**UML DIAGRAMS:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Introduced Classes:**

**EmployeeController.java =** I've refactored this class to adhere to the Single Responsibility Principle (SRP). Initially, this class was performing two different tasks: validating employee data and saving employee data. I've created two new classes, SaveEmployee.java to handle saving employee data and ValidateEmployee.java to validate employee information. Now, my EmployeeController.java class is dedicated to processing employee data.

**ValidateEmployee.java=** This class is responsible for validating employee data. It checks various aspects of employee information to meet specific criteria.

**SaveEmployee.java=** SaveEmployee class is responsible for saving employee data in two different formats: **JSON and Text**. And This class uses PersistenceService.class to perform the actual file writing.

**PersistenceService.java=** This class is responsible for saving employee data to a file. It abstracts the process of saving data by using **Formatter.java** to format employee data.

**EmployeeService.java=** In EmployeeService.java, there were methods that were not used by the classes implementing this interface. For instance, it was calculating bonuses, which are not needed for contract employees, and it was calculating renewal dates, which are not needed for permanent employees. I have refactored the EmployeeService.java interface to include only methods that all implementing classes require.

**PermanentEmployee.java=** This class implements the EmployeeService.java interface. It overrides two methods (getSalary and calculateTotalCompensation) from the interface it implements. Additionally, it has methods to calculate bonuses and pensions.

**ContractEmployee.java=** This class implements the EmployeeService.java interface. It overrides two methods (getSalary and calculateTotalCompensation) from the interface it implements. This class ensures that the method to get the renewal date is properly calculated

**TextFormatter.java=** This class implements the Formatter.java interface and is responsible for formatting Employee objects as key-value pairs.

**JUNIT:**

**-PersistenceServiceTest.java:** This class tests the functionality of PersistenceService.java. It checks whether the saveEmployee method in PersistenceService is functioning correctly. In the Setting method, it creates an instance of PersistenceService using a custom formatter. The cleanUp method cleans up by deleting any existing tests. The testSaveEmployee method conducts the actual testing, which involves several steps:

1. Creating an employee.
2. Calling the saveEmployee method of PersistenceService to save the employee’s data.
3. Reading the content of the testing file.

**ContractEmployeeImplTest.java:** This class tests the functionality of ContractEmployeeImpl.java. It performs various tests for the methods mentioned above. The testGetSalary method creates an instance of ContractEmployeeImpl with a salary of 85,000 and uses the assertEquals method to check if the expected and actual values match. The testCalculateTotalCompensation method also creates an instance of ContractEmployeeImpl, but with a salary of 8,500, and uses assertEquals to compare expected and actual values. The testRenewalDate method verifies that the actual renewal date returned by the renewalDate function is within 1 second of the expected date by comparing the time difference.

**-PermanentEmployeeImpTest.Java:** PermanentEmployeeImplTest.java: This class tests the functionalities of PermanentEmployeeImpl.java. It specifically focuses on testing the bonus and total compensation calculation methods.