****

**PROJECT REPORT**

**HOSTEL MANANGENT SYSTEM**

**SUBMITTED TO: MA’AM SANEEHA AMIR**

**SUBMITTED BY:**

**ZAMIN RAZA**

**FA21-BCS-096**

(OBJECT ORIENTED PROGRAMIING)

Aim of the project:

Aim of the project is to create GUI based user friendly management system that can be used in Hostels to keep records.

This management system will be helpful for the Hostel administration to store records of Hostellers, and to display the saved records. This project can be useful in small hostels for activities like Hosteller admission, fee payment, room reservation, etc.

Hostel management system manage all hostel activities like hostel admissions, fees, room, mess allotment, hostel stores & generates related reports for smooth transactions. It is also used to manage monthly bill calculation.

**Key Features**:

* Save records of hostellers as well as the employees working in hostels
* Delete records
* Search specific records
* Room reservation
* Indicate room availability and location of the room.
* Generating bills / Rent calculations.

**Tools and technologies:**

* Java as programming language
* Java swing library for GUI
* Data will be saved in files using File handling.

**Important Functions:**

* Add records
* Delete records
* Update
* Room Creation
* Search on the basis of institute/Room No/Id
* Search if Id already exist
* Fine Updating
* Bill Updating
* Checking Room vacancy

**Code of Important Functions**

import java.io.EOFException;  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.FileNotFoundException;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.ObjectInputStream;  
import java.io.ObjectOutputStream;  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class FileOperations {  
  
public static void writeroom(Object o){  
  
 try {  
 File f = new File("ALLROOMS.txt");  
 ObjectOutputStream oos;  
 if (f.exists()) {  
 oos = new MyobjectoutputStream(new FileOutputStream(f, true));  
 } else {  
 oos = new ObjectOutputStream(new FileOutputStream(f));  
 }  
 oos.writeObject(o);  
 oos.close();  
 } catch (Exception e) {  
 System.*out*.println("Error: ");  
 e.printStackTrace();  
 }  
}

Writing to reserved room file  
  
 public static void writeToreservedFile(Object o) {  
  
 try {  
 File f = new File("reserved3.txt");  
 ObjectOutputStream oos;  
 if (f.exists()) {  
 oos = new MyobjectoutputStream(new FileOutputStream(f, true));  
 } else {  
 oos = new ObjectOutputStream(new FileOutputStream(f));  
 }  
 oos.writeObject(o);  
 oos.close();  
 } catch (Exception e) {  
 System.*out*.println("Error: ");  
 e.printStackTrace();  
 }  
 }  
  
 public static ArrayList<hostelite> readAllreservedfiles(){  
 ArrayList<hostelite> arr = new ArrayList<hostelite>();  
 try {  
 ObjectInputStream is = new ObjectInputStream(new FileInputStream("reserved3.txt"));  
  
 while (true) {  
 hostelite s = (hostelite) is.readObject();  
 arr.add(s);  
  
 }  
  
  
 } catch (ClassNotFoundException | FileNotFoundException c) {  
 System.*out*.println("Exception Found: Class not found || File not found! ");  
 c.printStackTrace();  
 } catch (EOFException e) {  
 System.*out*.println("EOF!");  
  
 } catch (IOException e) {  
// e.printStackTrace();  
 }  
 return arr;  
 }  
  
  
  
  
  
 SEARCH }  
  
 public static hostelite searchbyname(String naam){  
 ArrayList<hostelite> p = *readAllfiles*();  
// p = readAllfiles();  
 for(hostelite i : p){  
 if(i.getName().equalsIgnoreCase(naam)){  
 System.*out*.println("desired data");  
 return i;  
 }  
 else {  
 System.*out*.println("NOT FOUNDED");  
 }  
 }  
 return null;  
 }

Already exist id check:  
  
 public static boolean searchidalready(String id){  
 ArrayList<hostelite> p = *readAllfiles*();  
 boolean f = false;  
// p = readAllfiles();  
 for(hostelite i : p){  
 if(i.getId().equalsIgnoreCase(id)){  
 System.*out*.println("ALREADY EXIST");  
 return true;  
 }  
 else {  
 System.*out*.println("NOT FOUNDED");  
 }  
 }  
 return f;  
 }  
  
 public static void delete(String id) {  
 ArrayList<hostelite> r = *readAllfiles*();  
  
  
 for (hostelite i : r) {  
 if (i.getId().equalsIgnoreCase(id)) {  
 System.*out*.println("founded");  
 r.remove(i);  
 break;  
 }  
 }  
  
 try {  
 try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("ALLSTUDENTS.txt"))) {  
 for (hostelite g : r) {  
 oos.writeObject(g);  
 }  
 }  
 }  
 catch(IOException e){  
 System.*out*.println("ERROR");  
 }  
 }

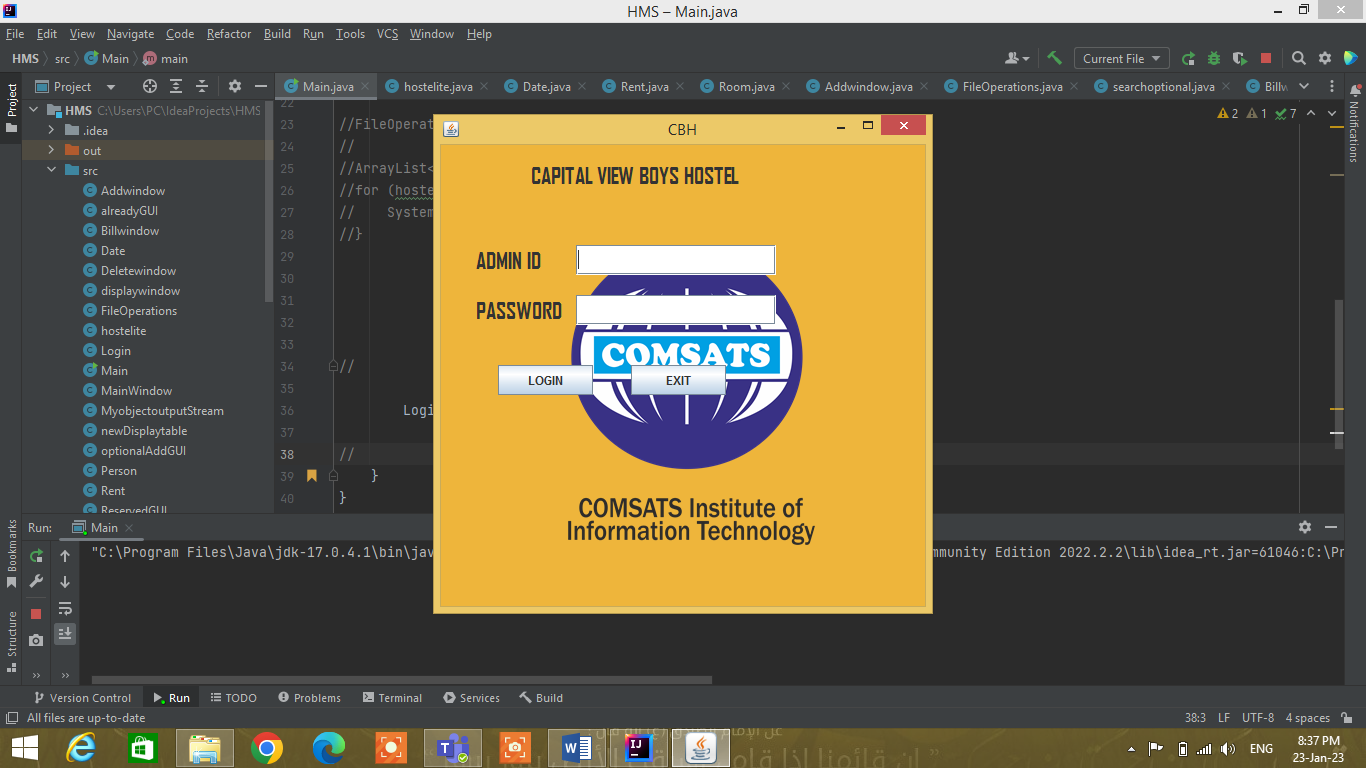
Update Room Number:  
  
  
 public static void updateroom(int check,int number){  
 ArrayList<hostelite> n = *readAllfiles*();  
 for(hostelite g : n){  
 if(g.getRoom().getNumber() == number){  
 g.setRoom(new Room(1,2,number,'B'));  
 break;  
 }  
 }  
 try {  
 try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("ALLSTUDENTS.txt"))) {  
 for (hostelite g : n) {  
 oos.writeObject(g);  
 }  
 }  
 }  
 catch(IOException e){  
 System.*out*.println("ERROR");  
 }  
 }

Deletion:  
  
 public static void delcurrent(int check){  
 ArrayList<Room> n = *readAllRoomsfiles*();  
 for(Room g : n){  
 if(g.getNumber() == check){  
   
 g.setCurr(g.getCurr() - 1);  
   
 break;  
 }  
 }  
 try {  
 try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("ALLROOMS.txt"))) {  
 for (Room g : n) {  
 oos.writeObject(g);  
 }  
 }  
 }  
 catch(IOException e){  
 System.*out*.println("ERROR");  
 }  
 }  
  
Fine And concession updating:

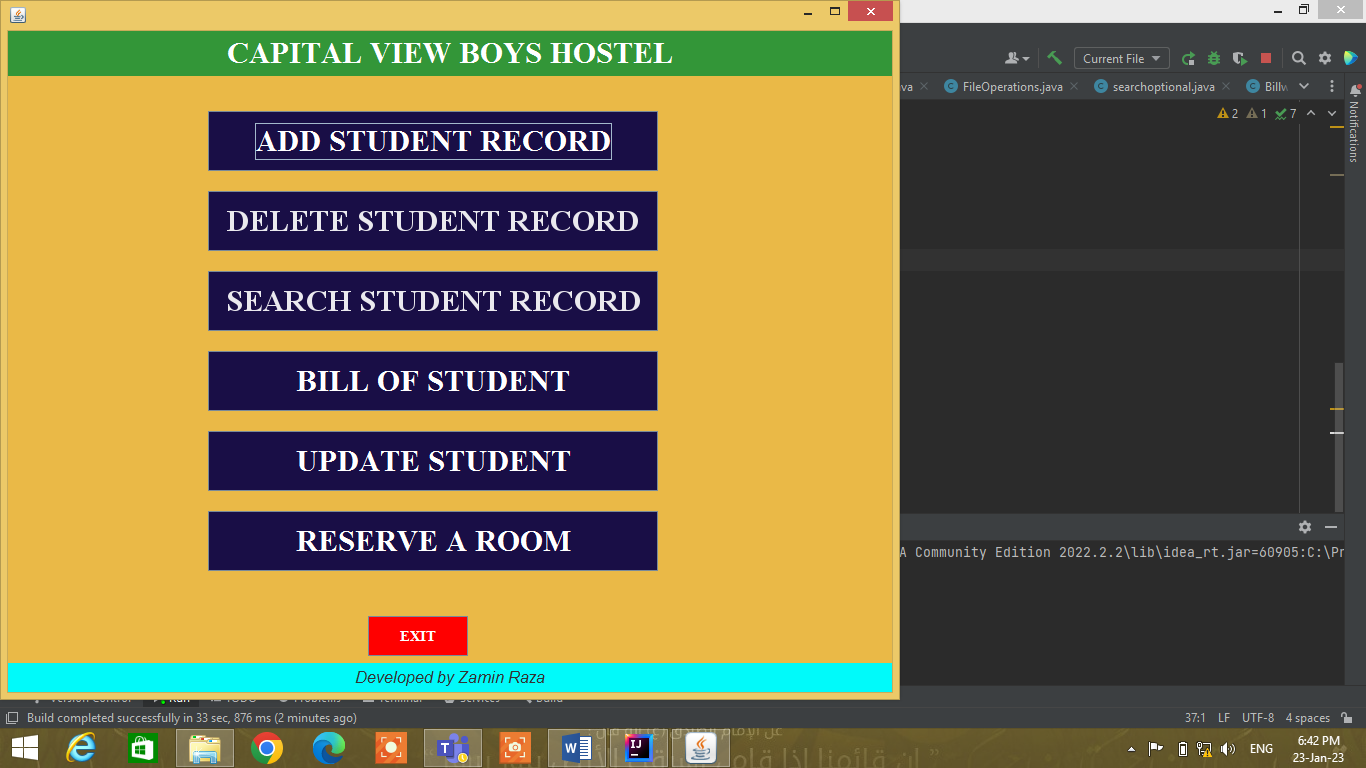
public static void updatefine(String check,String number){  
 ArrayList<hostelite> n = *readAllfiles*();  
 for(hostelite g : n){  
 if(g.getId().equalsIgnoreCase(check)){  
 g.getBill().setFine(Integer.*parseInt*(number));  
 System.*out*.println(g.getBill().getDues());  
 break;  
 }  
 }  
 try {  
 try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("ALLSTUDENTS.txt"))) {  
 for (hostelite g : n) {  
 oos.writeObject(g);  
 }  
 }  
 }  
 catch(IOException e){  
 System.*out*.println("ERROR");  
 }  
 }  
  
 public static void updateconcession(String check,String number){  
 ArrayList<hostelite> n = *readAllfiles*();  
 for(hostelite g : n){  
 if(g.getId().equalsIgnoreCase(check)){  
 g.getBill().setConcession(Integer.*parseInt*(number));  
// System.out.println(g.getBill().getDues());  
 break;  
 }  
 }  
 try {  
 try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("ALLSTUDENTS.txt"))) {  
 for (hostelite g : n) {  
 oos.writeObject(g);  
 }  
 }  
 }  
 catch(IOException e){  
 System.*out*.println("ERROR");  
 }  
 }

**GUI**

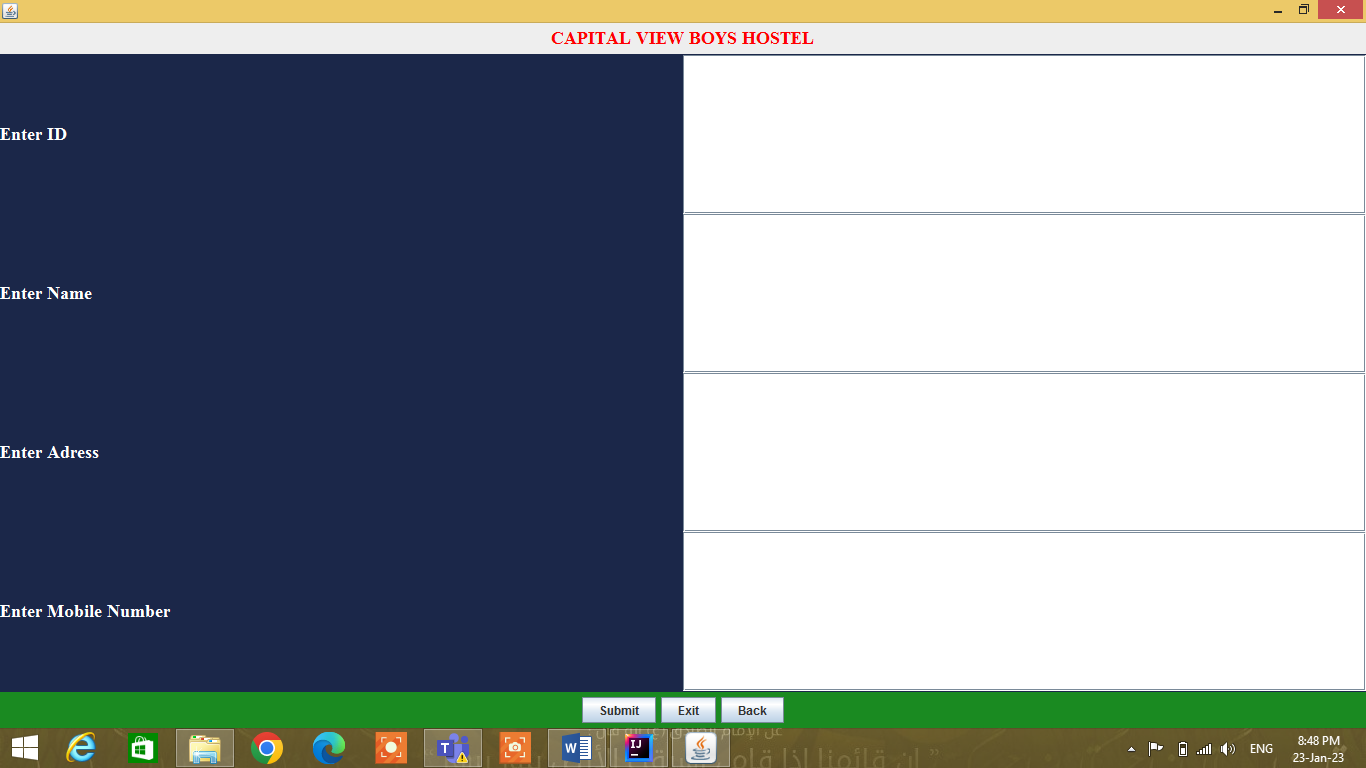
**Log in window**

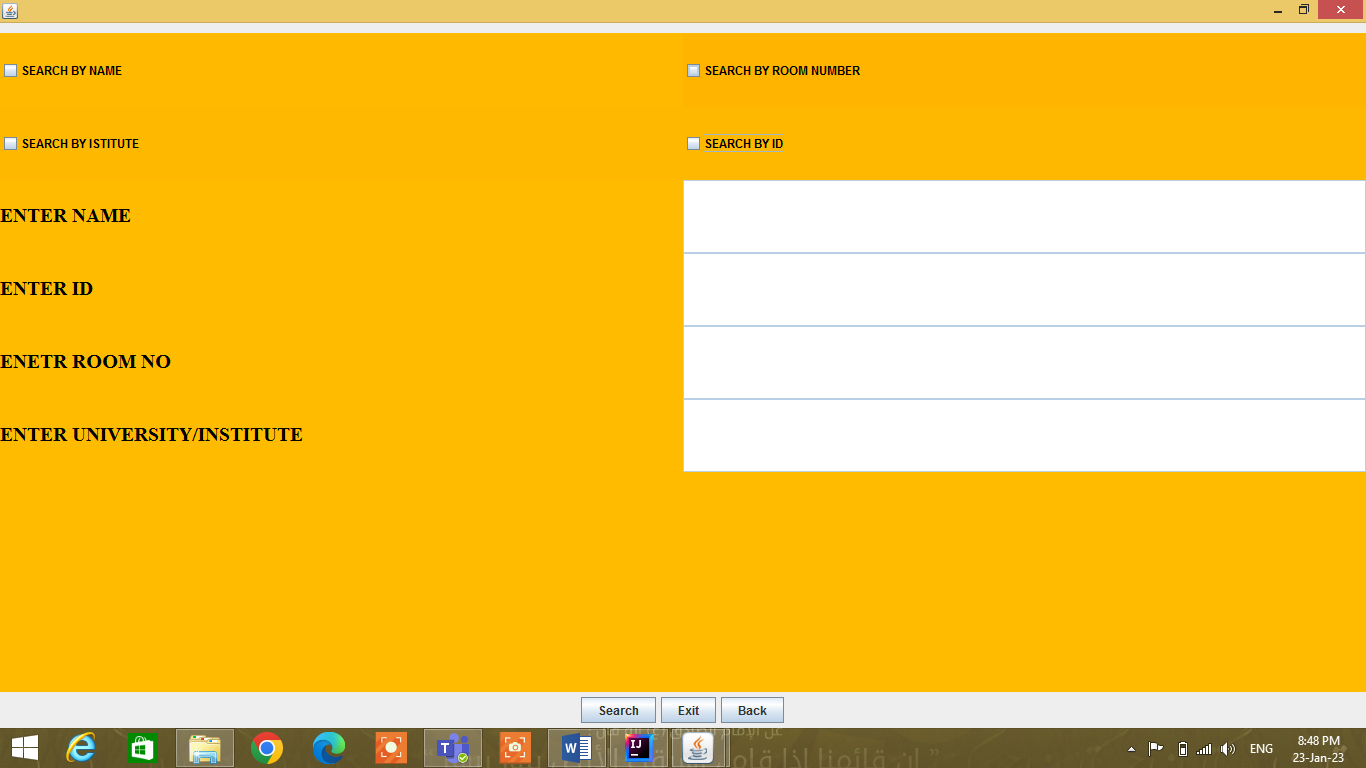
****

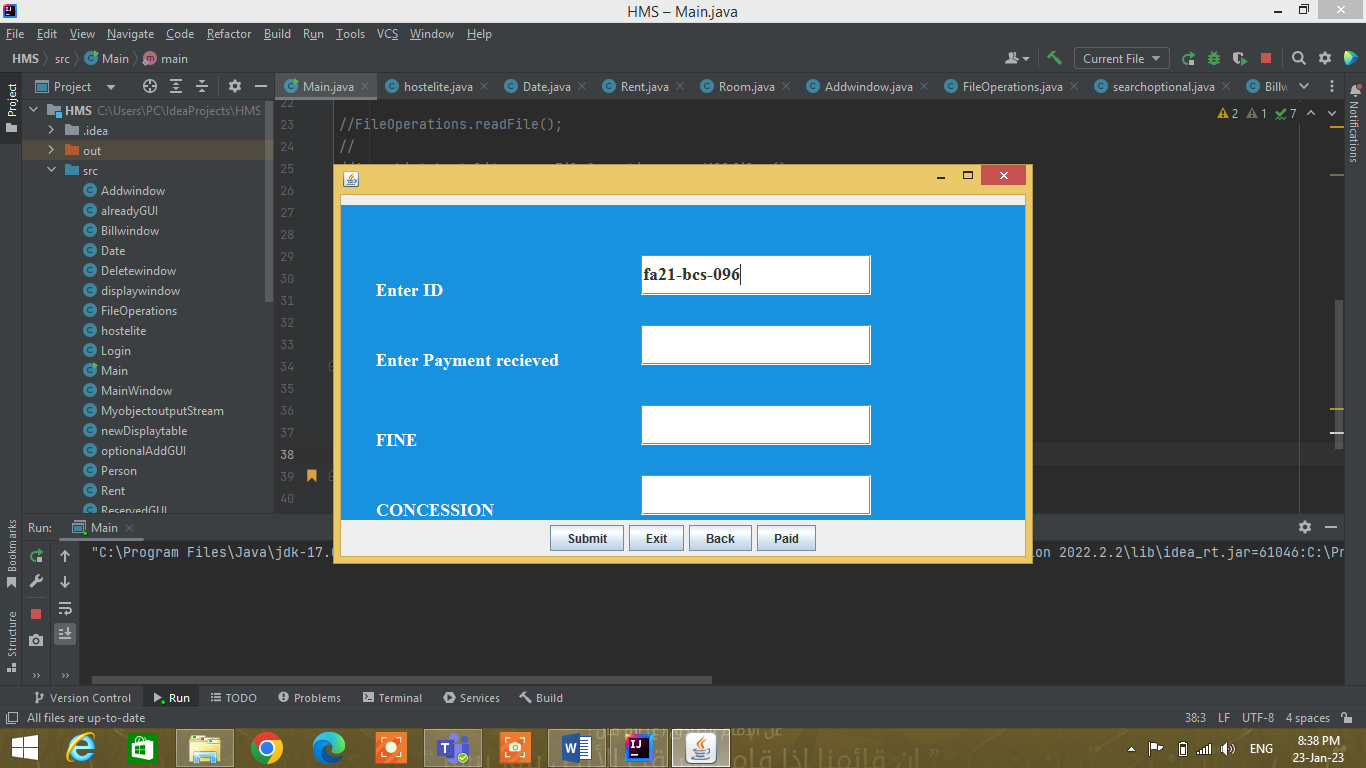
**Main Window**

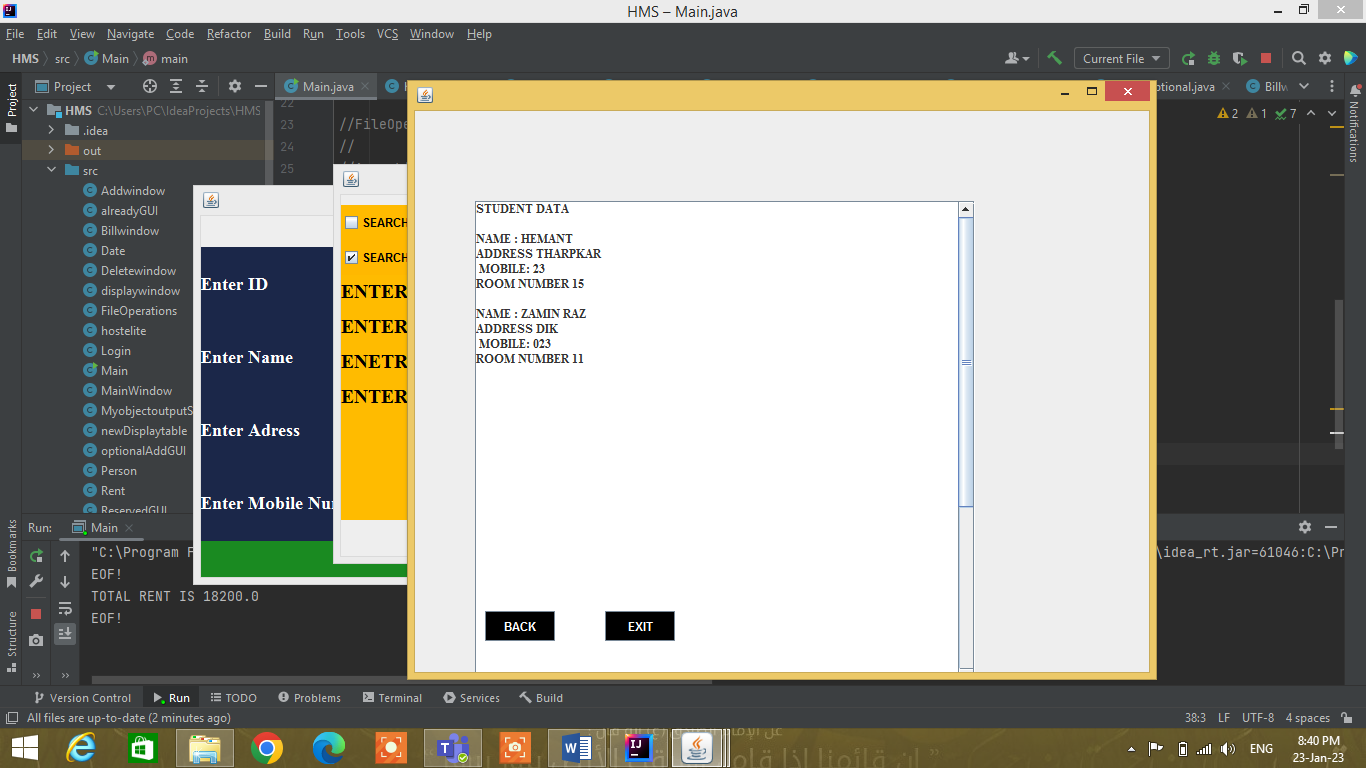
****

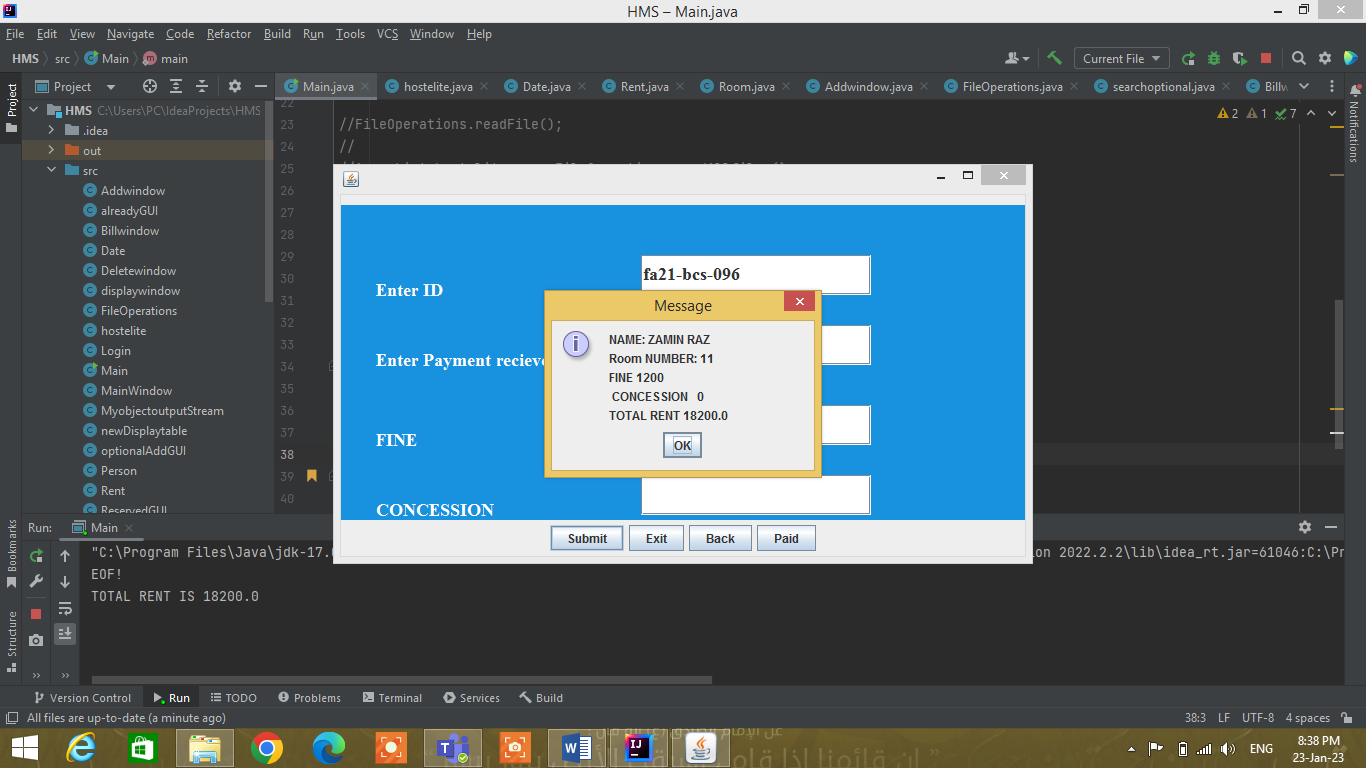
**Other Record entry Windows**

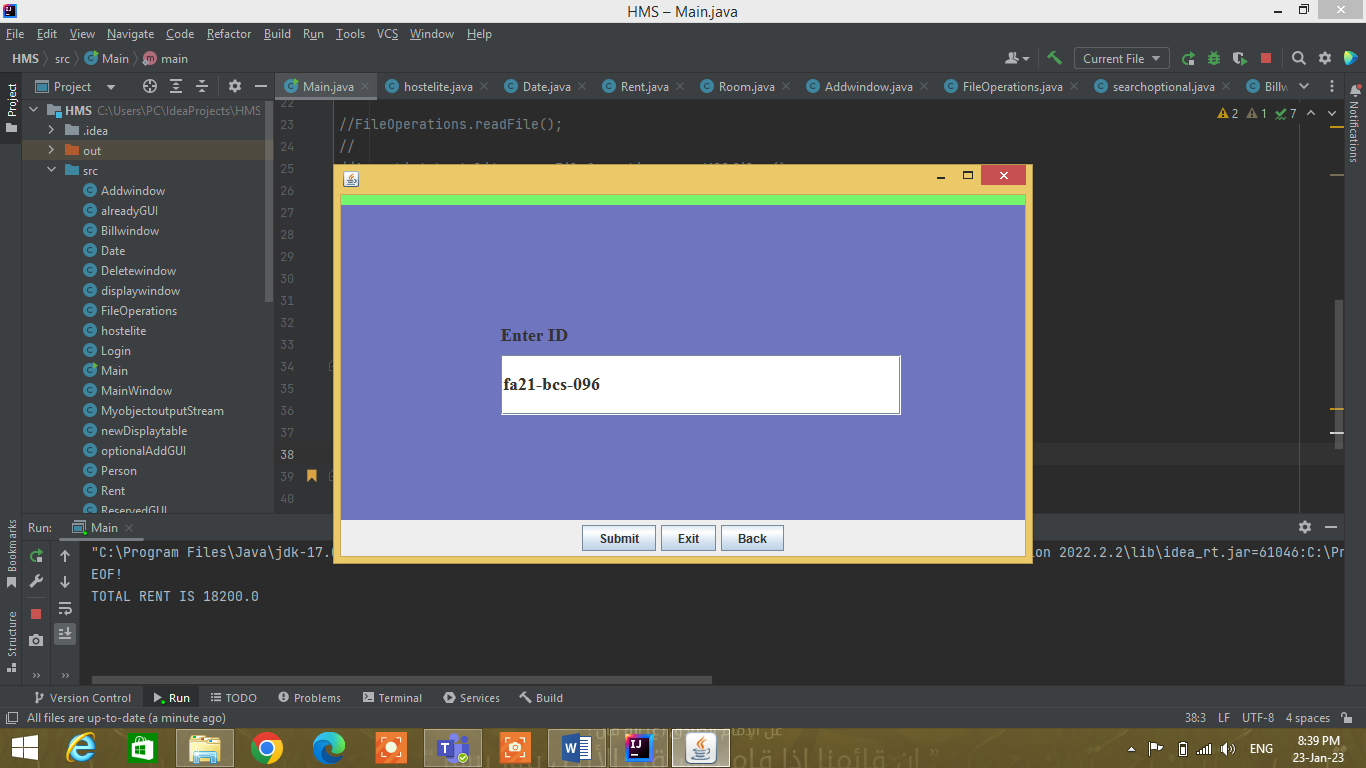
****

****

****

****

****

****