Name: Bhagyashri Thorat
Test: mindex-java-code-challenge
git repo: https://github.com/bhagyashrit24/mindex-java-code-challenge
How to run: Build using gradlew bootRun
To test: Run the application using **ChallengeApplication** class and import the Postman collection provided in the git repo.

As a part of this coding challenge there were 2 tasks as follows -
Task 1
Create a new type, ReportingStructure, that has two properties: employee and numberOfReports.

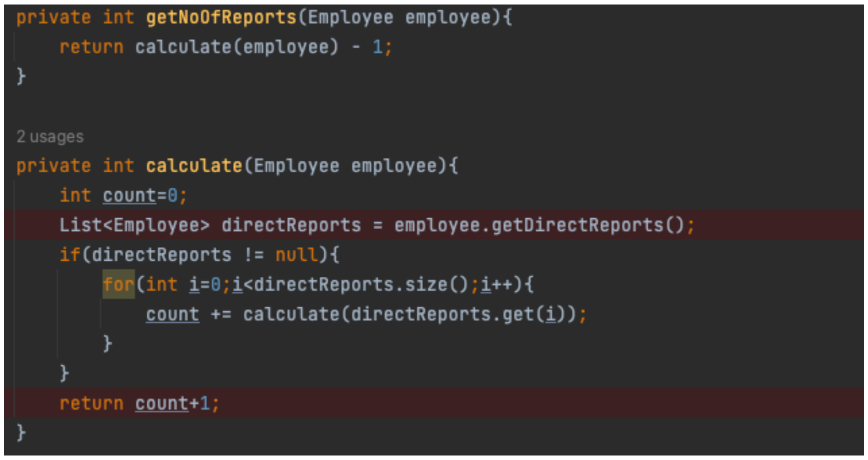
For the field "numberOfReports", this should equal the total number of reports under a given employee. The number of
reports is determined to be the number of directReports for an employee and all of their distinct reports. For example,
given the following employee structure:
```
                    John Lennon
                /               \
         Paul McCartney         Ringo Starr
                               /       \
                          Pete Best     George Harrison
```
The numberOfReports for employee John Lennon (employeeId: 16a596ae-edd3-4847-99fe-c4518e82c86f) would be equal to 4.

This new type should have a new REST endpoint created for it. This new endpoint should accept an employeeId and return the fully filled out ReportingStructure for the specified employeeId. The values should be computed on the fly and will not be persisted.

As a part of this task -
- I created **ReportingStructure** class with 2 attributes - employee(of type Employee) and numberOfReports(of type int) which indicated the total number of reportees under a particular employee. For creating constructors (no arguments and all arguments constructor) , getters and setters I used **lombok** instead of writing them manually.
- Also created an API for creating this reporting structure -
  /reportingStructure/{employeeId}

It took employeeId and created Reporting structure for that particular employee.

- For the above API created an implementation where I assembled the ReportingStructure which included calculating the total number of reportees as follows -



```
private int getNoOfReports(Employee employee){
    return calculate(employee) - 1;
}

2 usages
private int calculate(Employee employee){
    int count=0;
    List<Employee> directReports = employee.getDirectReports();
    if(directReports != null){
        for(int i=0;i<directReports.size();i++){
            count += calculate(directReports.get(i));
        }
    }
    return count+1;
}
```

Here I basically use recursive Depth first search for counting the total number of reportees under one employee and added it to the reporting structure.

- Wrote test case for the same as follows -
    - testCreateReportingStructureZeroDR - No direct reports at all
    - testCreateReportingStructureTwoDR - Two total reports

```
                    John Doe
                   /        \
          Pete Best          Paul Maccarthy
```

    - testCreateReportingStructureFiveDR - Five total reports

```
              John Doe
             /        \
    Pete Best          Paul Maccarthy
   /                         \
Penelope Alvarez              Julia Roberts
                                    \
                                     Dave Smith
```

TO TEST -

To test the reporting structure API first **create employee** collection. Collect the employeeID created and paste in the URL in the **get reporting structure** collection in place of id.

Task 2

Create a new type, Compensation. A Compensation has the following fields: employee, salary, and effectiveDate. Create two new Compensation REST endpoints. One to create and one to read by employeeId. These should persist and query the Compensation from the persistence layer.

As a part of this task -

- I created a new class **Compensation** with attributes employee(of type Employee), salary(of type String) and effective date(of Type Instant) and also mongo CompensationRepository. For creating constructors (no arguments and all arguments constructor) , getters and setters I used **lombok** instead of writing them manually.
- Created a new controller with 2 new APIs -
  1. To persist compensation
     - /employeeCompensation

  2. To read compensation from the repository using employeeId
     - /employeeCompensation/{id}

- Then created implementation for the above created controllers where I inserted the compensation in the db and read the compensation from the DB. While inserting the compensation I first checked if the employee is already present in the database or not. If it is present the compensation is directly added, if not the employee is created i.e. employeeID is generated for that employee and then compensation is persisted in DB.

- Also created test cases for creating and reading compensation from the mongoDB repository. There are 2 test cases. One case in which an employee is already present in the database. And second in which an employee is not present and hence created and the compensation is added against that particular employeeId.

TO TEST -

1. **Employee created -**
   First create the employee using **create employee** collection present in the Postman folder in the git repo. Copy the employeeID and paste it in the body against the employeeId field name in the **create compensation** postman collection. Compensation will be created.

   Then copy and paste the employeeID in the **read compensation** collection in place of id. Compensation for that employeeId will be created.

2. **Employee not present -**
   In this case simply pass compensation request body without employeeId in the **create compensation** collection. After clicking send employeeID will be created.

   Copy and paste the employeeID in the **read compensation** collection in place of id. Compensation for that employeeId will be created.