                    hm.addDoctor(dname, speciality);
                    System.out.println("");
```

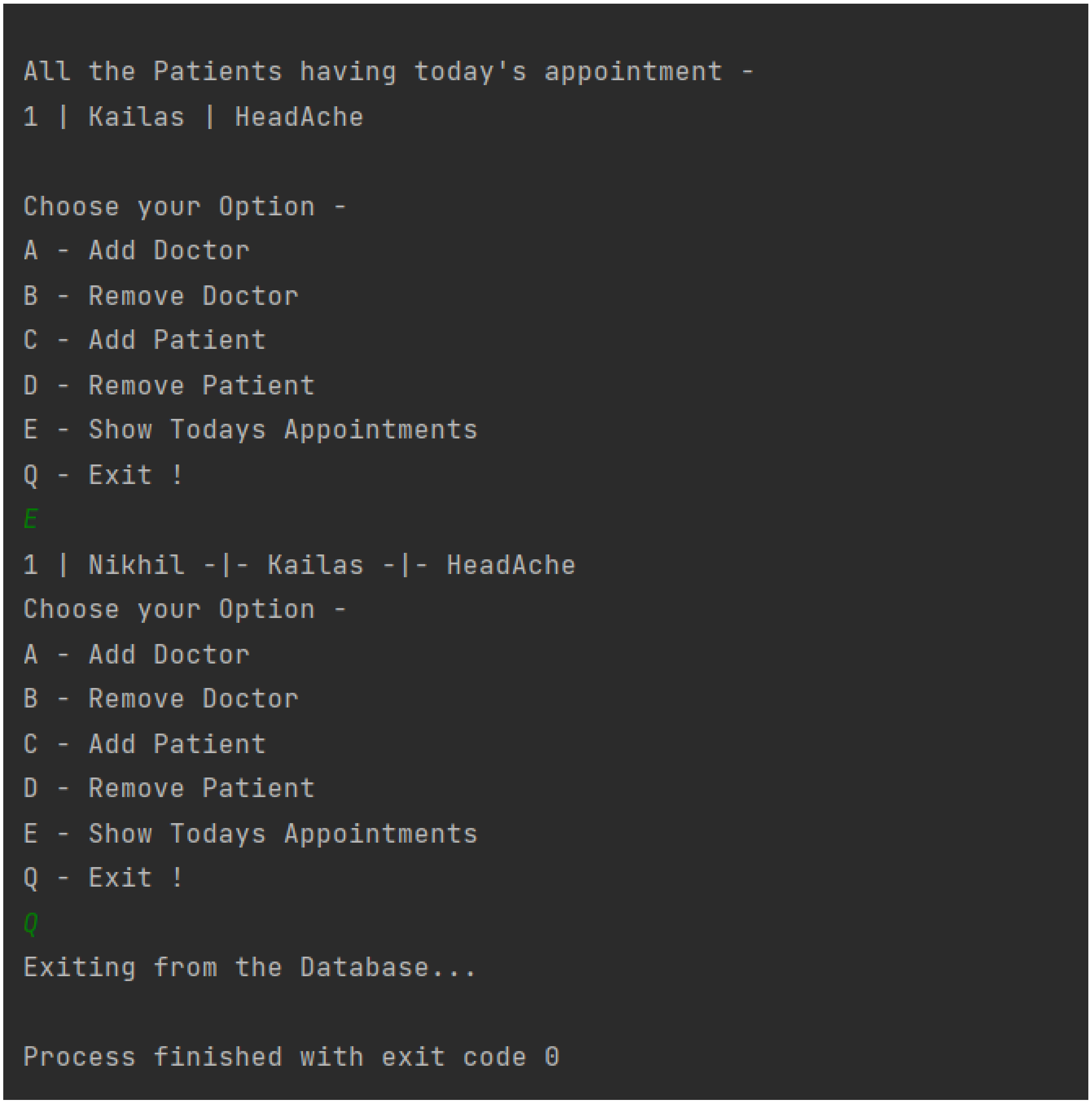
```java
                    hm.showDoctors();
                    break;
                case "B" :
                    System.out.println("Please Enter the name of the Doctor - ");
                    String drname = scan.next();
                    hm.removeDoctor(drname);
                    break;
                case "C" :
                    System.out.println("Please Enter the name of the Patient - ");
                    String pname = scan.next();
                    System.out.println("Please Enter the Problem - ");
                    String problem = scan.next();
                    System.out.println("Which Specialist do you need ?");
                    String reqspecial = scan.next();
                    hm.addPatient(pname, problem, reqspecial);
                    System.out.println("");
                    hm.showPatients();
                    break;
                case "D" :
                    System.out.println("Please Enter the name of the Patient - ");
                    String prname = scan.next();
                    hm.removePatient(prname);
                    break;
                case "E" :
                    hm.viewAppointment();
                    break;
                case "Q" :
                    System.out.println("Exiting from the Database...");
                    quit = true;
                    break;
                default :
                    System.out.println("Please Enter the correct choice - ");
                    break;
            }
        }


//          System.out.println("");
//          hm.addDoctor("Neeta", "Dentist");
//          hm.addDoctor("Kevin", "Neurologist");
//          hm.showDoctors();
//          System.out.println("");
//          hm.addPatient("Praveen", "Brain Stroke", "Neurologist");
//          hm.showPatients();
//          System.out.println("");
////          hm.removePatient("Praveen");
////          hm.showPatients();
//          System.out.println("These are today's Appointments - ");
//          hm.viewAppointment();
//          hm.removeDoctor("Neeta");
//          hm.showDoctors();
    }

}
```



```
D:\so_cket\Capgemini\java\bin\java.exe "-javaagent:D:\Joining Capgemini\
Welcome to the Hospital Management System...

Choose your Option -
A - Add Doctor
B - Remove Doctor
C - Add Patient
D - Remove Patient
E - Show Todays Appointments
Q - Exit !
A
Please Enter the name of the Doctor -
Praveen
Enter the Speciality -
Neuro
Doctor datails added successfully...



All the available doctors in the hospital -
1 | Praveen | Neuro

Choose your Option -
A - Add Doctor
B - Remove Doctor
C - Add Patient
D - Remove Patient
E - Show Todays Appointments
Q - Exit !
B
Please Enter the name of the Doctor -
Praveen
Doctor has been removed successfully...
```



```
Choose your Option -
A - Add Doctor
B - Remove Doctor
C - Add Patient
D - Remove Patient
E - Show Todays Appointments
Q - Exit !
A
Please Enter the name of the Doctor -
Nikhil
Enter the Speciality -
Neuro
Doctor datails added successfully...


All the available doctors in the hospital -
1 | Nikhil | Neuro

Choose your Option -
A - Add Doctor
B - Remove Doctor
C - Add Patient
D - Remove Patient
E - Show Todays Appointments
Q - Exit !
C
Please Enter the name of the Patient -
Kailas
Please Enter the Problem -
HeadAche
Which Specialist do you need ?
Neuro
```



```
All the Patients having today's appointment -
1 | Kailas | HeadAche

Choose your Option -
A - Add Doctor
B - Remove Doctor
C - Add Patient
D - Remove Patient
E - Show Todays Appointments
Q - Exit !
E
1 | Nikhil -|- Kailas -|- HeadAche
Choose your Option -
A - Add Doctor
B - Remove Doctor
C - Add Patient
D - Remove Patient
E - Show Todays Appointments
Q - Exit !
Q
Exiting from the Database...

Process finished with exit code 0
```