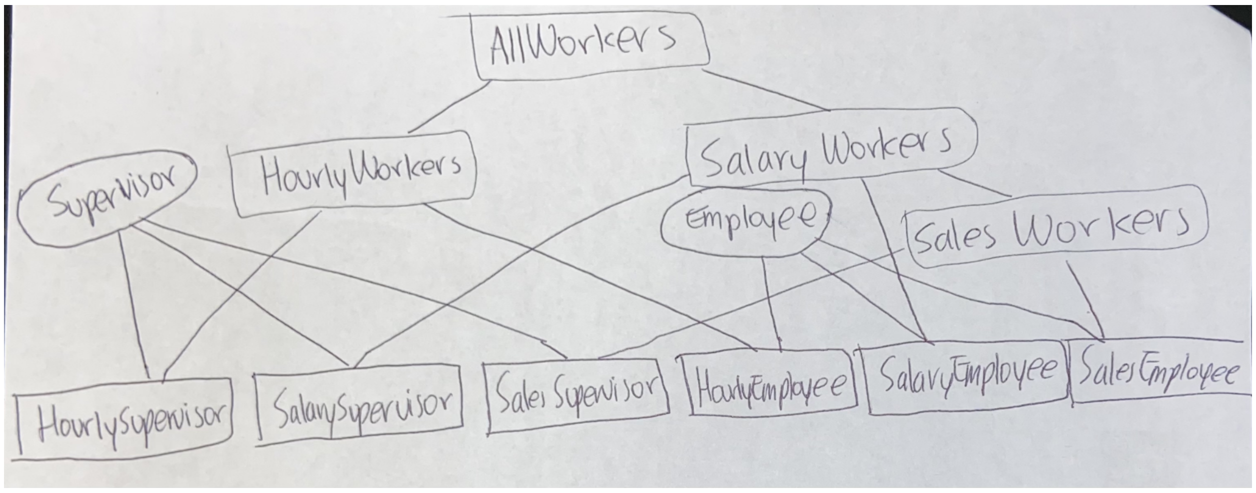
CSDS 132 Project #3 Testing Report

Hierarchy Diagram



AllWorkers Abstract Class

- Method getFirstName() should be tested to see if it correctly return the value stored in the private field firstName of this AllWorkers object in String.
- Method getLastName() should be tested to see if it return the value stored in the private field lastName of this AllWorkers object in String.
- Method setName(String firstName, String lastName) should be tested to see if it change the value of the private fields firstName and lastName of this AllWorkers object to what's in the input parameter.
- Method getNumber() should be tested to see if it return what's stored in the private field number of this object. The return value should be a String.
- Method getBonus() should be tested to see if it return what's stored in the private field bonus of this object. The return value should be a double.
- Method setBonus(double bonus) should be tested to see if it change what's stored in the private field "bonus" to be the value of "bonus" in the method input parameter.

- Method equals(Object worker) be tested to see if it correctly takes in different types of AllWorkers (types that are narrower than AllWorkers) and only takes in objects within the AllWorkers hierarchy. On top of that, it should also be tested if it correctly returns true only if first name, last name, and number of the input worker is the same as this worker (test for 0 matching, 1 matching, 2 matching, and all matching; only return true if all of them match).
- Method setSupervisor(Supervisor supervisor) should be tested to see if it sets this object's supervisor(private field) to be the supervisor of the Supervisor type in the input parameter.
- Method getSupervisor() should be tested to see if it return what's stored in the private field "supervisor" of this object. The return value should be a Supervisor.

HourlyWorkers Abstract Class

- Method getHourlyRate() should be tested to see if it return what's stored in the private field "hourlyRate" of this object. The return value should be a double.
- Method setHourlyRate(double rate) should be tested to see if it change what's stored in the private field "hourlyRate" to be the value of "rate" in the method input parameter.
- Method getHoursWorked() should be tested to see if it return what's stored in the private field "hoursWorked" of this object. The return value should be a double.
- Method setHoursWorked(double rate) should be tested to see if it change what's stored in the private field "hoursWorked" to be the value of "hoursWorked" in the method input parameter.
- Method getAmountEarned() of the HourlyWorkers class should be tested to see if it correctly returns the calculation of (hourly rate) *(hours worked) + bonus of this hourly worker.
- Method adjustPay(double percentage) should be tested to see if it correctly changes the value of hourlyRate field of this object with hourly rate by percentage when the input parameter is negative, positive, and 0.

SalaryWorkers Abstract Class

- Method getSalary() should be tested to see if it return what's stored in the private field "salary" of this object. The return value should be String.
- Method setSalary(double salary) should be tested to see if it change what's stored in the private field "salary" to be the value of "salary" in the method input parameter.
- Method getAmountEarned() of the SalaryWorkers should be tested to see if it correctly returns the calculation of (salary) + bonus of this salary worker.
- Method adjustPay(double percentage) of the SalaryWorkers class should be tested to see if it correctly changes the value of "salary" field of this object with salary by percentage when the input parameter is negative, positive, and 0.

SalesWorkers Abstract Class

- Method getCommission() should be tested to see if it return what's stored in the private field "commission" of this object. The return value should be double.
- Method setCommission(double commission) should be tested to see if it change what's stored in the private field "commission" to be the value of "commission" in the method input parameter.
- Method getNumSales() should be tested to see if it return what's stored in the private field "numSales" of this object. The return value should be int.
- Method setNumSales(int numSales) should be tested to see if it change what's stored in the private field "numSales" to be the value of "numSales" in the method input parameter.
- Method getAmountEarned() of the SalesWorker class should be tested to see if it is correctly overridden to return the calculation of salary + (commission * numSales) + bonus of this salary worker.
- Method adjustPay(double percentage) of the SalesWorker class should be tested to see if it is correctly overridden to change the value of "salary" field of this object with commission by percentage when the input parameter is negative, positive, and 0.

Supervisor Interface

- Method compareToByName(Supervisor s) of the Supervisor interface should be tested to see if it accurately returns 0 (meaning the input and this supervisor are the same people) when (and only when) both getFirstName() of this and s are equal. In case they're not the same person, it should be given instances where first names are the same but not the last names and vise versa, and when they are all different.
- Method compareToByEarnings(Supervisor s) of the Supervisor interface should be tested to see if it accurately returns -1 when this.getAmountEarned() is smaller than s.getAmountEarned(), 1 when this.getAmountEarned() is larger than s.getAmountEarned(), and 0 when this.getAmountEarned() is equal to s.getAmountEarned(),

Employee Interface

- Method compareToByName(Employee e) of the Employee interface should be tested to see if it accurately returns 0 (meaning the input and this employee are the same people) when (and only when) both getFirstName() of this and s are equal. In case they're not the same person, it should be given instances where first names are the same but not the last names and vise versa, and when they are all different.
- Method compareToByEarnings(Employee s) of the Employee interface should be tested to see if it accurately returns -1 when this.getAmountEarned() is smaller than e.getAmountEarned(), 1 when this.getAmountEarned() is larger than e.getAmountEarned(), and 0 when this.getAmountEarned() is equal to e.getAmountEarned(),

HourlySupervisor Class

- Method toString() of HourlySupervisor class should be tested to see if it correctly overrides the Object toString method to return a String "*number*: *last name*, *first name*, Hourly Supervisor", where number is the value of getNumber(), last name the value of getLastName(), first name the value of getFirstName().

SalariedSupervisor Class

- Method toString() of SalarySupervisor class should be tested to see if it correctly overrides the Object toString method to return a String "*number*: *last name*, *first name*, Salaried Supervisor", where number is the value of getNumber(), last name the value of getLastName(), first name the value of getFirstName().

SalesSupervisor Class

- Method toString() of SalesSupervisor class should be tested to see if it correctly overrides the Object toString method to return a String "*number*: *last name*, *first name*, Sales Supervisor", where number is the value of getNumber(), last name the value of getLastName(), first name the value of getFirstName().

HourlyEmployee Class

- Method toString() of HourlyEmployee class should be tested to see if it correctly overrides the Object toString method to return a String "*number*: *last name*, *first name*, Hourly Employee", where number is the value of getNumber(), last name the value of getLastName(), first name the value of getFirstName().

SalariedEmployee Class

- Method toString() of SalesEmployee class should be tested to see if it correctly overrides the Object toString method to return a String "*number*: *last name*, *first name*, Salaried Employee", where number is the value of getNumber(), last name the value of getLastName(), first name the value of getFirstName().

SalesEmployee Class

- Method toString() of SalesEmployee class should be tested to see if it correctly overrides the Object toString method to return a String "*number*: *last name*, *first name*, Sales Employee", where number is the value of getNumber(), last name the value of getLastName(), first name the value of getFirstName().

EmployeeDatabase Class

- Method add(T worker) should be tested to see if workers from all 6 types could be added to the database (HourlySupervisor, SalarySupervisor, SalesSupervisor, HourlyEmployee, SalaryEmployee, SalesEmployee).
- Method remove(String firstName, String lastName, String number) should be tested to see if it correctly runs when the worker is in the beginning, middle, end of the database. It should also be tested to make sure NoSuchEmployeeException is thrown when there is no input parameter worker in the database. To check the if statement, it should be tested if only one of the three conditions is met and when all the conditions are met.
- Method find(String firstName, String lastName, String number) should be tested to see if it correctly finds and returns the worker by the information given in the input parameter. To check the if statement, it should be tested if only one of the three conditions are met and when all the conditions are met. It should also be tested to make sure NoSuchEmployeeException is thrown when there is no input parameter worker in the database.
- Method getPayrollAmount() should be tested to see if it correctly returns the total payroll amount of all the workers added to the database when it is made up of 0, 1, and many workers.
- Method getMaximumEarned() should be tested to see if it correctly returns the worker with the highest earning when there is 0, 1, and many workers in the database. Also, it should be tested to see if it works when the highest earner is in the beginning, middle, and end of the database.

- Method getMinimumEarned() should be tested to see if it correctly returns the worker with the lowest earning when there is 0, 1, and many workers in the database. Also, it should be tested to see if it works when the lowest earner is in the beginning, middle, and end of the database.
- Method getMaxSales() should be tested to see if it correctly returns the worker with the highest sales when there is 0, 1, and many sales workers in the database. Also, it should be tested to see if it works when the sales worker with the highest sales is in the beginning, middle, and end of the database. It should also be tested to see if it throws NoSuchEmployeeException when there are no sales workers in the database.
- Method getMinSales() should be tested to see if it correctly returns the worker with the highest sales when there is 0, 1, and many sales workers in the database. Also, it should be tested to see if it works when the sales worker with the lowest sales is in the beginning, middle, and end of the database. It should also be tested to see if it throws NoSuchEmployeeException when there are no sales workers in the database.
- Method getMaxSales() should be tested to see if it correctly returns the worker with the highest sales when there is 0, 1, and many sales workers in the database. Also, it should be tested to see if it works when the sales worker with the highest sales is in the beginning, middle, and end of the database. It should also be tested to see if it throws NoSuchEmployeeException when there are no sales workers in the database.
- Method getMaxHoursWorked() should be tested to see if it correctly returns the hourly worker with the highest hours worked when there are 0, 1, and many hourly workers in the database. Also, it should be tested to see if it works when the hourly worker with the highest hour is in the beginning, middle, and end of the database. It should also be tested to see if it throws NoSuchEmployeeException when there are no hourly workers in the database.
- Method getMinHoursWorked() should be tested to see if it correctly returns the hourly worker with the lowest hours worked when there are 0, 1, and many hourly workers in the database. Also, it should be tested to see if it works when the hourly worker with the lowest hour is in the beginning, middle, and end of the database. It should also be tested to see if it throws NoSuchEmployeeException when there are no hourly workers in the database.

- Method getSupervisees(Supervisor supervisor) should be tested to see if it correctly returns the array storing all supervisees of the input supervisor when there is 0, 1, and many supervisees in the database.