CareerHub Coding Challenge -- JAVA

Name: **Atharva Padawale**

Github: https://github.com/AtharvaPadawale/hexaware-Java-Batch-5

1.Create and implement the mentioned class and the structure in your application.

JobListing Class:

Attributes:

- JobID (int): A unique identifier for each job listing.
- CompanyID (int): A reference to the company offering the job.
- JobTitle (string): The title of the job.
- JobDescription (string): A detailed description of the job.
- JobLocation (string): The location of the job.
- Salary (decimal): The salary offered for the job.
- JobType (string): The type of job (e.g., Full-time, Part-time, Contract).
- PostedDate (DateTime): The date when the job was posted.

```
public class JobListing {
    private int jobID;
    private int companyID;
    private String jobTitle;
    private String jobDescription;
    private String jobLocation;
    private double salary;
    private String jobType;
    private LocalDateTime postedDate;

    private List<JobApplication> applicants = new ArrayList<>();

    public JobListing() {}

    public JobListing(int jobID, int companyID, String jobTitle, String jobDescription,
                      String jobLocation, double salary, String jobType, LocalDateTime postedDate) {
        this.jobID = jobID;
        this.companyID = companyID;
        this.jobTitle = jobTitle;
        this.jobDescription = jobDescription;
        this.jobLocation = jobLocation;
        this.salary = salary;
        this.jobType = jobType;
        this.postedDate = postedDate;
    }
```

Methods:

• Apply(applicantID: int, coverLetter: string): Allows applicants to apply for the job by providing their ID and a cover letter.

```
public void apply(int applicantID, String coverLetter) {
    applicants.add(new JobApplication(0, this.jobID, applicantID, LocalDateTime.now(), coverLetter));
}
```

• GetApplicants(): List<Applicant>: Retrieves a list of applicants who have applied for the job.

```
public List<JobApplication> getApplications() {
    return applications;
}
```

Company Class:

Attributes:

- CompanyID (int): A unique identifier for each company.
- CompanyName (string): The name of the hiring company.
- Location (string): The location of the company.

```java
public class Company {
    private int companyID;
    private String companyName;
    private String location;
    private List<JobListing> jobs = new ArrayList<>();

    public Company() {}

    public Company(int companyID, String companyName, String location) {
        this.companyID = companyID;
        this.companyName = companyName;
        this.location = location;
    }
```

Methods:

- PostJob(jobTitle: string, jobDescription: string, jobLocation: string, salary: decimal, jobType: string): Allows a company to post a new job listing.
- GetJobs(): List<JobListing>: Retrieves a list of job listings posted by the company.

```java
public void postJob(String jobTitle, String jobDescription, String jobLocation, double salary, String jobType) {
    JobListing job = new JobListing(0, this.companyID, jobTitle, jobDescription, jobLocation, salary, jobType, java.time.LocalDateTime.now());
    jobs.add(job);
}

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

Applicant Class:

Attributes:

- ApplicantID (int): A unique identifier for each applicant.
- FirstName (string): The first name of the applicant.
- LastName (string): The last name of the applicant.
- Email (string): The email address of the applicant.
- Phone (string): The phone number of the applicant.
- Resume (string): The applicant's resume or a reference to the resume file.

```java
public class Applicant {
    private int applicantID;
    private String firstName;
    private String lastName;
    private String email;
    private String phone;
    private String resume;
    private List<JobApplication> applications = new ArrayList<>();

    public Applicant() {}

    public Applicant(int applicantID, String firstName, String lastName, String email, String phone, String resume) {
        this.applicantID = applicantID;
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.phone = phone;
        this.resume = resume;
    }
```

Methods:

• CreateProfile(email: string, firstName: string, lastName: string, phone: string): Allows applicants to create a profile with their contact information.

```
public void createProfile(String email, String firstName, String lastName, String phone) {
    this.email = email;
    this.firstName = firstName;
    this.lastName = lastName;
    this.phone = phone;
}
```

• ApplyForJob(jobID: int, coverLetter: string): Enables applicants to apply for a specific job listing.

```
public void applyForJob(int jobID, String coverLetter) {
    JobApplication app = new JobApplication(0, jobID, this.applicantID, java.time.LocalDateTime.now(), coverLetter);
    applications.add(app);
}
```

JobApplication Class:

Attributes:

- ApplicationID (int): A unique identifier for each job application.
- JobID (int): A reference to the job listing.
- ApplicantID (int): A reference to the applicant.
- ApplicationDate (DateTime): The date and time when the application was submitted.
- CoverLetter (string): The cover letter submitted with the application.

```
public class JobApplication {
    private int applicationID;
    private int jobID;
    private int applicantID;
    private LocalDateTime applicationDate;
    private String coverLetter;

    public JobApplication() {}

    public JobApplication(int applicationID, int jobID, int applicantID, LocalDateTime applicationDate, String coverLetter) {
        this.applicationID = applicationID;
        this.jobID = jobID;
        this.applicantID = applicantID;
        this.applicationDate = applicationDate;
        this.coverLetter = coverLetter;
    }
```

2.DatabaseManager Class:

• InitializeDatabase(): Initializes the database schema and tables.

• InsertJobListing(job: JobListing): Inserts a new job listing into the "Jobs" table.

```
===== CareerHub Job Board =====
1. Insert Company
2. Insert Applicant
3. Insert Job Listing
4. Insert Job Application
5. View All Companies
6. View All Applicants
7. View All Job Listings
8. View Applications for a Job
9. Exit
```

```
Choose an option: 3
Job ID: 3011
Company ID: 4
Job Title: azure dev
Description: handle azure
Location: pune
Salary: 770000
Job Type (Full-time/Part-time): Full-time
Job Listing inserted.
```

• **InsertCompany(company: Company): Inserts a new company into the "Companies" table.**

```
===== CareerHub Job Board =====
1. Insert Company
2. Insert Applicant
3. Insert Job Listing
4. Insert Job Application
5. View All Companies
6. View All Applicants
7. View All Job Listings
8. View Applications for a Job
9. Exit
```

```
Choose an option: 1
Company ID: 8
Company Name: apmosys
Location: navi mumbai
Company inserted.
```

• **InsertApplicant: Inserts a new applicant into the "Applicants" table.**

```
===== CareerHub Job Board =====
1. Insert Company
2. Insert Applicant
3. Insert Job Listing
4. Insert Job Application
5. View All Companies
6. View All Applicants
7. View All Job Listings
8. View Applications for a Job
9. Exit
```

```
Choose an option: 2
Applicant ID: 112
First Name: tom
Last Name: cruz
Email: tom@gmail.com
Phone: 1231231234
Resume File Path (e.g.
Applicant inserted.
```

• **InsertJobApplication: Inserts a new job application into the "Applications" table.**

```
===== CareerHub Job Board =====
1. Insert Company
2. Insert Applicant
3. Insert Job Listing
4. Insert Job Application
5. View All Companies
6. View All Applicants
7. View All Job Listings
8. View Applications for a Job
9. Exit
```

```
Choose an option: 4
Application ID: 115
Job ID: 203
Applicant ID: 115
Cover Letter: fresher hire me
Job Application submitted.
```

• **GetJobListings(): List<JobListing>: Retrieves a list of all job listings.**

```
===== CareerHub Job Board =====
1. Insert Company
2. Insert Applicant
3. Insert Job Listing
4. Insert Job Application
5. View All Companies
6. View All Applicants
7. View All Job Listings
8. View Applications for a Job
9. Exit
Choose an option: 5
```

```
--- All Job Listings ---
201: Java Developer at New York
202: Data Analyst at San Francisco
203: DevOps Engineer at Chicago
204: QA Tester at Boston
205: UI/UX Designer at Austin
206: Project Manager at Seattle
207: Cloud Engineer at Denver
```

• **GetCompanies(): List<Company>: Retrieves a list of all companies.**

```
===== CareerHub Job Board =====
1. Insert Company
2. Insert Applicant
3. Insert Job Listing
4. Insert Job Application
5. View All Companies
6. View All Applicants
7. View All Job Listings
8. View Applications for a Job
9. Exit
Choose an option: 5
```

```
--- All Companies ---
1: TechCorp - New York
2: InnoWave - San Francisco
3: DataNimbus - Chicago
4: BrightMind - Boston
5: SkyLine Solutions - Austin
6: CyberVista - Seattle
7: NovaCore - Denver
```

• **GetApplicants(): List<Applicant>: Retrieves a list of all applicants.**

```
===== CareerHub Job Board =====
1. Insert Company
2. Insert Applicant
3. Insert Job Listing
4. Insert Job Application
5. View All Companies
6. View All Applicants
7. View All Job Listings
8. View Applications for a Job
9. Exit
Choose an option: 6
```

```
--- All Applicants ---
101: Alice Walker, alice.walker@example.com
102: Bob Smith, bob.smith@example.com
103: Charlie Brown, charlie.brown@example.com
104: Diana Prince, diana.prince@example.com
105: Ethan Clark, ethan.clark@example.com
106: Fiona Evans, fiona.evans@example.com
107: George Harrison, george.harrison@example.com
110: ayushi sharma, ayuhs.com
111: simba tiger, simba@gmail.com
```

• **GetApplicationsForJob(jobID: int): List<JobApplication>: Retrieves a list of job applications for a specific job listing.**

```
===== CareerHub Job Board =====
1. Insert Company
2. Insert Applicant
3. Insert Job Listing
4. Insert Job Application
5. View All Companies
6. View All Applicants
7. View All Job Listings
8. View Applications for a Job
9. Exit
Choose an option: 8
Enter Job ID: 203
```

```
--- Applications for Job ID 203 ---
ApplicationID: 303, ApplicantID: 103
```

3.Exceptions handling

Create and implement the following exceptions in your application.

• **Invalid Email Format Handling**:

o In the Job Board application, during the applicant registration process, users are required to enter their email addresses. Write a program that prompts the user to input an email address and implement exception handling to ensure that the email address follows a valid format (e.g., contains "@" and a valid domain). If the input is not valid, catch the exception and display an error message. If it is valid, proceed with registration.

```
Choose an option: 2
Applicant ID: 110
First Name: ayushi
Last Name: sharma
Email: ayuhs.com
Phone: 1223456789
Resume Text: ayush.resume
Applicant inserted.
```

```
Invalid email format: ayuhs.com
```

• **Salary Calculation Handling:**

o Create a program that calculates the average salary offered by companies for job listings. Implement exception handling to ensure that the salary values are non-negative when computing the average. If any salary is negative or invalid, catch the exception and display an error message, indicating the problematic job listings.

```
Job Title: python developer
Description: python application handling
Location: mumbai
Salary: -66000
Job Type (Full-time/Part-time): Full-time
```

```
Salary cannot be negative: -66000.0
```

- **File Upload Exception Handling:**

o In the Job Board application, applicants can upload their resumes as files. Write a program that handles file uploads and implements exception handling to catch and handle potential errors, such as file not found, file size exceeded, or file format not supported. Provide appropriate error messages in each case.

```
Choose an option: 2
Applicant ID: 111
First Name: simba
Last Name: tiger
Email: simba@gmail.com
Phone: 1231231231
Resume File Path (e.g., C:/resumes/resume.pdf): C:/resumes/resume.pdf
```

```
Resume file not found: C:/resumes/resume.pdf
```

- **Application Deadline Handling:**

o Develop a program that checks whether a job application is submitted before the application deadline. Implement exception handling to catch situations where an applicant tries to submit an application after the deadline has passed. Display a message indicating that the application is no longer accepted.

- **Database Connection Handling:**

o In the Job Board application, database connectivity is crucial. Create a program that establishes a connection to the database to retrieve job listings. Implement exception handling to catch database-related exceptions, such as connection errors or SQL query errors. Display appropriate error messages and ensure graceful handling of these exceptions.

```
===== CareerHub Job Board =====
1. Insert Company
2. Insert Applicant
3. Insert Job Listing
4. Insert Job Application
5. View All Companies
6. View All Applicants
7. View All Job Listings
8. View Applications for a Job
9. Exit
Choose an option:
```

```
Database schema initialized successfully.
```

4.Database Connectivity

Create and implement the following tasks in your application.

• **Job Listing Retrieval:** Write a program that connects to the database and retrieves all job listings from the "Jobs" table. Implement database connectivity using Entity Framework and display the job titles, company names, and salaries.

```
===== CareerHub Job Board =====
1. Insert Company
2. Insert Applicant
3. Insert Job Listing
4. Insert Job Application
5. View All Companies
6. View All Applicants
7. View All Job Listings
8. View Applications for a Job
9. Exit
10. View Job Listings with Company & Salary
Choose an option: 10
```

```
--- Job Listings with Company and Salary ---
Job: Java Developer, Company: TechCorp, Salary: ₹75000.0
Job: Data Analyst, Company: InnoWave, Salary: ₹68000.0
Job: DevOps Engineer, Company: DataNimbus, Salary: ₹80000.0
Job: QA Tester, Company: BrightMind, Salary: ₹60000.0
Job: UI/UX Designer, Company: SkyLine Solutions, Salary: ₹70000.0
Job: Project Manager, Company: CyberVista, Salary: ₹90000.0
Job: Cloud Engineer, Company: NovaCore, Salary: ₹85000.0
Job: azure dev, Company: BrightMind, Salary: ₹770000.0
```

• **Applicant Profile Creation:** Create a program that allows applicants to create a profile by entering their information. Implement database connectivity to insert the applicant's data into the "Applicants" table. Handle potential database-related exceptions.

```
===== CareerHub Job Board =====
1. Insert Company
2. Insert Applicant
3. Insert Job Listing
4. Insert Job Application
5. View All Companies
6. View All Applicants
7. View All Job Listings
8. View Applications for a Job
9. Exit
```

```
Choose an option: 2
Applicant ID: 112
First Name: tom
Last Name: cruz
Email: tom@gmail.com
Phone: 1231231234
Resume File Path (e.g.
Applicant inserted.
```

• **Job Application Submission:** Develop a program that allows applicants to apply for a specific job listing. Implement database connectivity to insert the job application details into the "Applications" table, including the applicant's ID and the job ID. Ensure that the program handles database connectivity and insertion exceptions.

```
===== CareerHub Job Board =====
1. Insert Company
2. Insert Applicant
3. Insert Job Listing
4. Insert Job Application
5. View All Companies
6. View All Applicants
7. View All Job Listings
8. View Applications for a Job
9. Exit
```

```
Choose an option: 4
Application ID: 115
Job ID: 203
Applicant ID: 115
Cover Letter: fresher hire me
Job Application submitted.
```

• **Company Job Posting:** Write a program that enables companies to post new job listings. Implement database connectivity to insert job listings into the "Jobs" table, including the company's ID. Handle database-related exceptions and ensure the job posting is successful.

```
===== CareerHub Job Board =====
1. Insert Company
2. Insert Applicant
3. Insert Job Listing
4. Insert Job Application
5. View All Companies
6. View All Applicants
7. View All Job Listings
8. View Applications for a Job
9. Exit
```

```
Choose an option: 3
Job ID: 208
Company ID: 4
Job Title: gamer
Description: game testing
Location: gujrat
Salary: 10000
Job Type (Full-time/Part-time): Part-time
Job Listing inserted.
```

• **Salary Range Query:** Create a program that allows users to search for job listings within a specified salary range. Implement database connectivity to retrieve job listings that match the user's criteria, including job titles, company names, and salaries. Ensure the program handles database connectivity and query exceptions.

```
Choose an option: 11
Enter min salary: 50000
Enter max salary: 70000

--- Jobs in Salary Range ---
Job: Data Analyst, Company: InnoWave, Salary: ₹68000.0
Job: QA Tester, Company: BrightMind, Salary: ₹60000.0
```

```
===== CareerHub Job Board =====
1. Insert Company
2. Insert Applicant
3. Insert Job Listing
4. Insert Job Application
5. View All Companies
6. View All Applicants
7. View All Job Listings
8. View Applications for a Job
9. Exit
10. View Job Listings with Company & Salary
11. Search Jobs by Salary Range
```