Name: **Anirban Das** Class: **BCSE-II** Sec: **A3** Roll No.: **001910501077**

**OOP PROGRAMMING LAB (JAVA)**

**ASSIGNMENT 1**

Problem 1:

CODE:

**Student.java**

import java.util.Scanner;

class Date { //class to store date (might use default class later)

private int dd;

private int mm;

private int yy;

public Date() {

dd = -1; mm = -1; yy = -1;

}

public Date(Date D) {

dd = D.dd; mm = D.mm; yy = D.yy;

}

public void getInput() {

Scanner sc = new Scanner(System.in);

System.out.println("Enter date in numbers: ");

dd = sc.nextInt();

sc.nextLine();

System.out.println("Enter month in numbers: ");

mm = sc.nextInt();

sc.nextLine();

System.out.println("Enter year in numbers: ");

yy = sc.nextInt();

sc.nextLine();

}

public String toString() {

return (dd + "/" + mm + "/" + yy);

}

public int retDate() {

return dd;

}

public int retMonth() {

return mm;

}

public int retYear() {

return yy;

}

public String retYearLast2Digs() { //return last 2 digits of the roll

return String.valueOf(yy % 100);

}

}

//stores student data

class Student {

protected String roll;

private String name;

private String course;

private Date admissionDate = new Date(); //instance of date class

protected static final int n\_subs = 5;

protected int[] marks = null;

static final int n\_stud = 1000;

static Student[] studentList = new Student[n\_stud];

static int studentsAdmitted = 0;

public Student() {

roll = null; name = null; course = null;

}

public boolean isEvaluated() { //check presence of marks

return (marks != null);

}

public Student(Student S) { //construcctor

roll = new String(S.roll);

name = new String(S.name);

course = new String(S.course);

admissionDate = new Date(S.admissionDate);

}

public static Student search(String r) { //search student by full roll

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

if (studentList[i].roll.equals(r)) {

return studentList[i];

}

}

return null;

}

public static boolean remove(String r) { //remove student

int i;

for (i = 0; i < studentsAdmitted; ++i) {

if (studentList[i].roll.equals(r)) {

break;

}

}

if (i == studentsAdmitted) {

return false;

}

for (; i < studentsAdmitted; ++i) {

studentList[i] = studentList[i + 1];

}

studentList[i] = null;

--studentsAdmitted;

return true;

}

public int getInput() { //get input

Scanner sc = new Scanner(System.in);

System.out.println("Enter Name: ");

name = sc.nextLine();

System.out.println("Enter Course: ");

course = sc.nextLine();

admissionDate.getInput();

return 1;

}

private String getRollFormat(int r, int tot) { //generate roll acc to format

String temp = String.valueOf(r);

String res = new String();

for (int i = 0; i < (tot - temp.length()); ++i) {

res += "0";

}

res += temp;

return res;

}

public void admit(int nextRoll) { //add student data in the list

roll += admissionDate.retYearLast2Digs();

roll += getRollFormat(nextRoll, 3);

studentList[studentsAdmitted] = this;

++studentsAdmitted;

}

public void getMarks() { //entry of marks

Scanner sc = new Scanner(System.in);

marks = new int[n\_subs];

System.out.println("Enter marks for 5 subjects: ");

for (int i = 0; i < n\_subs; ++i) {

marks[i] = sc.nextInt();

if(marks[i]>100){

System.out.println("Marks Out Of Bounds!");

return;

}

sc.nextLine();

}

}

public int[] returnMarks() {

return marks;

}

public String getRoll() {

return roll;

}

public String getName() {

return name;

}

int getTotalMarks() { //evaluate total marks

if (marks == null)

return -1;

int sum = 0;

for (int i = 0; i < n\_subs; ++i) {

sum += marks[i];

}

return sum;

}

public void showMarkSheet() { //marksheet of student

System.out.println("\n\*\*\*\*\*\*MARKSHEET\*\*\*\*\*\*");

System.out.println("---------------------");

System.out.println("\nRoll: " + roll);

System.out.println("Name: " + name);

System.out.println("Course: " + course);

System.out.println("Date of Admission: " + admissionDate);

if (marks == null) {

System.out.println("Marks not evaluated.");

return;

}

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

for (int i = 0; i < n\_subs; ++i) {

System.out.println("Subject" + (i + 1) + ": " + marks[i]);

}

System.out.println("Total Marks: " + getTotalMarks());

}

}

**ExtendedStudent.java**

import java.util.\*;

class ExtendedStudent extends Student {

private String dept;

public ExtendedStudent() {

super();

dept = null;

}

public ExtendedStudent(Student S, int idx) {

super(S);

dept = new String(DeptList.getDept(idx));

}

@Override

public int getInput() {

super.getInput();

DeptList.showDeptList();

Scanner sc = new Scanner(System.in);

System.out.println("Enter Your Choice:");

int choice = sc.nextInt();

sc.nextLine();

dept = DeptList.getDept(choice);

roll = new String(DeptList.getDeptCode(choice));

return choice;

}

public void setMarks(int m[]) {

marks = new int[n\_subs];

for (int i = 0; i < n\_subs; ++i) {

marks[i] = m[i];

}

}

public static final Comparator<ExtendedStudent> OnTotalMarks = new Comparator<ExtendedStudent>() {

public int compare(ExtendedStudent a, ExtendedStudent b) {

return Integer.compare(b.getTotalMarks(), a.getTotalMarks());

}

};

public boolean equals(ExtendedStudent obj) {

return roll.equals(obj.getRoll());

}

}

**DeptList.java**

public class DeptList {

static int n\_dept = 4;

static String[] depts = {"Computer Science and Engineering",

"Electronics and Telecommunication Engineering",

"Chemical Engineering",

"Mechanical Engineering"};

static String[] deptcodes = {"BCSE", "ETCE", "BCHE", "MECH"};

static int[] deptAdmitted = new int[n\_dept];

public static void showDeptList() {

System.out.println("Departments Available: ");

for (int i = 0; i < n\_dept; ++i) {

System.out.println(String.valueOf(i + 1) + ". " + depts[i]);

}

}

public static int getIndex(String deptcode) {

for (int i = 0; i < n\_dept; ++i) {

if (deptcodes[i].equals(deptcode)) {

return i + 1;

}

}

return 0;

}

public static String getDept(int choice) {

return depts[choice - 1];

}

public static String getDeptCode(int choice) {

return deptcodes[choice - 1];

}

public static int getNextRoll(int choice) {

return deptAdmitted[choice - 1] + 1;

}

public static void updateDeptRoll(int choice) {

++deptAdmitted[choice - 1];

}

}

**Activities.java**

import java.util.\*;

public class activities {

static int menu() {

System.out.println("\n\*\*MENU\*\*");

System.out.println("1. Add Student.");

System.out.println("2. Add Marks of Student.");

System.out.println("3. Display MarkSheet of a Student");

System.out.println("4. Display Number of Admitted Students.");

System.out.println("5. Department RankList based on Marks.");

System.out.println("6. Remove Student");

System.out.println("7. Exit.");

System.out.println("Enter Your choice: ");

int choice;

Scanner sc = new Scanner(System.in);

choice = sc.nextInt();

sc.nextLine();

return choice;

}

public static void main (String args[]) {

Student S;

//ararylist storing students of particular department

ArrayList<ExtendedStudent>[] deptList = new ArrayList[DeptList.n\_dept];

for (int i = 0; i < DeptList.n\_dept; ++i) {

deptList[i] = new ArrayList<ExtendedStudent>();

}

ExtendedStudent tempE;

int choice = 0;

int idx = 0;

String r;

Scanner sc = new Scanner(System.in);

while (choice != 7) {

choice = menu();

switch (choice) {

case 1:

S = new ExtendedStudent();

idx = S.getInput();

S.admit(DeptList.getNextRoll(idx));

System.out.println("Your roll number is: " + S.getRoll()); //autognereted roll

tempE = new ExtendedStudent(S, idx);

deptList[idx - 1].add(tempE);

DeptList.updateDeptRoll(idx);

break;

case 2:

System.out.println("Enter roll number: ");

r = sc.nextLine();

S = Student.search(r);

if (S == null) { //student not present

System.out.println("Invalid Roll Number.");

}

else {

S.getMarks();

idx = DeptList.getIndex(S.getRoll().substring(0, 4));

for (ExtendedStudent E : deptList[idx]) {

if (E.getRoll().equals(S.getRoll())) {

E.setMarks(S.returnMarks());

break;

}

}

System.out.println("Marks Added Successfully.");

}

break;

case 3:

System.out.println("Enter roll number: ");

r = sc.nextLine();

S = Student.search(r);

if (S == null) { //student not present

System.out.println("Invalid Roll Number.");

}

else {

S.showMarkSheet();

}

break;

case 4:

System.out.println("Number of students taken admission: " + Student.studentsAdmitted);

break;

case 5:

DeptList.showDeptList();

System.out.println("Choose the department: ");

idx = sc.nextInt();

sc.nextLine();

Collections.sort(deptList[idx-1], ExtendedStudent.OnTotalMarks);

for (ExtendedStudent E : deptList[idx - 1]) {

if (E.isEvaluated()) {

System.out.println(E.getName() + ": " + E.getTotalMarks());

}

else{

System.out.println(E.getName() + ": Marks Not Entered Yet!");

}

}

break;

case 6:

System.out.println("Enter roll number to be removed: ");

r = sc.nextLine();

S = Student.search(r);

if (S == null) {

System.out.println("Invalid Roll.");

break;

}

idx = DeptList.getIndex(S.getRoll().substring(0, 4));

tempE = new ExtendedStudent(S, idx);

int i = 0;

for (ExtendedStudent E : deptList[idx - 1]) {

if (E.equals(tempE)) {

break;

}

++i;

}

deptList[idx - 1].remove(i);

Student.remove(r);

System.gc(); //garbage collector

S = null;

tempE = null;

System.gc();

System.out.println("Student removed!");

break;

case 7:

break;

default:

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

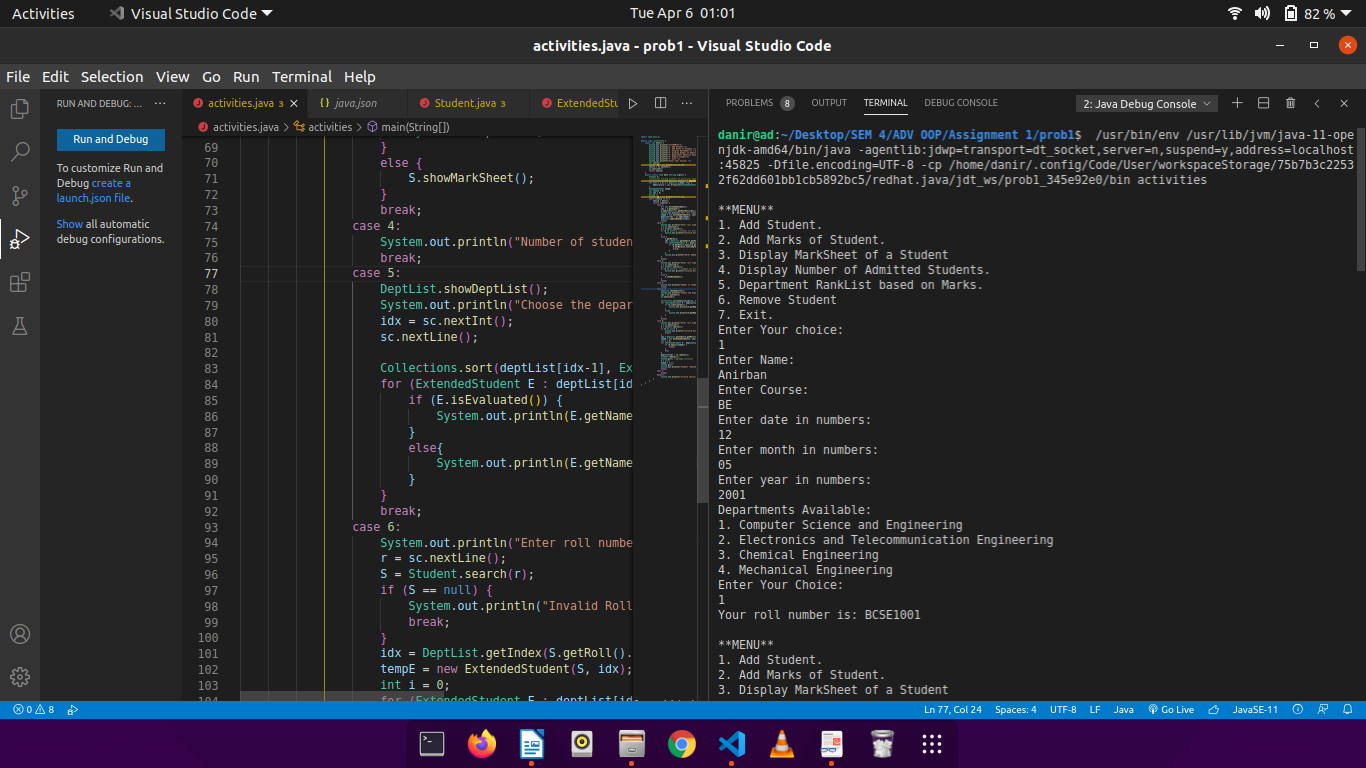
}

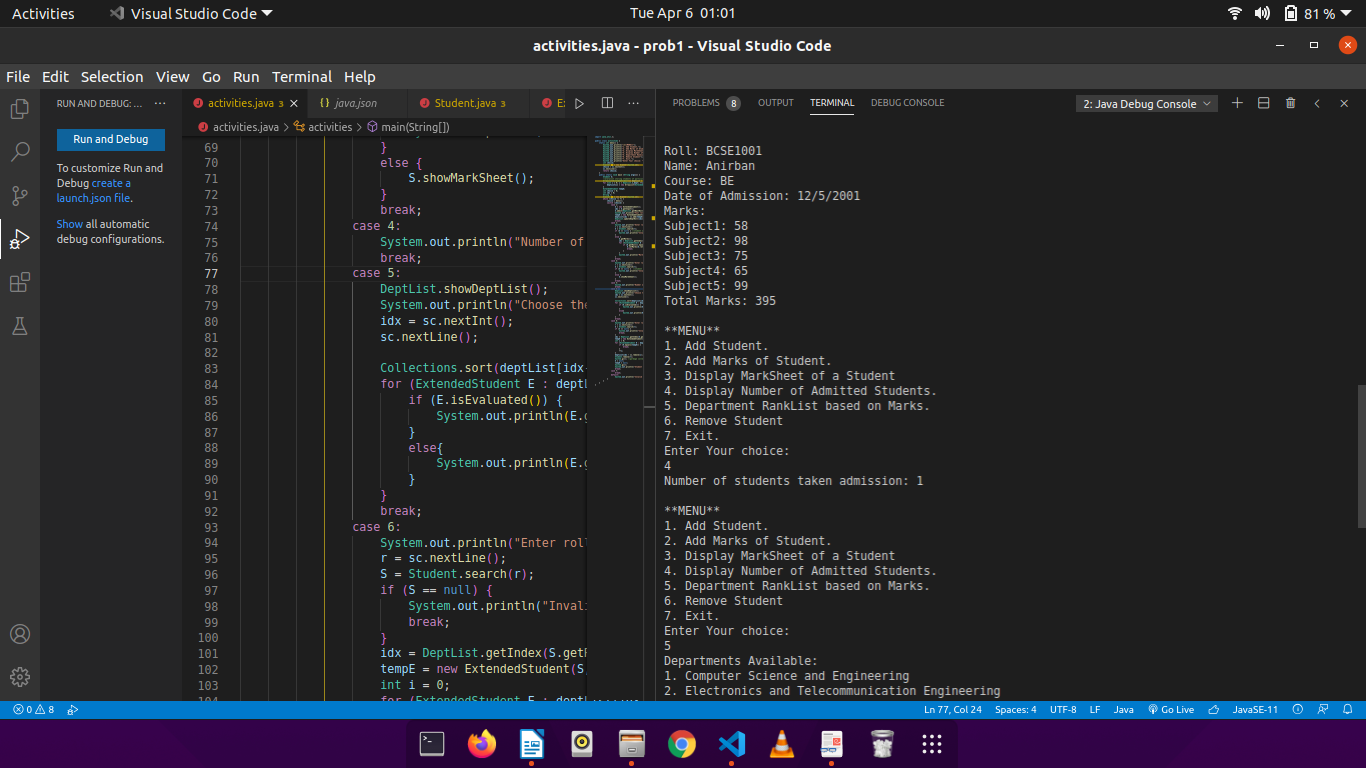
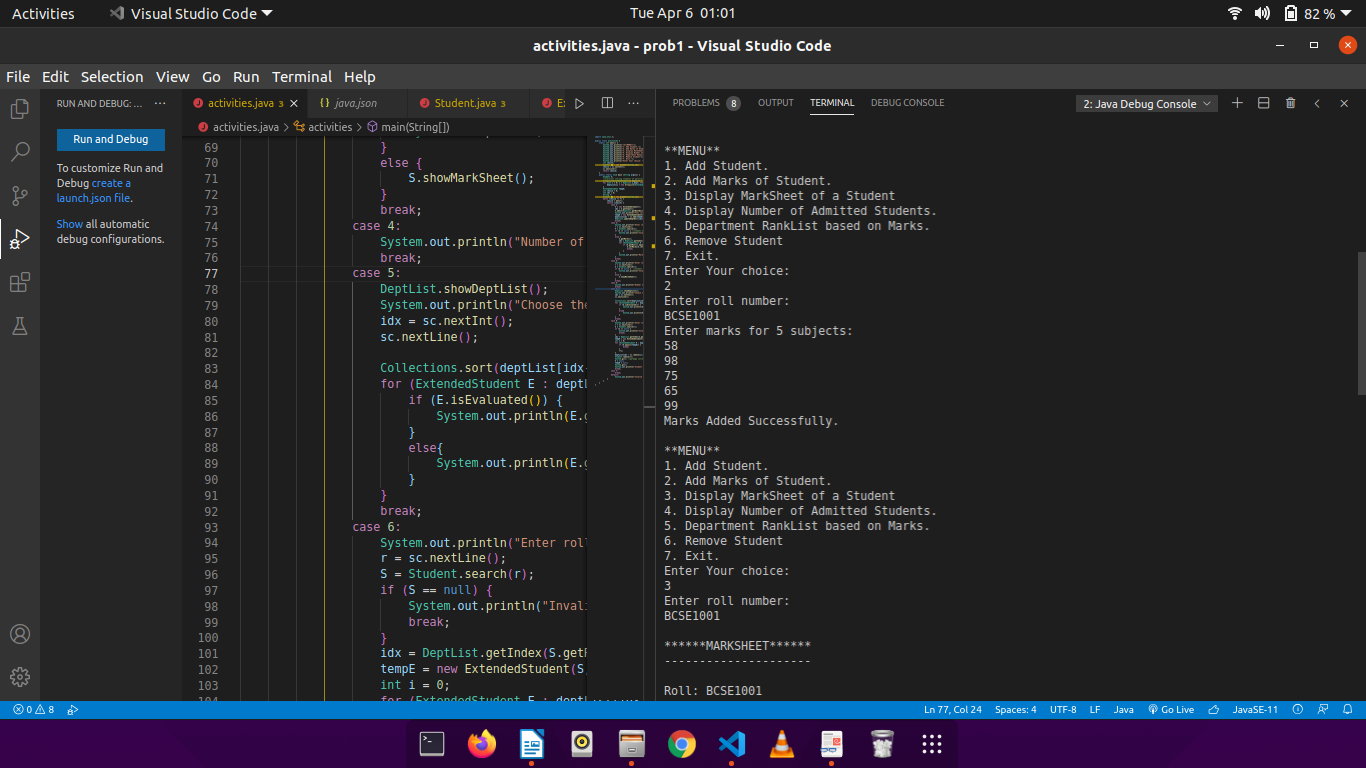
}

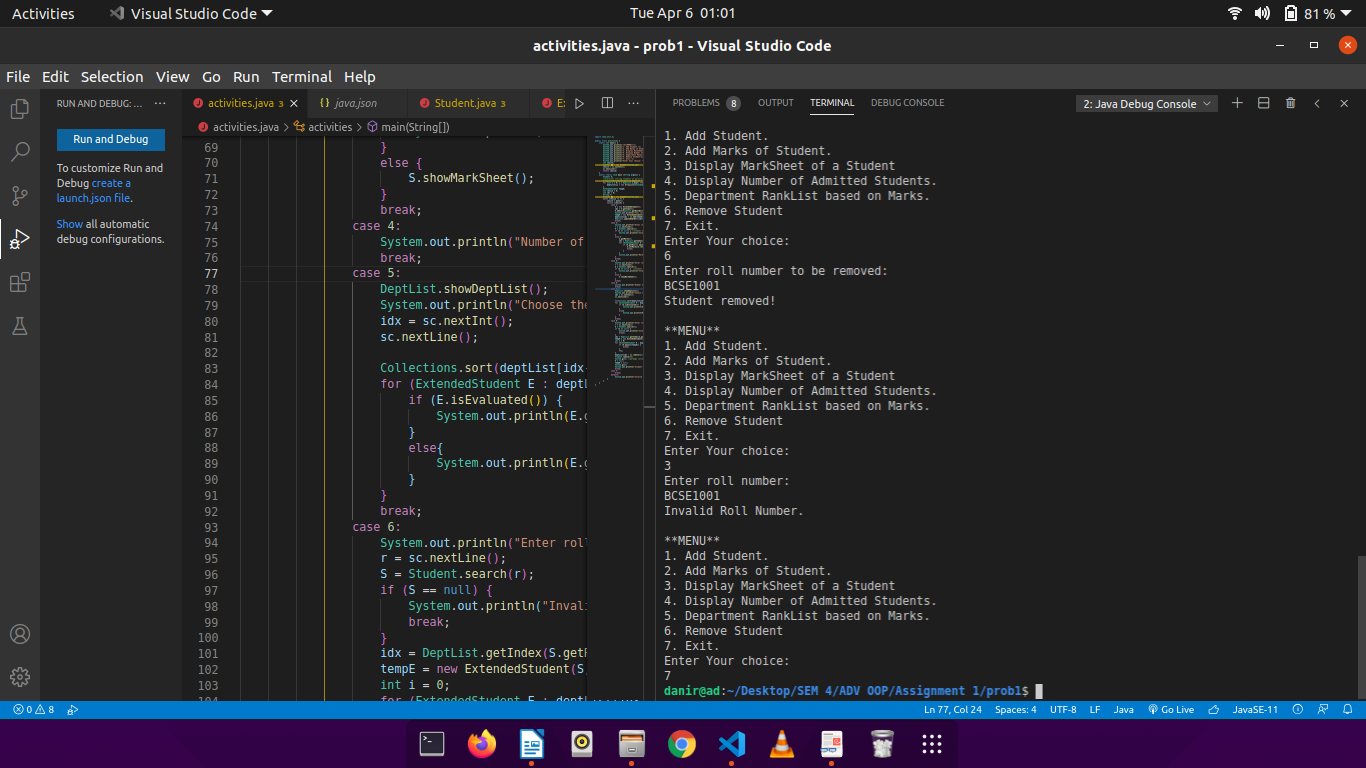
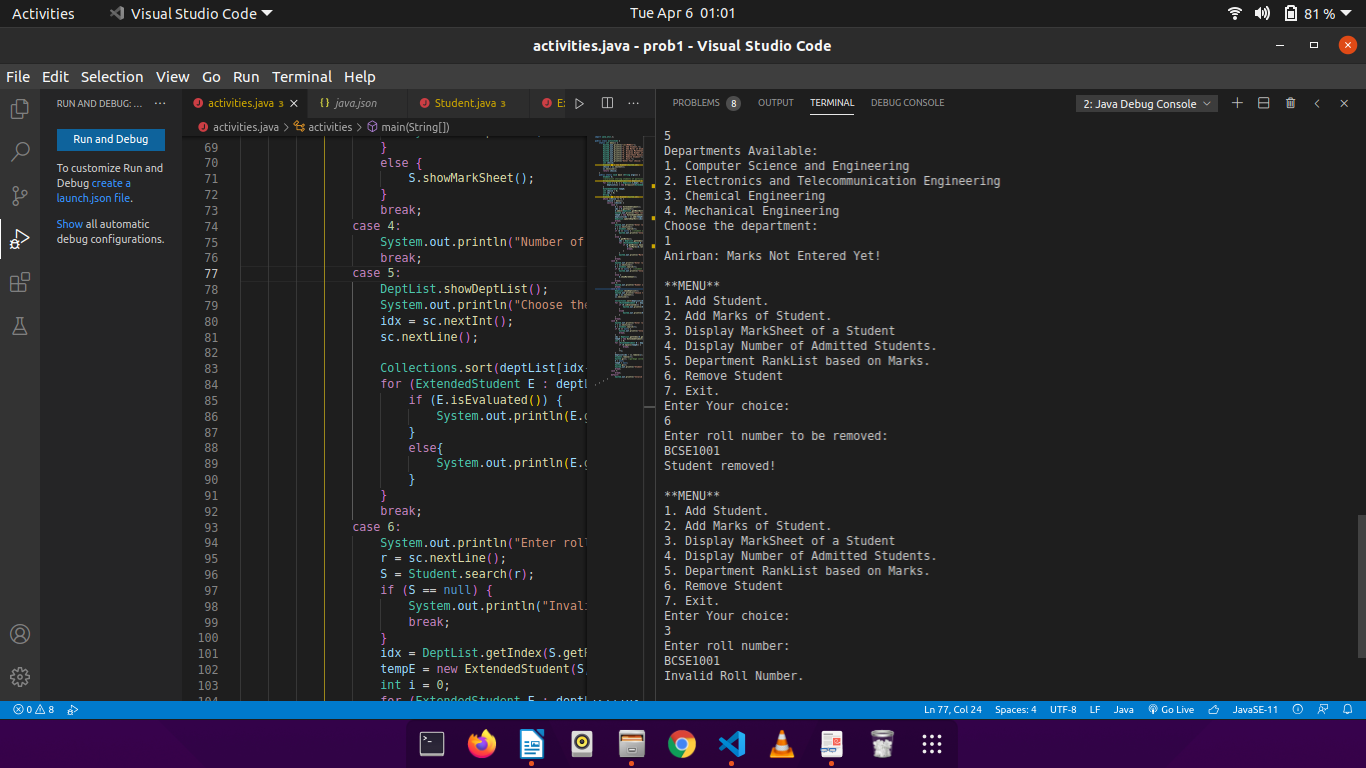
}

}

OUTPUT:







Problem 2

CODE:

**Item.java**

import java.util.Scanner;

// store item information

public class Item {

private String item\_code;

private String name;

private int rate;

private int quantity;

// constructor

public Item(){

item\_code = "";

name = "";

rate = -1;

quantity = 0;

}

// accept data from user

public void getData(){

Scanner sc = new Scanner(System.in);

System.out.println("Enter Name: ");

name = sc.nextLine();

System.out.println("Enter Rate: ");

rate = sc.nextInt();

sc.nextLine();

quantity++;

}

// display

public void showData(){

System.out.println("\nItem Code: " + item\_code);

System.out.println("Item Name: " + name);

System.out.println("Item Price: " + rate);

System.out.println("Quantity Available: " + quantity);

}

// modifier function to change rate

public void changeRate(int r){

rate = r;

}

// generate item\_code automatically in the format ITEMNAME + 00 + SEREALNUMBER

public void generateItemCode(int n){

String temp;

if(n<10){

temp = "00";

}

else{

temp = "0";

}

temp = temp + String.valueOf(n);

item\_code = name.substring(0,3).toUpperCase() + temp;

}

// modify quantity for issue and receive

public void changeQuantity(int isIssue, int items){

if(isIssue == 1){

quantity = quantity - items;

}

else{

quantity = quantity + items;

}

}

// check if item is available

public boolean isAvailable(int items){

return ((quantity-items) > 0);

}

// accessor function to return name

public String returnName(){

return name;

}

// accessor function to return code

public String returnCode(){

return item\_code;

}

// accessor function to return rate

public int returnRate(){

return rate;

}

}

**ItemList.java**

import java.util.\*;

// store list of items (single list of electronics, furniture, etc)

public class ItemList {

ArrayList<Item> i\_list = new ArrayList<Item>();

static int items\_added = 0;

public ItemList(){

}

// add an item to the list

public void addItem(){

Scanner sc = new Scanner(System.in);

Item it = new Item();

it.getData();

if(this.isNamePresent(it.returnName())){

it = this.replaceDuplicate(it.returnName());

this.changeQuantity(it);

}

else{

items\_added++;

it.generateItemCode(items\_added);

i\_list.add(it);

System.out.println("Auto-generated Item Code: " + it.returnCode());

System.out.println("Item Added!");

}

}

// change the rate of an item

public void changeRate(String code, int r){

for(Item i: i\_list){

if(i.returnCode().equals(code))

i.changeRate(r);

}

System.out.println("Rate Changed!");

}

// show data of a particular item

public void showDetails(String code){

for(Item i: i\_list){

if(i.returnCode().equals(code))

i.showData();

}

}

// issue an item

public void issue(int number, String code){

for(Item i: i\_list){

if(i.returnCode().equals(code)){

if(i.isAvailable(number)){

i.changeQuantity(1, number);

int index = this.getIndexInList(i.returnCode());

i\_list.set(index-1, i);

System.out.println("Items Issued!");

}

else{

System.out.println("Item not available for issue, kindly re-stock!");

}

}

}

}

// receive an item

public void receive(int number, String code){

for(Item i: i\_list){

if(i.returnCode().equals(code)){

i.changeQuantity(0, number);

int index = this.getIndexInList(i.returnCode());

i\_list.set(index-1, i);

System.out.println("Items Received!");

}

}

}

// display on condition

public void displayElementsWithMinimumPrice(int price){

Scanner sc = new Scanner(System.in);

int count = 0;

for(Item i: i\_list){

if(i.returnRate() > price){

count++;

}

}

System.out.println("There are " + count + " items that match the criteria.");

String isYes = "";

System.out.println("Want to see details of items? (Yes/No)");

isYes = sc.nextLine();

if(isYes.equals("Yes")){

for(Item i: i\_list){

if(i.returnRate() > price){

i.showData();

}

}

}

else return;

}

// utility functions

// check if item is alreaady present

public boolean isNamePresent(String temp){

for(Item i: i\_list){

if(i.returnName().equals(temp))

return true;

}

return false;

}

// check if code is in the list or not

public boolean isCodePresent(String temp){

for(Item i: i\_list){

if(i.returnCode().equals(temp))

return true;

}

return false;

}

// get the index of item in the list

public int getIndexInList(String code){

if(code.charAt(code.length() - 2) == '0' ){

char lastChar = code.charAt(code.length() - 1);

return Character.getNumericValue(lastChar);

}

String temp = code.substring(4, 5);

return Integer.parseInt(temp);

}

// change the quantity of the item

public void changeQuantity(Item it){

Scanner sc = new Scanner(System.in);

System.out.println("Item Already Present. Do you want to update quantity? (Yes/No):");

String temp = sc.nextLine();

if(temp.equals("Yes")){

System.out.println("Enter Amount of Items to be added: ");

int i = sc.nextInt();

sc.nextLine();

it.changeQuantity(0, i);

int index = this.getIndexInList(it.returnCode());

i\_list.set(index-1, it);

System.out.println("Item Quantity Changed!");

}

}

// this function reuturns the Item object if it is already present

public Item replaceDuplicate(String name){

for(Item i: i\_list){

if(i.returnName().equals(name)){

return i;

}

}

return null;

}

}

**Operations.java**

import java.util.Scanner;

public class operations {

// menu 1

static int SeoMenu() {

System.out.println("\n\*\*STOCK ENTRY OPERATOR MENU\*\*");

System.out.println("1. Add Item.");

System.out.println("2. Change Rate.");

System.out.println("3. Check Item Details.");

System.out.println("4. Issue Item.");

System.out.println("5. Receive Item.");

System.out.println("6. Check Items with Minimum Price.");

System.out.println("7. Exit.");

System.out.println("Enter Your choice: ");

int choice;

Scanner sc = new Scanner(System.in);

choice = sc.nextInt();

sc.nextLine();

return choice;

}

// menu 2

static int SkMenu() {

System.out.println("\n\*\*SHOPKEPER MENU\*\*");

System.out.println("1. Issue Item.");

System.out.println("2. Receive Item.");

System.out.println("3. Check Item Details.");

System.out.println("4. Check Items with Minimum Price.");

System.out.println("5. Exit.");

System.out.println("Enter Your choice: ");

int choice;

Scanner sc = new Scanner(System.in);

choice = sc.nextInt();

sc.nextLine();

return choice;

}

public static void main(String args[]){

ItemList IL = new ItemList(); //instance of ItemList class

int choice = 0;

int tempInt = 0;

String tempString = "";

Scanner sc = new Scanner(System.in);

while(!tempString.equals("STOP")){

System.out.println("\nStock Entry Operator(SEO) OR Shopkeeper(SK) OR STOP ? ");

tempString = sc.nextLine();

if(tempString.equals("SEO")){

while (choice != 7) {

choice = SeoMenu();

switch (choice) {

case 1:

IL.addItem();

break;

case 2:

System.out.println("Enter Item Code:");

tempString = sc.nextLine();

// check if code is already present

if(IL.isCodePresent(tempString)){

System.out.println("Enter New Rate: ");

tempInt = sc.nextInt();

sc.nextLine();

IL.changeRate(tempString, tempInt);

}

else{

System.out.println("Invalid Item Code!");

}

break;

case 3:

System.out.println("Enter Item Code:");

tempString = sc.nextLine();

// check if code is already present

if(IL.isCodePresent(tempString)){

IL.showDetails(tempString);

}

else{

System.out.println("Invalid Item Code!");

}

break;

case 4:

System.out.println("Enter Item Code:");

tempString = sc.nextLine();

if(IL.isCodePresent(tempString)){

System.out.println("Enter quantity to be Issued:");

tempInt = sc.nextInt();

sc.nextLine();

IL.issue(tempInt, tempString);

}

else{

System.out.println("Invalid Item Code!");

}

break;

case 5:

System.out.println("Enter Item Code:");

tempString = sc.nextLine();

if(IL.isCodePresent(tempString)){

System.out.println("Enter quantity to be Received:");

tempInt = sc.nextInt();

sc.nextLine();

IL.receive(tempInt, tempString);

}

else{

System.out.println("Invalid Item Code!");

}

break;

case 6:

System.out.println("Enter Minimum Price:");

tempInt = sc.nextInt();

sc.nextLine();

IL.displayElementsWithMinimumPrice(tempInt);

break;

case 7:

break;

default:

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

}

}

}

else if(tempString.equals("SK")){

while (choice != 5) {

choice = SkMenu();

switch (choice) {

case 1:

System.out.println("Enter Item Code:");

tempString = sc.nextLine();

if(IL.isCodePresent(tempString)){

System.out.println("Enter Number to be Issued:");

tempInt = sc.nextInt();

sc.nextLine();

IL.issue(tempInt, tempString);

}

else{

System.out.println("Invalid Item Code!");

}

break;

case 2:

System.out.println("Enter Item Code:");

tempString = sc.nextLine();

if(IL.isCodePresent(tempString)){

System.out.println("Enter Number to be Received:");

tempInt = sc.nextInt();

sc.nextLine();

IL.receive(tempInt, tempString);

}

else{

System.out.println("Invalid Item Code!");

}

break;

case 3:

System.out.println("Enter Item Code:");

tempString = sc.nextLine();

if(IL.isCodePresent(tempString)){

IL.showDetails(tempString);

}

else{

System.out.println("Invalid Item Code!");

}

break;

case 4:

System.out.println("Enter Minimum Price:");

tempInt = sc.nextInt();

sc.nextLine();

IL.displayElementsWithMinimumPrice(tempInt);

break;

case 5:

break;

default:

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

}

}

}

else if(!tempString.equals("STOP") && !tempString.equals("SK") && ! tempString.equals("SEO")){

System.out.println("No More Users Available! Kindly contact the Shop Owner");

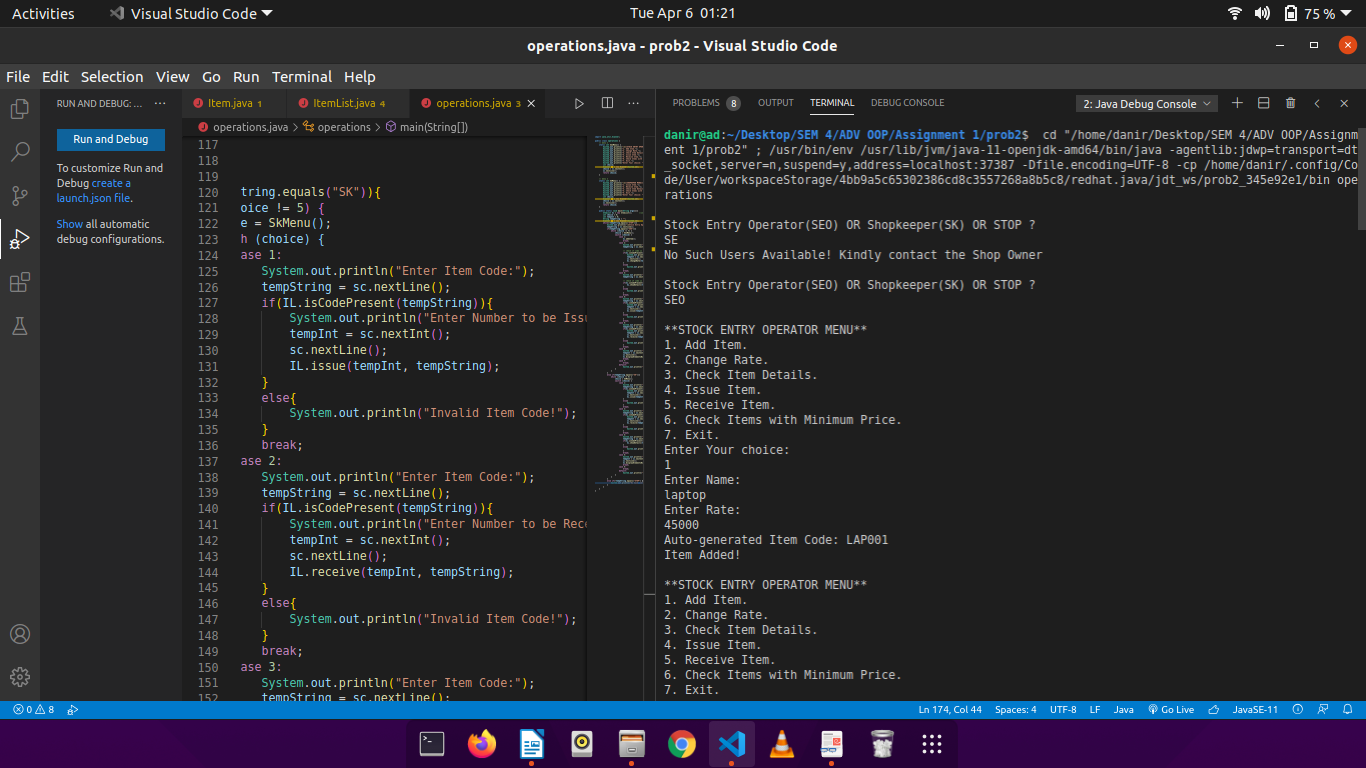
}

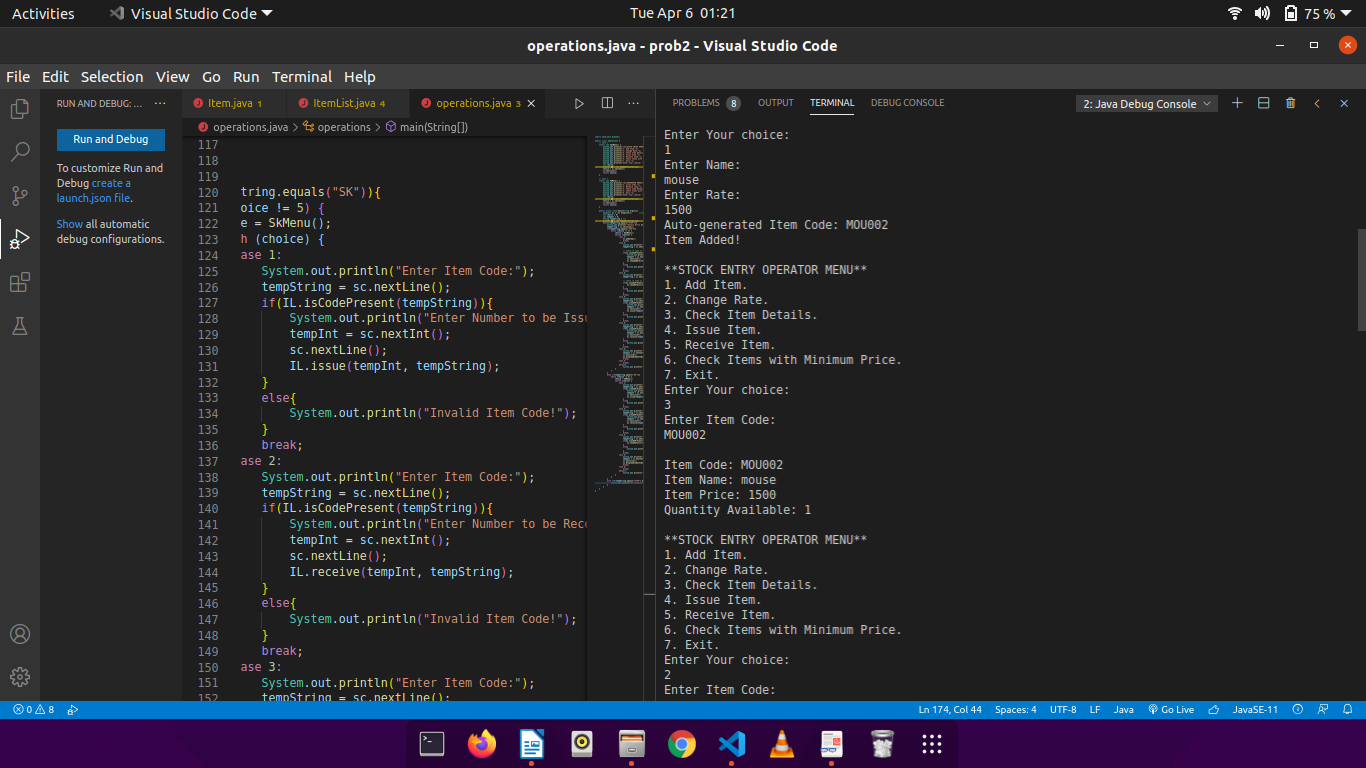
}

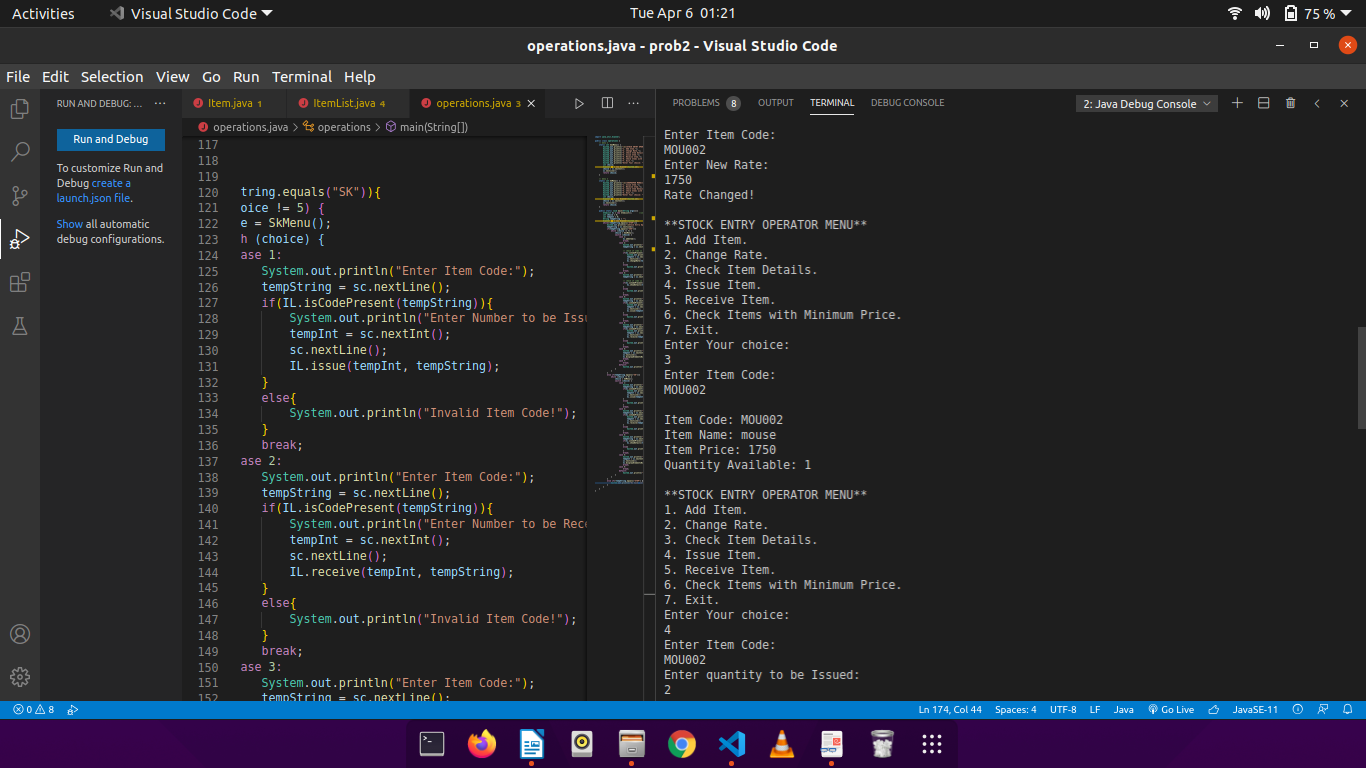
}

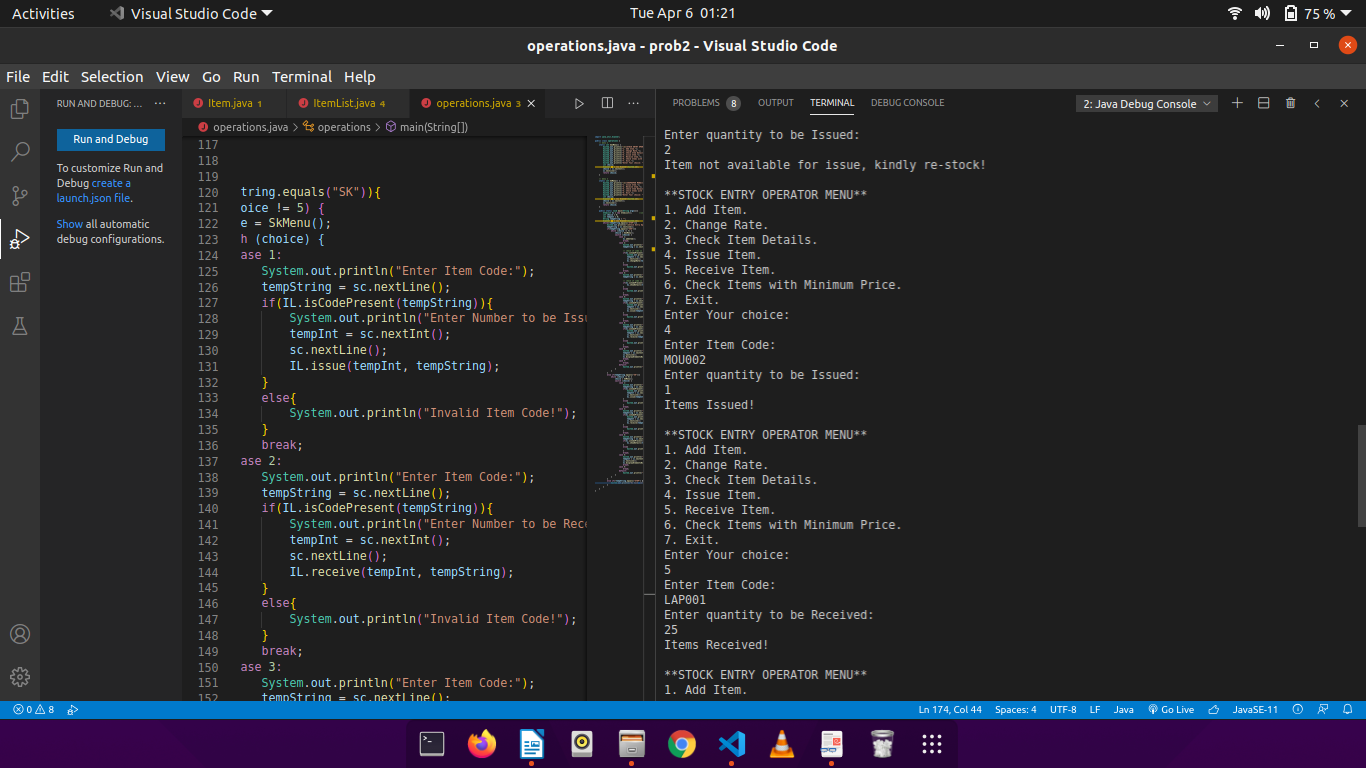
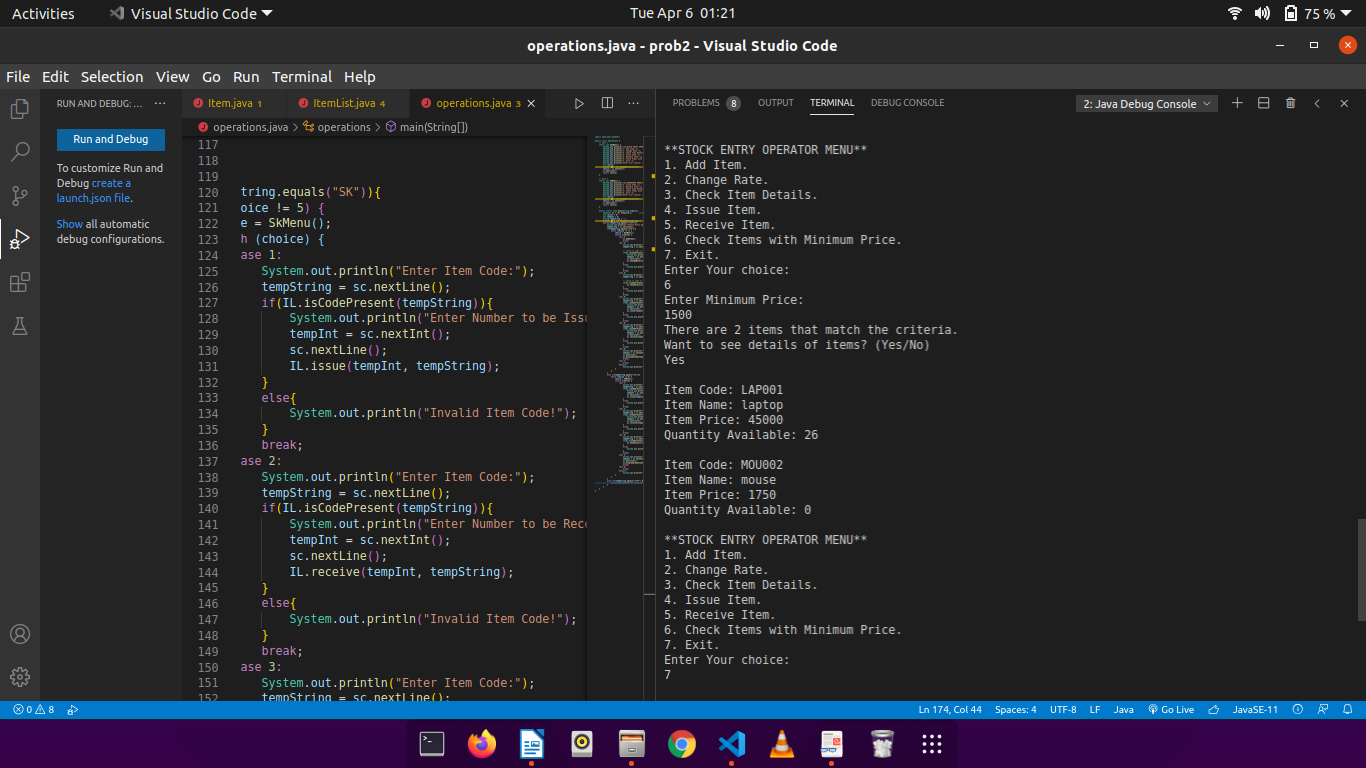
}

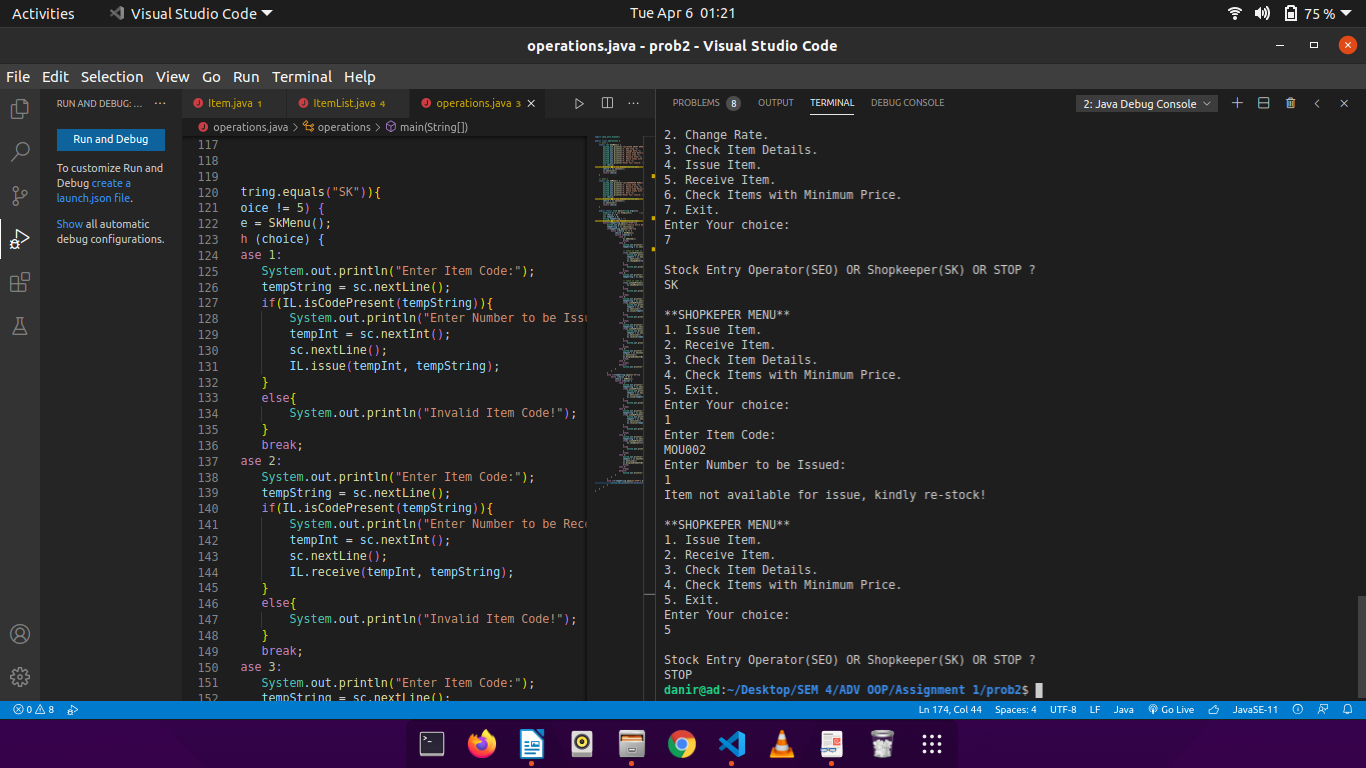
OUTPUT:









Problem 3:

CODE:

**MyStack.java**

public class my\_stack{ //stack implementation using array

private int max;

private char[] arr; //need to make generic

private int top;

public my\_stack(int s){ //constructor

max = s;

top = -1;

arr = new char[max];

}

public void push(char c){ //add data

arr[++top] = c;

}

public char pop(){

return arr[top--];

}

public char return\_top(){ //returns top element

return arr[top];

}

//check for underflow and overflow

public boolean isEmpty(){

return top == -1 ;

}

public boolean isFull(){

return top == (max -1);

}

}

**Parenthesis.java**

import java.util.Scanner;

public class parenthesis {

public static void main(String args[]){

String expression;

Scanner sc = new Scanner(System.in);

System.out.println("Enter the expression to be checked: "); //takes expression with parenthesis

expression = sc.nextLine();

if(isBalanced(expression))

System.out.println("TRUE (The expression is balanced!)");

else

System.out.println("FALSE (The expression is not balanced!)");

sc.close();

}

//function returns true if the expression is balanced

public static boolean isBalanced(String exp){

my\_stack s = new my\_stack(exp.length());

char present\_char, popped\_char;

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

present\_char = exp.charAt(i);

if(present\_char == '(' || present\_char == '{' || present\_char == '[' ){ //adds if opening bracket is encountered

s.push(present\_char);

continue;

}

if(s.isEmpty())

return false;

popped\_char = s.pop();

switch(present\_char){

case ')':

if(popped\_char == '}' || popped\_char == ']')

return false;

break;

case '}':

if(popped\_char == ']' || popped\_char == ')')

return false;

break;

case ']':

if(popped\_char == '}' || popped\_char == ')')

return false;

break;

}

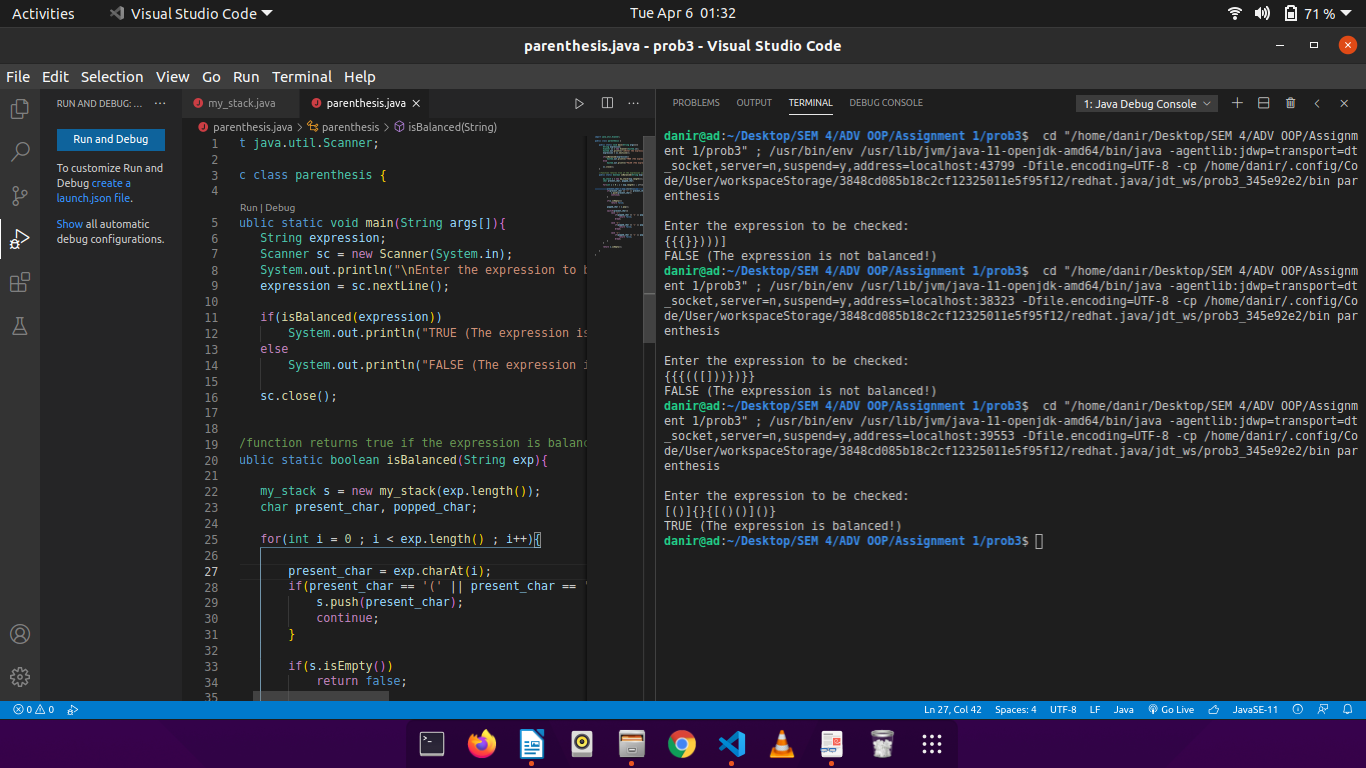
}

return s.isEmpty();

}

}

OUTPUT:



Problem 4:

CODE:

**Quotes.java**

// stores hard coded data set of strings

public class Quotes{

String[] quotes = new String[]{

"'You have to write the book that wants to be written. And if the book will be too difficult for grown-ups, then you write it for children.'--Madeleine L'Engle",

"'If you don't have time to read, you don't have the time (or the tools) to write. Simple as that.'--Stephen King",

"'We write to taste life twice, in the moment and in retrospect.'--Anaïs Nin",

"'Substitute 'damn' every time you're inclined to write 'very;' your editor will delete it and the writing will be just as it should be.'--Mark Twain",

"'If there's a book that you want to read, but it hasn't been written yet, then you must write it.'--Toni Morrison",

"'One day I will find the right words, and they will be simple.'--Jack Kerouac, The Dharma Bums",

"'Either write something worth reading or do something worth writing.'--Benjamin Franklin",

"'You never have to change anything you got up in the middle of the night to write.'-- Saul Bellow",

"'No tears in the writer, no tears in the reader. No surprise in the writer, no surprise in the reader.--Robert Frost",

"'Read, read, read. Read everything -- trash, classics, good and bad, and see how they do it. Just like a carpenter who works as an apprentice and studies the master. Read! You'll absorb it. Then write. If it's good, you'll find out. If it's not, throw it out of the window.'-- William Faulkner",

"'You must stay drunk on writing so reality cannot destroy you.'--Ray Bradbury, Zen in the Art of Writing",

"'Words can be like X-rays if you use them properly -- they'll go through anything. You read and you're pierced.'--Aldous Huxley, Brave New World",

"'How vain it is to sit down to write when you have not stood up to live.'--Henry David Thoreau",

"'I can shake off everything as I write; my sorrows disappear, my courage is reborn.'--Anne Frank",

"'A writer is someone for whom writing is more difficult than it is for other people.'-- Thomas Mann, Essays of Three Decades",

"'Let me live, love, and say it well in good sentences.'--Sylvia Plath, The Unabridged Journals of Sylvia Plath",

"'Here is a lesson in creative writing. First rule: Do not use semicolons. They are transvestite hermaphrodites representing absolutely nothing. All they do is show you've been to college.'--Kurt Vonnegut Jr., A Man Without a Country",

"'Don't bend; don't water it down; don't try to make it logical; don't edit your own soul according to the fashion. Rather, follow your most intense obsessions mercilessly.'-- Franz Kafka",

"'I kept always two books in my pocket, one to read, one to write in.'--Robert Louis Stevenson",

"'You can make anything by writing.'--C.S. Lewis",

"'A word after a word after a word is power.'--Margaret Atwood",

"'Tears are words that need to be written.'--Paulo Coelho",

"'You should write because you love the shape of stories and sentences and the creation of different words on a page. Writing comes from reading, and reading is the finest teacher of how to write.'--Annie Proulx",

"'To survive, you must tell stories.'--Umberto Eco, The Island of the Day Before",

"'Always be a poet, even in prose.'--Charles Baudelaire",

"'If my doctor told me I had only six minutes to live, I wouldn't brood. I'd type a little faster.'--Isaac Asimov",

"'The purpose of a writer is to keep civilization from destroying itself.'--Albert Camus",

"'I write to discover what I know.'--Flannery O'Connor",

"'Ideas are like rabbits. You get a couple and learn how to handle them, and pretty soon you have a dozen.'― John Steinbeck",

"'Words do not express thoughts very well. They always become a little different immediately after they are expressed, a little distorted, a little foolish.'― Hermann Hesse",

"'Almost anyone can be an author; the business is to collect money and fame from this state of being.'--A. A. Milne",

"'A blank piece of paper is God's way of telling us how hard it is to be God.'--Sidney Sheldon"

};

public Quotes(){ //default constructor

}

}

**QuoteOfTheDay.java**

import java.lang.Math;

public class QuoteOfTheDay {

public static void main(String args[]){

Quotes q = new Quotes(); //instance of class Quotes

int length = q.quotes.length; //stores length of hardcoded string array

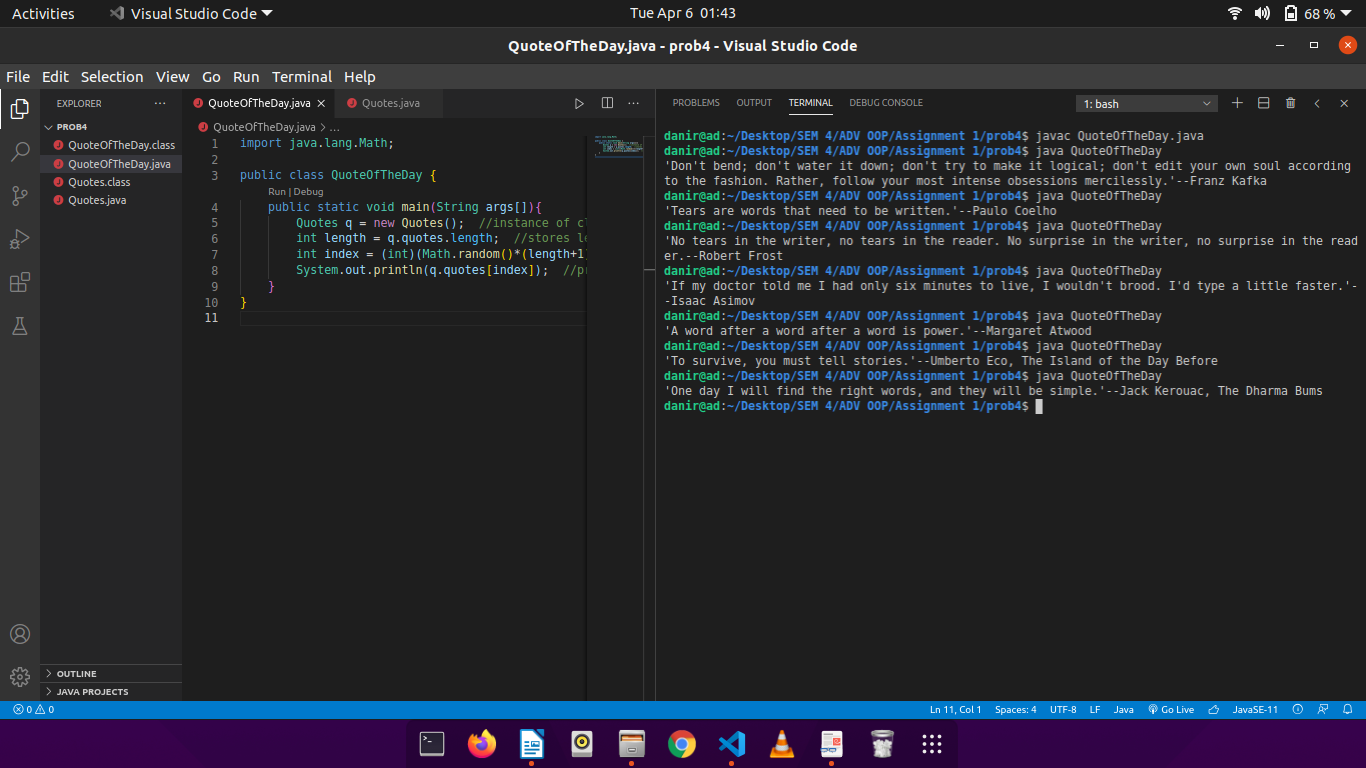
int index = (int)(Math.random()\*(length+1)); //generated random integer between 0 and length

System.out.println(q.quotes[index]); //prints random string

}

}

OUTPUT:



Problem 5:

CODE:

**IndexingBook.java**

import java.io.\*;

import java.util.\*;

public class IndexingBook{

public static void main(String args[]) throws IOException {

File fobj = new File("test.txt"); //opening a file (creating if is not already present)

Scanner sc = new Scanner(fobj);

Map<String, List<Integer>> hm = new HashMap<>(); //HashMap to store <word, line \ numbers> pairs

HashSet<String> uniqueWords; //HashSet of strings to store unique words from each line

String[] words;

int line\_number = 0;

ArrayList<Integer> lines; //array of integers to store line numbers of each unique word

System.out.println();

while(sc.hasNextLine()){

line\_number++;

String line = sc.nextLine();

words = line.split("\\s+|\\."); //storing words of each line in an array

uniqueWords = new HashSet<String>(Arrays.asList(words));

for(String s : uniqueWords ){ //adding values to the hashMap

if(hm.containsKey(s)){

hm.get(s).add(line\_number);

}

else{

lines = new ArrayList<Integer>();

lines.add(line\_number);

hm.put(s, lines);

}

}

}

//making a hashTable from the existing hashMap

Hashtable<String, List<Integer>> HT = new Hashtable<String, List<Integer>>(hm);

//sorting the keys in alphabetical order

List<String> temp = Collections.list(HT.keys());

Collections.sort(temp, String.CASE\_INSENSITIVE\_ORDER);

Iterator<String> it = temp.iterator();

while(it.hasNext()){

String s = it.next();

System.out.println(s + ": " + HT.get(s));

}

}

}

**Test.txt**

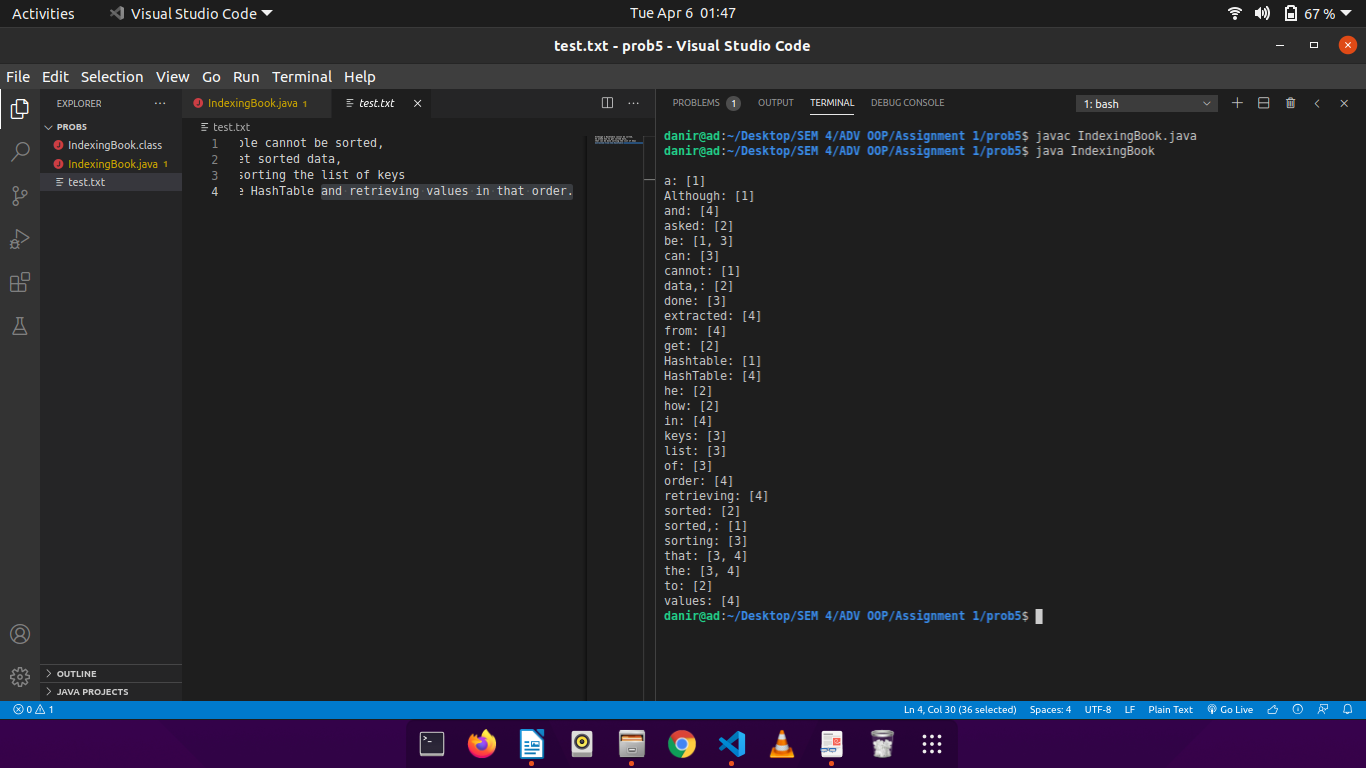
Although a Hashtable cannot be sorted,

he asked how to get sorted data,

that can be done sorting the list of keys

extracted from the HashTable and retrieving values in that order.

OUTPUT:



Problem 6:

CODE:

**Patient.java**

import java.util.Scanner;

// store patient information

public class Patient{

private String patient\_id;

private String name;

private int age;

private String address;

private String contact\_no;

private Doctor doctor;

private float blood\_pressure;

private int temparature; //in kelvin

private boolean hasADoctor;

// constructor

public Patient(){

doctor = null;

hasADoctor = false;

}

// get data from the user

public void getData(){

Scanner sc = new Scanner(System.in);

System.out.println("Enter Patient Name: ");

name = sc.nextLine();

name = name.substring(0, 1).toUpperCase() + name.substring(1);

System.out.println("Enter Patient Age: ");

age = sc.nextInt();

sc.nextLine();

System.out.println("Enter Patient Address: ");

address = sc.nextLine();

System.out.println("Enter Paatient Contact No.: ");

contact\_no = sc.nextLine();

System.out.println("Enter Patient Blood Pressure:");

blood\_pressure = sc.nextFloat();

sc.nextLine();

System.out.println("Enter Patient Temparature:");

temparature = sc.nextInt();

sc.nextLine();

}

// display

public void showData(){

System.out.println("\nPatient ID: " + patient\_id);

System.out.println("Patient Name: " + name);

System.out.println("Temparature: " + temparature);

System.out.println("Blood Pressure: " + blood\_pressure);

}

// assign a doctor with the patient

public void assignDoctor(Doctor d){

doctor = d;

hasADoctor = true;

}

// check if the patient is already assigned a doctor

public boolean isDoctorAssigned(){

return hasADoctor;

}

// generate the patient code automatically in a specific format

public void generatePatientID(String name, int number){

patient\_id = "PAT\_" + name.substring(0, 3).toUpperCase() + "\_00" + String.valueOf(number);

}

// accessor function to return ID

public String returnID(){

return patient\_id;

}

// accessor function to return name

public String returnName(){

return name;

}

}

**Doctor.java**

import java.util.\*;

// store doctor data

public class Doctor {

private String doc\_id;

private String name;

private String address;

private String contact\_no;

private ArrayList<Patient> patient\_list = new ArrayList<Patient>();

public Doctor(){

}

// get doctor data from the user

public void getData(){

Scanner sc = new Scanner(System.in);

System.out.println("Enter Doctor Name: ");

name = sc.nextLine();

name = name.substring(0, 1).toUpperCase() + name.substring(1);

System.out.println("Enter Doctor Address: ");

address = sc.nextLine();

System.out.println("Enter Doctor Contact No.: ");

contact\_no = sc.nextLine();

}

// automatically generate Doctor ID

public void generateDocID(String name, int number){

doc\_id = "DR\_" + name.substring(0,3).toUpperCase() + "\_00" + String.valueOf(number);

}

// assign patient to a particular doctor

public String addPatient(String name, Patient p, int n){

patient\_list.add(p);

patient\_list.get(patient\_list.size() - 1).assignDoctor(this);

patient\_list.get(patient\_list.size() - 1).generatePatientID(name, n);

return patient\_list.get(patient\_list.size() - 1).returnID();

}

// check if patient is assigned to a particular doctor

public boolean doesPatientExists(String ID){

for(Patient p : patient\_list){

if(ID.equals(p.returnID())){

return true;

}

}

return false;

}

// show data of a single patient, along with doctor assigned

public void showSinglePatientData(String ID){

for(Patient p : patient\_list){

if(ID.equals(p.returnID())){

p.showData();

}

}

}

// display doctor info, along with assigned patients

public void showDoctorData(){

System.out.println("DoctorID: " + doc\_id);

System.out.println("Name: Dr. " + name);

System.out.println("Contact No.: " + contact\_no);

}

// show all patients under a particular doctor

public void showAllPatients(){

if(patient\_list.isEmpty()){

System.out.println("No Patients assigned yet!");

return;

}

for(Patient p : patient\_list){

p.showData();

}

}

// accessor function to return name

public String returnName(){

return name;

}

// accessor function to return ID

public String returnID(){

return doc\_id;

}

}

**Hospital.java**

import java.util.\*;

import java.lang.\*;

public class Hospital {

// menu 1

static int menu(){

System.out.println("\n\*\*HOSPITAL RECEPTION MENU\*\*");

System.out.println("1. Add Doctor.");

System.out.println("2. Add Patient.");

System.out.println("3. Check Patient Details.");

System.out.println("4. Show All Patients under a Doctor.");

System.out.println("5. Show All Doctors.");

System.out.println("6. Exit.");

System.out.println("Enter Your choice: ");

int choice;

Scanner sc = new Scanner(System.in);

choice = sc.nextInt();

sc.nextLine();

return choice;

}

public static void main(String args[]){

ArrayList<Doctor> doc\_list = new ArrayList<Doctor>(); // list of doctors in the hospital

int noOfPatients = 0;

int choice = 0;

int tempInt = -1;

String tempString = "";

Scanner sc = new Scanner(System.in);

while(choice != 6){

choice = menu();

switch(choice){

case 1:

Doctor doc = new Doctor();

doc.getData();

doc.generateDocID(doc.returnName(), doc\_list.size() + 1);

doc\_list.add(doc);

System.out.println("Auto-generated Doctor ID: " + doc.returnID());

break;

case 2:

Patient pat = new Patient();

if(doc\_list.isEmpty()){

System.out.println("No Doctors Available! Kindly Contact the Management!");

break;

}

// assign the patient to a random doctor

// (might change this feature later, to add doctor specification)

tempInt = (int)(Math.random()\*(doc\_list.size()));

pat.getData();

tempString = pat.returnName();

noOfPatients++;

System.out.println("Auto-generated Patient ID: " + doc\_list.get(tempInt).addPatient(tempString, pat, noOfPatients));

doc\_list.set(tempInt, doc\_list.get(tempInt));

System.out.println("The patient has been assigned to Dr. " + doc\_list.get(tempInt).returnName() + ".");

break;

case 3:

System.out.println("Enter Patient ID:");

tempString = sc.nextLine();

int flag = 0;

for(Doctor d : doc\_list){

if(d.doesPatientExists(tempString) == true){

flag = 1;

d.showSinglePatientData(tempString);

System.out.println("Assigned Doctor:\n");

d.showDoctorData();

break;

}

}

if(flag == 0){

System.out.println("Patient ID does not exists!");

break;

}

break;

case 4:

if(doc\_list.isEmpty()){

System.out.println("No Doctors Available! Kindly Contact the Management!");

break;

}

System.out.println("Enter Doctor ID: ");

tempString = sc.nextLine();

flag = 0;

for(Doctor d : doc\_list ){

if(tempString.equals(d.returnID())){

flag = 1;

d.showAllPatients();

}

}

if(flag == 0){

System.out.println("Doctor ID does not exists!");

break;

}

break;

case 5:

if(doc\_list.isEmpty()){

System.out.println("No Doctors Available! Kindly Contact the Management!");

break;

}

for(Doctor d : doc\_list){

d.showDoctorData();

System.out.println();

}

break;

case 6:

break;

default:

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

}

}

}

}

OUTPUT:

