



DEPARTMENT OF COMPUTER SCIENCE

&

ENGINEERING.

COURSE NAME: Programming in Java.

COURSE CODE: CSE310.

Project Topic: Cleaning Management System.

Submitted To: Puneet Kumar.

Submitted By:

Name	Roll no	Reg. No	Section	Group
V. Koushik	A01	12112291	K21PG	1
B. Vamsi	A02	12113081	K21PG	1
D. Dileep	A03	12111650	K21PG	1

TABLE OF CONTENTS:

Acknowledgement	03
Introduction	04
Project Description	05
Classes	06
Project codes.....	09
Result Screenshots.....	17
Work done by Individual member.....	20
Conclusion	21

ACKNOWLEDGEMENT:

First and foremost, we would like to express our gratitude to our Mentor, Prof. Puneet Kumar, who was a continual source of inspiration. He pushed us to think imaginatively and urged us to do this homework without hesitation. His vast knowledge, extensive experience, and professional competence in Java Programming enabled us to successfully accomplish this project. This endeavor would not have been possible without his help and supervision. We could not have asked for a finer mentor in our studies. This initiative would not have been a success without the contributions of each and every individual in the group. We were always there to cheer each other on, and that is what kept us together until the end.

I'd like to thank The Lovely Professional University for providing me with the opportunity to work on the project for **“Cleaning Management System”**. Last but not least, I would like to express my gratitude to our friends for their invaluable assistance, and I am deeply grateful to everyone who has contributed to the successful completion of this project.

INTRODUCTION:

The cleaning sector is an important aspect of our society, with companies offering crucial services to both residential and commercial clientele. A dependable and effective system is required to handle the cleaning company's activities successfully. A system like that would make it possible for the business to efficiently manage its staff, job responsibilities, and timetables, ensuring the efficiency of its operations. This report's objective is to give an outline of the system that a cleaning firm could employ. The system will be created to enable the business to efficiently manage its personnel and task assignments.

In the cleaning industry the allocation of employees and their current location and the work they do is registered in day-to-day work schedule of the company employees in the system. The system allocates the job or the location of the client they need to work for and the shift timings are also mentioned to the employee.

Running a cleaning business is demanding and running a cleaning business is demanding and requires excellent management skills especially when you have a team of cleaners working for your business. You need to choose a cleaning business software that will help you schedule appointments efficiently. Commercial home and office cleaning services providers are always on the road and need to be able to provide quotes and schedule new appointments from anywhere at anytime. So the system they use helps them in this activity.

PROJECT DESCRIPTION:

The objective of this project is to develop a Java application for a cleaning business that hires a number of cleaners who are given jobs at various places. A number of reports, including those that list the employees and their information and allocate jobs to certain cleaners, should be available through the system. A list of the tasks that are currently being allocated to each cleaner should be visible.

To achieve this goal, we have created the following classes:

- Cleaner: This class represents a cleaner and contains their personal details such as name, address, and contact information.
- Job: This class represents a cleaning job and contains information such as the location, type of cleaning required, and the date and time of the job.
- JobAssignment: This class represents the assignment of a job to a cleaner and contains information such as the cleaner assigned to the job and the date and time the job was assigned.
- CleanerList: This class maintains a list of all the cleaners employed by the company.
- JobList: This class maintains a list of all the cleaning jobs that need to be completed.
- JobAssignmentList: This class maintains a list of all the job assignments made to cleaners.
- ReportGenerator: This class generates reports based on the data in the CleanerList, JobList, and JobAssignmentList.

The cleaning service can manage its cleaners, jobs, and job assignments by using these classes. The system can be used to update job assignments, allocate tasks to cleaners, and generate reports based on data stored there. This will enable the business to more effectively and efficiently manage its cleaning operations.

CLASSES:

Cleaner class:

This class represents a cleaner and contains their personal details such as name, address, and contact information. The Cleaner class has the following attributes:

- name: a string that represents the name of the cleaner.
- address: a string that represents the address of the cleaner.
- contactInfo: a string that represents the contact information of the cleaner.

The Cleaner class also has a constructor that initializes these attributes, as well as getter and setter methods.

Job class:

This class represents a cleaning job and contains information such as the location, type of cleaning required, and the date and time of the job. The Job class has the following attributes:

location: a string that represents the location where the cleaning job needs to be done.

- cleaningType: a string that represents the type of cleaning required.

- dateAndTime: a string that represents the date and time the job needs to be done.

The Job class also has a constructor that initializes these attributes, as well as getter and setter methods.

JobAssignment class:

This class represents the assignment of a job to a cleaner and contains information such as the cleaner assigned to the job and the date and time the job was assigned.

The JobAssignment class has the following attributes:

- cleaner: a Cleaner object that represents the cleaner assigned to the job.
- job: a Job object that represents the job that was assigned.
- dateAndTime: a string that represents the date and time the job was assigned.

The JobAssignment class also has a constructor that initializes these attributes, as well as getter and setter methods.

CleanerList class:

This class maintains a list of all the cleaners employed by the company. The CleanerList class has the following attributes:

- cleaners: an ArrayList of Cleaner objects that represents the list of cleaners.

The CleanerList class also has methods to add and remove cleaners from the list, as well as to get a list of all the cleaners.

JobList class:

This class maintains a list of all the cleaning jobs that need to be completed. The JobList class has the following attributes:

- jobs: an ArrayList of Job objects that represents the list of jobs.

The JobList class also has methods to add and remove jobs from the list, as well as to get a list of all the jobs.

JobAssignmentList class:

This class maintains a list of all the job assignments made to cleaners. The JobAssignmentList class has the following attributes:

- jobAssignments: an ArrayList of JobAssignment objects that represents the list of job assignments.
- The JobAssignmentList class also has methods to add and remove job assignments from the list, as well as to get a list of all the job assignments.

ReportGenerator class:

This class generates reports based on the data in the CleanerList, JobList, and JobAssignmentList. The ReportGenerator class has methods to generate reports such as:

- List of all cleaners and their details.
- List of all jobs and their details.
- List of all job assignments and their details.
- List of jobs currently assigned to each cleaner.

PROJECT CODE:

File-Cleaner.java

```
public class Cleaner {
    private String name;
    private String address;
    private String phone;
    private String email;

    public Cleaner(String name, String address, String phone, String email) {
        this.name = name;
        this.address = address;
        this.phone = phone;
        this.email = email;
    }

    public String getName() {
        return name;
    }

    public String getAddress() {
        return address;
    }

    public String getPhone() {
        return phone;
    }

    public String getEmail() {
        return email;
    }

    @Override
    public String toString() {
        return "Cleaner{" +
            "name='" + name + '\'' +
            ", address='" + address + '\'' +
            ", phone='" + phone + '\'' +
            ", email='" + email + '\'' +
            '}';
    }
}
```

File-CleanerList.java

```
import java.util.ArrayList;
import java.util.List;

public class CleanerList {
    private List<Cleaner> cleaners;

    public CleanerList() {
        this.cleaners = new ArrayList<>();
    }

    public void addCleaner(Cleaner cleaner) {
        cleaners.add(cleaner);
    }

    public void removeCleaner(Cleaner cleaner) {
        cleaners.remove(cleaner);
    }

    public List<Cleaner> getCleaners() {
        return cleaners;
    }

    public Cleaner getCleanerByName(String name) {
        for (Cleaner cleaner : cleaners) {
            if (cleaner.getName().equals(name)) {
                return cleaner;
            }
        }
        return null;
    }

    @Override
    public String toString() {
        return "CleanerList{" +
            "cleaners=" + cleaners +
            '}';
    }
}
```

File-Job.java

```
import java.time.LocalDateTime;

public class Job {
    private String location;
    private String type;
    private LocalDateTime date;

    public Job(String location, String type, LocalDateTime date) {
        this.location = location;
        this.type = type;
        this.date = date;
    }

    public String getLocation() {
        return location;
    }

    public String getType() {
        return type;
    }

    public LocalDateTime getDate() {
        return date;
    }

    @Override
    public String toString() {
        return "Job{" +
            "location='" + location + '\'' +
            ", type='" + type + '\'' +
            ", date=" + date +
            '}';
    }
}
```

File-JobAssignment.java

```
import java.time.LocalDateTime;

public class JobAssignment {
    private Cleaner cleaner;
    private Job job;
    private LocalDateTime dateAssigned;

    public JobAssignment(Cleaner cleaner, Job job, LocalDateTime dateAssigned)
    {
        this.cleaner = cleaner;
        this.job = job;
        this.dateAssigned = dateAssigned;
    }

    public Cleaner getCleaner() {
        return cleaner;
    }

    public Job getJob() {
        return job;
    }

    public LocalDateTime getDateAssigned() {
        return dateAssigned;
    }

    @Override
    public String toString() {
        return "JobAssignment{" +
            "cleaner=" + cleaner +
            ", job=" + job +
            ", dateAssigned=" + dateAssigned +
            '}';
    }
}
```

File-JobAssignmentList.java

```
import java.util.ArrayList;
import java.util.List;

public class JobAssignmentList {
    private List<JobAssignment> jobAssignments;

    public JobAssignmentList() {
        this.jobAssignments = new ArrayList<>();
    }

    public void assignJob(Cleaner cleaner, Job job) {
        JobAssignment assignment = new JobAssignment(cleaner, job,
            java.time.LocalDateTime.now());
        jobAssignments.add(assignment);
    }

    public void unassignJob(Cleaner cleaner, Job job) {
        for (JobAssignment assignment : jobAssignments) {
            if (assignment.getCleaner().equals(cleaner) &&
                assignment.getJob().equals(job)) {
                jobAssignments.remove(assignment);
                break;
            }
        }
    }

    public List<JobAssignment> getJobAssignments() {
        return jobAssignments;
    }

    public List<JobAssignment> getJobAssignmentsForCleaner(Cleaner cleaner) {
        List<JobAssignment> result = new ArrayList<>();
        for (JobAssignment assignment : jobAssignments) {
            if (assignment.getCleaner().equals(cleaner)) {
                result.add(assignment);
            }
        }
        return result;
    }

    @Override
    public String toString() {
        return "JobAssignmentList{" +
            "jobAssignments=" + jobAssignments +
            '}';
    }
}
```

File-JobList.java

```
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

public class JobList {
    private List<Job> jobs;

    public JobList() {
        this.jobs = new ArrayList<>();
    }

    public void addJob(Job job) {
        jobs.add(job);
    }

    public void removeJob(Job job) {
        jobs.remove(job);
    }

    public List<Job> getJobs() {
        return jobs;
    }

    public List<Job> getJobsByLocation(String location) {
        List<Job> result = new ArrayList<>();
        for (Job job : jobs) {
            if (job.getLocation().equals(location)) {
                result.add(job);
            }
        }
        return result;
    }

    public List<Job> getJobsByDate(LocalDateTime date) {
        List<Job> result = new ArrayList<>();
        for (Job job : jobs) {
            if (job.getDate().equals(date)) {
                result.add(job);
            }
        }
        return result;
    }

    @Override
    public String toString() {
        return "JobList{" +
            "jobs=" + jobs +
            '}';
    }
}
```

File-Main.java

```
import java.time.LocalDateTime;

public class Main {
    public static void main(String[] args) {
        // create cleaners
        Cleaner john = new Cleaner("John Smith", "123 Main St", "555-1234",
            "john@email.com");
        Cleaner jane = new Cleaner("Jane Doe", "456 Oak Ave", "555-5678",
            "jane@email.com");
        // create jobs
        Job office = new Job("123 Elm St", "office cleaning",
            LocalDateTime.of(2023, 4, 15, 10, 0));
        Job home = new Job("789 Maple Ave", "home cleaning",
            LocalDateTime.of(2023, 4, 16, 14, 0));

        // create lists
        CleanerList cleanerList = new CleanerList();
        cleanerList.addCleaner(john);
        cleanerList.addCleaner(jane);

        JobList jobList = new JobList();
        jobList.addJob(office);
        jobList.addJob(home);

        JobAssignmentList jobAssignmentList = new JobAssignmentList();
        jobAssignmentList.assignJob(john, office);
        jobAssignmentList.assignJob(jane, home);

        // print lists
        System.out.println("Cleaners:");
        System.out.println(cleanerList);

        System.out.println("Jobs:");
        System.out.println(jobList);

        System.out.println("Job assignments:");
        System.out.println(jobAssignmentList);
    }
}
```

File-ReportGenerator.java

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.List;

public class ReportGenerator {
    public static void main(String[] args) {
        // create lists
        CleanerList cleanerList = new CleanerList();
        JobList jobList = new JobList();
        JobAssignmentList jobAssignmentList = new JobAssignmentList();

        // populate lists with sample data
        Cleaner john = new Cleaner(" John Smith", "123 Main St", "555-1234",
"john@email.com");
        Cleaner jane = new Cleaner("Jane Doe", "456 Oak Ave", "555-5678",
"jane@email.com");
        cleanerList.addCleaner(john);
        cleanerList.addCleaner(jane);

        Job office = new Job("123 Elm St", "office cleaning",
java.time.LocalDateTime.of(2023, 4, 15, 10, 0));
        Job home = new Job("789 Maple Ave", "home cleaning",
java.time.LocalDateTime.of(2023, 4, 16, 14, 0));
        jobList.addJob(office);
        jobList.addJob(home);

        jobAssignmentList.assignJob(john, office);
        jobAssignmentList.assignJob(jane, home);

        // generate reports
        try (FileWriter writer = new FileWriter("report.txt")) {
            writer.write("List of cleaners:\n");
            List<Cleaner> cleaners = cleanerList.getCleaners();
            for (Cleaner cleaner : cleaners) {
                writer.write(cleaner + "\n");
            }

            writer.write("\nList of jobs:\n");
            List<Job> jobs = jobList.getJobs();
            for (Job job : jobs) {
                writer.write(job + "\n");
            }

            writer.write("\nList of job assignments:\n");
            List<JobAssignment> jobAssignments =
jobAssignmentList.getJobAssignments();
```



```

        for (JobAssignment assignment : jobAssignments) {
            writer.write(assignment + "\n");
        }

        writer.write("\nList of jobs assigned to John:\n");
        List<JobAssignment> johnAssignments =
jobAssignmentList.getJobAssignmentsForCleaner(john);
        for (JobAssignment assignment : johnAssignments) {
            writer.write(assignment.getJob() + "\n");
        }
        System.out.println("---> Report has been generated.");
    } catch (IOException e) {
        System.err.println("Error writing to file: " + e.getMessage());
    }
}
}

```

RESULT SCREENSHOTS:

Main.java

The screenshot shows the Visual Studio Code editor with the `Main.java` file open. The code defines a `Main` class with a `main` method that creates cleaners, jobs, and job assignments, and then writes a report to a file. The terminal output shows the execution of the program, displaying the list of cleaners, jobs, and job assignments.

```

Main.java
1  import java.time.LocalDateTime;
2
3  public class Main {
4      Run | Debug
5      public static void main(String[] args) {
6          // create cleaners
7          Cleaner john = new Cleaner(name="John Smith", address="123 Main St", phone="555-1234", email="john@email.com");
8          Cleaner jane = new Cleaner(name="Jane Doe", address="456 Oak Ave", phone="555-5678", email="jane@email.com");
9          // create jobs
10         Job office = new Job(location="123 Elm St", type="office cleaning", LocalDateTime.of(year:2023, month:4, dayOfMonth:15,
11         Job home = new Job(location="789 Maple Ave", type="home cleaning", LocalDateTime.of(year:2023, month:4, dayOfMonth:16,
12
13         // create lists
14         CleanerList cleanerList = new CleanerList();
15         cleanerList.addCleaner(john);
16         cleanerList.addCleaner(jane);
17
18         JobList jobList = new JobList();
19         jobList.addJob(office);
20         jobList.addJob(home);
21     }
22 }

```

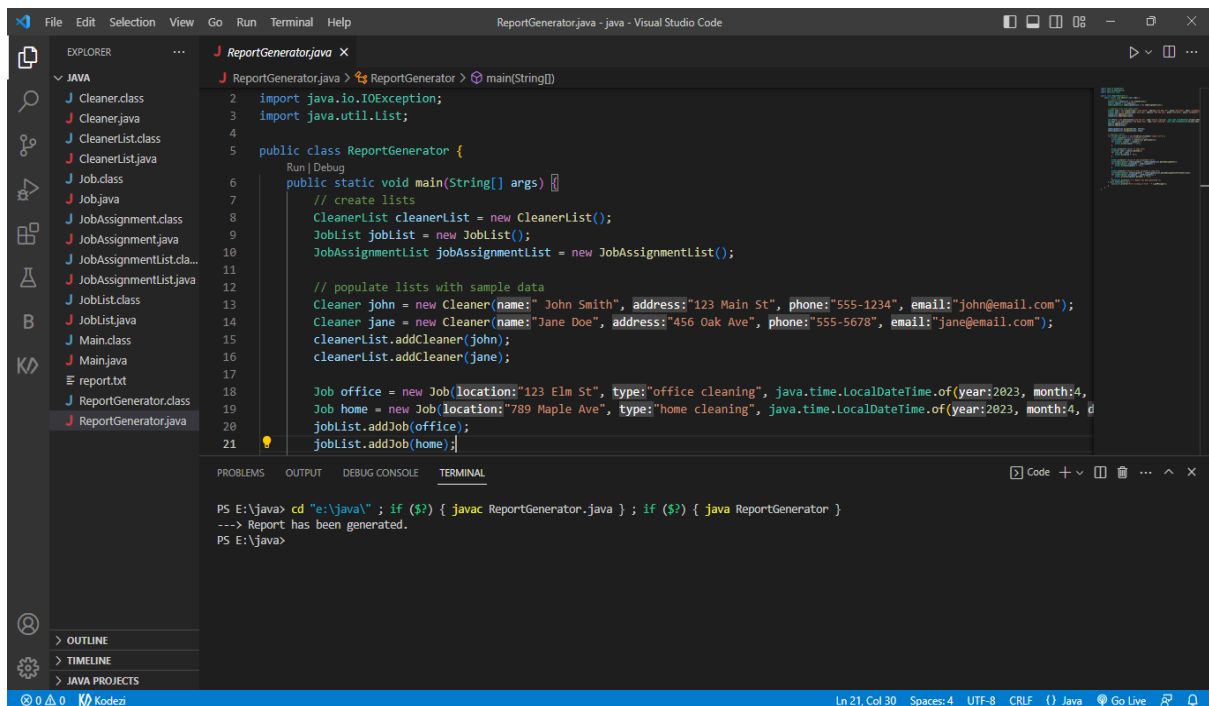
Terminal Output:

```

Cleaners:
CleanerList{cleaners=[Cleaner{name='John Smith', address='123 Main St', phone='555-1234', email='john@email.com'}, Cleaner{name='Jane Doe', address='456 Oak Ave', phone='555-5678', email='jane@email.com'}]}
Jobs:
JobList{jobs=[Job{location='123 Elm St', type='office cleaning', date=2023-04-15T10:00}, Job{location='789 Maple Ave', type='home cleaning', date=2023-04-16T14:00}]}
Job assignments:
JobAssignmentList{jobAssignments=[JobAssignment{cleaner=Cleaner{name='John Smith', address='123 Main St', phone='555-1234', email='john@email.com'}, job=Job{location='123 Elm St', type='office cleaning', date=2023-04-15T10:00}, dateAssigned=2023-04-15T16:31:00.553222}, JobAssignment{cleaner=Cleaner{name='Jane Doe', address='456 Oak Ave', phone='555-5678', email='jane@email.com'}, job=Job{location='789 Maple Ave', type='home cleaning', date=2023-04-16T14:00}, dateAssigned=2023-04-15T16:31:00.553222}]}
PS E:\java>

```

In the output of the main program the details of the employees are given life did name, address, e-mail ID, phone number. And the output also contains their current location and the shift timings of their work and it also contains their job location and the type of work they do.



The screenshot displays the Visual Studio Code interface with the `ReportGenerator.java` file open. The Explorer sidebar on the left shows a project structure with files like `Cleaner.class`, `Cleaner.java`, `CleanerList.class`, `CleanerList.java`, `Job.class`, `Job.java`, `JobAssignment.class`, `JobAssignment.java`, `JobAssignmentList.class`, `JobAssignmentList.java`, `JobList.class`, `JobList.java`, `Main.class`, `Main.java`, `report.txt`, `ReportGenerator.class`, and `ReportGenerator.java`. The main editor shows the source code of `ReportGenerator.java`, which includes imports for `java.io.IOException` and `java.util.List`, and a `main` method that creates lists, populates them with sample data (John Smith and Jane Doe), and adds jobs (office and home cleaning). The bottom panel shows the TERMINAL output, which displays the command to run the program and the resulting message: "Report has been generated."

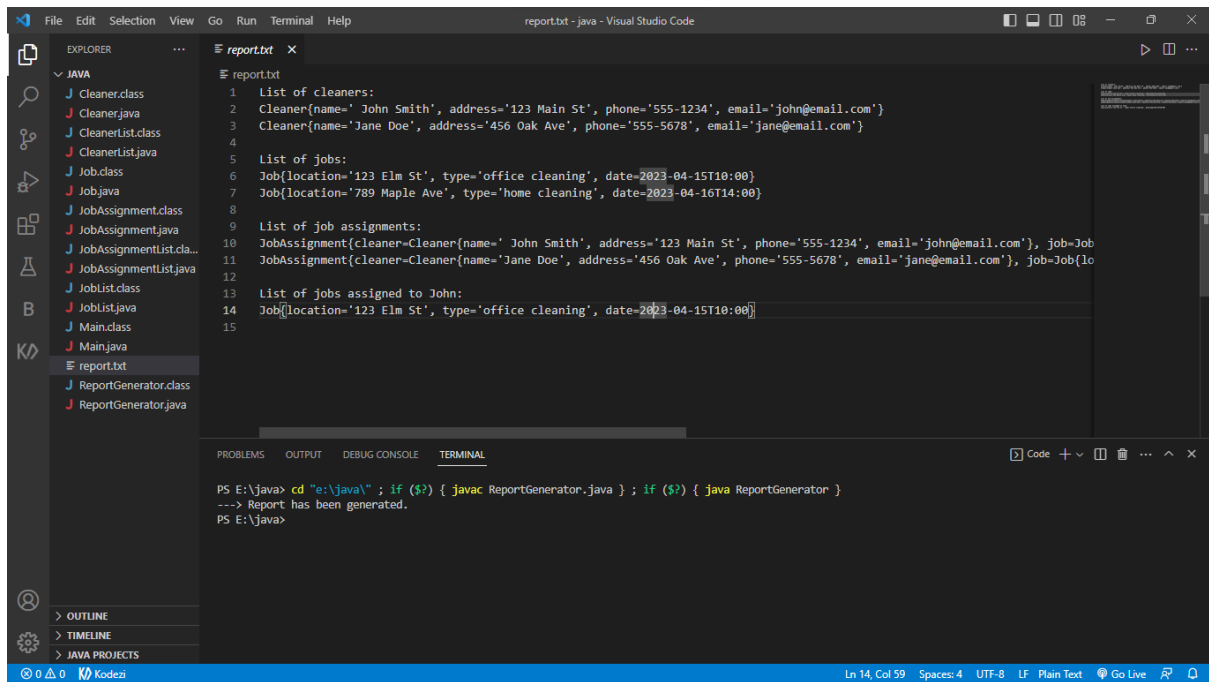
```
File Edit Selection View Go Run Terminal Help
ReportGenerator.java - Java - Visual Studio Code

EXPLORER
  JAVA
    Cleaner.class
    Cleaner.java
    CleanerList.class
    CleanerList.java
    Job.class
    Job.java
    JobAssignment.class
    JobAssignment.java
    JobAssignmentList.class
    JobAssignmentList.java
    JobList.class
    JobList.java
    Main.class
    Main.java
    report.txt
    ReportGenerator.class
    ReportGenerator.java

J ReportGenerator.java > ReportGenerator > main(String[])
2 import java.io.IOException;
3 import java.util.List;
4
5 public class ReportGenerator {
6     Run | Debug
7     public static void main(String[] args) {
8         // create lists
9         CleanerList cleanerList = new CleanerList();
10        JobList jobList = new JobList();
11        JobAssignmentList jobAssignmentList = new JobAssignmentList();
12
13        // populate lists with sample data
14        Cleaner john = new Cleaner(name:" John Smith", address:"123 Main St", phone:"555-1234", email:"john@email.com");
15        Cleaner jane = new Cleaner(name:"Jane Doe", address:"456 Oak Ave", phone:"555-5678", email:"jane@email.com");
16        cleanerList.addCleaner(john);
17        cleanerList.addCleaner(jane);
18
19        Job office = new Job(location:"123 Elm St", type:"office cleaning", java.time.LocalDateTime.of(year:2023, month:4, d
20        Job home = new Job(location:"789 Maple Ave", type:"home cleaning", java.time.LocalDateTime.of(year:2023, month:4, d
21        jobList.addJob(office);
22        jobList.addJob(home);
23    }
24}

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS E:\java> cd "e:\java\" ; if ($?) { javac ReportGenerator.java } ; if ($?) { java ReportGenerator }
--> Report has been generated.
PS E:\java>
```

As shown in the above picture a report has been generated in the form of a notepad and it has been saved as “report.txt” . Whenever new information is added to the code or an employee added the text in the report is being overwritten in the same report file .When we run the report Generator it gives the output as “-→Report has been generated.” when the report is generated .



The screenshot shows the Visual Studio Code interface with a Java project. The Explorer panel on the left lists files: Cleaner.class, Cleaner.java, CleanerList.class, CleanerList.java, Job.class, JobAssignment.class, JobAssignmentList.class, JobAssignmentList.java, JobList.class, JobList.java, Main.class, Main.java, report.txt, ReportGenerator.class, and ReportGenerator.java. The main editor displays the content of report.txt, which contains the following text:

```
1 List of cleaners:
2 Cleaner{name=' John Smith', address='123 Main St', phone='555-1234', email='john@email.com'}
3 Cleaner{name='Jane Doe', address='456 Oak Ave', phone='555-5678', email='jane@email.com'}
4
5 List of jobs:
6 Job{location='123 Elm St', type='office cleaning', date=2023-04-15T10:00}
7 Job{location='789 Maple Ave', type='home cleaning', date=2023-04-16T14:00}
8
9 List of job assignments:
10 JobAssignment{cleaner=Cleaner{name=' John Smith', address='123 Main St', phone='555-1234', email='john@email.com'}, job=Job{lo
11 JobAssignment{cleaner=Cleaner{name='Jane Doe', address='456 Oak Ave', phone='555-5678', email='jane@email.com'}, job=Job{lo
12
13 List of jobs assigned to John:
14 Job{location='123 Elm St', type='office cleaning', date=2023-04-15T10:00}
15
```

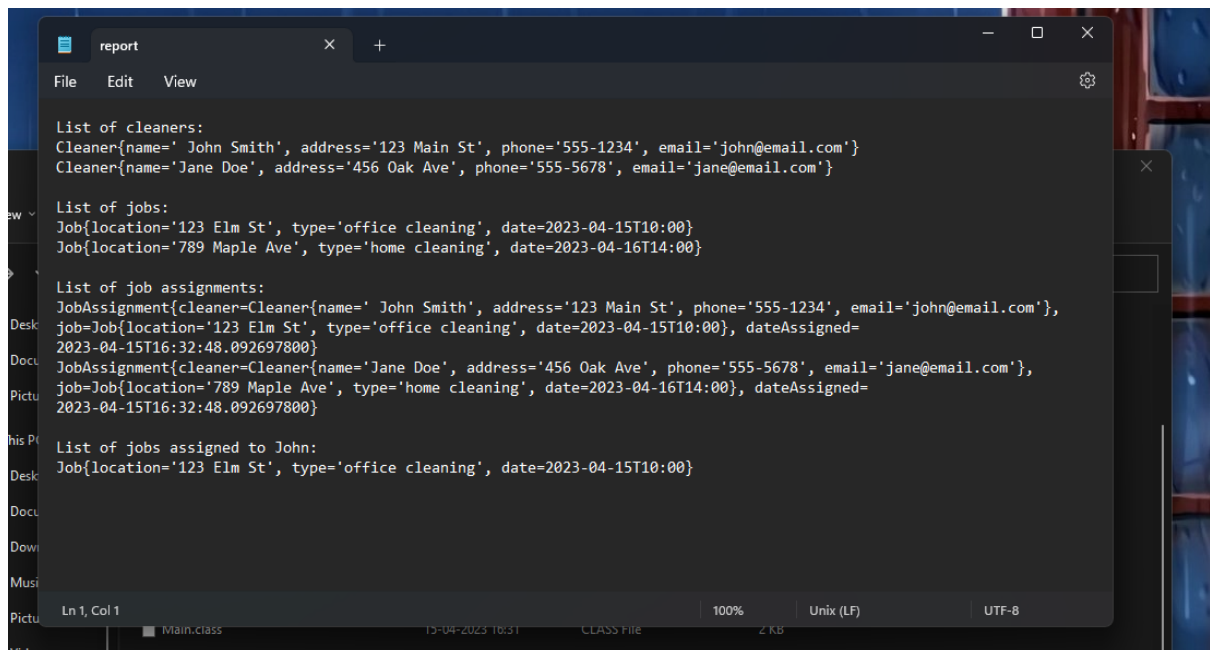
The terminal at the bottom shows the command `cd "e:\java" ; if ($?) { javac ReportGenerator.java } ; if ($?) { java ReportGenerator } --> Report has been generated.`

In the above picture it is shown that the report consists of the following categories-

- List of cleaners
- List of jobs
- List of job assignments
- List of jobs assigned to John

The report consists of the details of the cleaners the jobs and the assignments for each individual and the addresses they have been allocated along with their phone numbers it's been detailed into the report.

File- report.txt



```
File Edit View
List of cleaners:
Cleaner{name=' John Smith', address='123 Main St', phone='555-1234', email='john@email.com'}
Cleaner{name='Jane Doe', address='456 Oak Ave', phone='555-5678', email='jane@email.com'}

List of jobs:
Job{location='123 Elm St', type='office cleaning', date=2023-04-15T10:00}
Job{location='789 Maple Ave', type='home cleaning', date=2023-04-16T14:00}

List of job assignments:
JobAssignment{cleaner=Cleaner{name=' John Smith', address='123 Main St', phone='555-1234', email='john@email.com'},
job=Job{location='123 Elm St', type='office cleaning', date=2023-04-15T10:00}, dateAssigned=
2023-04-15T16:32:48.092697800}
JobAssignment{cleaner=Cleaner{name='Jane Doe', address='456 Oak Ave', phone='555-5678', email='jane@email.com'},
job=Job{location='789 Maple Ave', type='home cleaning', date=2023-04-16T14:00}, dateAssigned=
2023-04-15T16:32:48.092697800}

List of jobs assigned to John:
Job{location='123 Elm St', type='office cleaning', date=2023-04-15T10:00}

Ln 1, Col 1
main.class 12-04-2023 10:51 CLASS FILE 2 KB
100% Unix (LF) UTF-8
```

The above provided screenshots are the output of the codes or the system that we implemented. The screenshots show that the program ran without any errors and all the details of the cleaners and the job they are doing is being saved. And a report can be generated whenever we want.

WORD DONE BY INDIVIDUAL MEMBER:

<i>NAME</i>	<i>ROLE</i>	<i>RESPONSIBILITY</i>
V. Koushik	Developer	Coding & Report
D. Dileep Kumar	Developer	Coding & Report
B. Vamsi	Developer	Report.

CONCLUSION:

The cleaning company will gain a lot from the suggested approach, including improved communication, easier task delegation, and immediate access to crucial data. The organisation will be able to manage its personnel and job responsibilities more effectively by putting this system into place, which will lead to improved levels of client satisfaction and increased profitability. In order to provide a complete response to the issues faced by cleaning companies, the task assignment tracking module, reporting capabilities, and employee database administration module will collaborate. With this approach, the business will be able to expand and take on more projects while maintaining high standards for quality and client satisfaction.