CSG 100 Data Structures Fall 2004

Problem Set #3: Revision and some recursion

Due Date: 18th of November

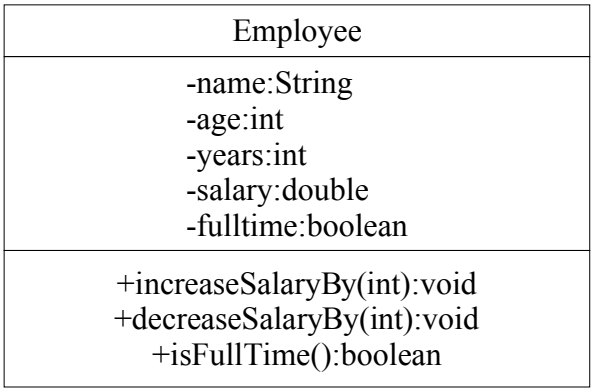
Goal:

In this exercise you are to play the role of a developer who is given a specification and description of the data structures and their operations. You then have to provide a Java implementation for this specification.

Instructions:

For each exercise make sure you provide a `main` method to run your code as well as **all** the test cases that you have used to test your code. Comment all of your code and provide any information about your homework in a separate text file called README.txt. Send **all** your files to *skotthe@ccs.neu.edu*.

1. Implement the Java class for Employee as depicted below

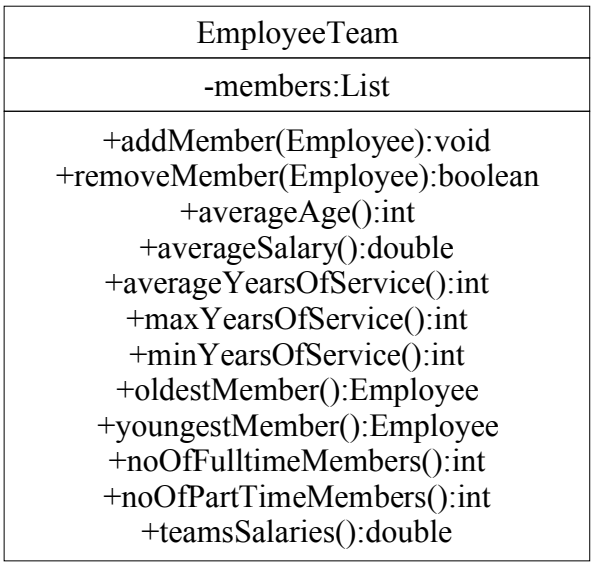


An Employee class contains the employee's name as a String, his age and years of service as an integer. The employee's salary is kept as a double and a boolean value is used to denote if the employee is full-time or part-time. For each of these attributes you should create setter and getter methods. Furthermore, an Employee object has the following methods.

- `increaseSalaryBy(int):void`, that increases the employee's salary by the amount given as argument to the method.
- `decreaseSalaryBy(int):void`, that decreases the employee's salary by the amount given as argument to the method.
- isFullTume():boolean, returns true if the employee is a fulltime employee and false if not.

(20 points)

2. Implement the Java class for EmployeeTeam of employee's as depicted below



An EmployeeTeam holds a list of its members, each member is an Employee. EmployeeTeam also provides the following methods:

- `addMember(Employee):void`, adds the Employee instance passed as an argument to the method to the list of team members
- `removeMember(Employee):boolean`, removes the Employee instance passed as an argument and returns true. If the employee instance does not exist then return false. (Hint: Use an Employee's name to check for equality between two employee instances)
- `averageAge():int`, returns back the team's average age.
- `averageSalary():double`, returns back the team's average salary.
- `averageYearsOfService():int`, returns back the team's average years of service.
- `maxYearsOfService():int`, returns back the team's maximum years of service.
- `minYearsOfService():int`, returns back the team's minimum years of service.
- `oldestMember():Employee`, returns back the team's oldest member.
- `youngestMember():Employee`, returns back the team's youngest member.
- `noOfFullTimeMembers():int`, returns back the number of full-time employees.
- `noOfPartTimeMembers():int`, returns back the number of part-time employees.
- `teamsSalaries():double`, returns back the sum of all members salaries.

(30 points)

3. You are asked to implement in Java sets of integers and some operations on them. A set contains integers only. A set can contain any number of integers, there is no limit on the size of the set. A set must support the following operations.

   (a) `addElement(int):void` , adds the integer given as an argument to the set.

   (b) `removeElement(int):boolean`, removes the integer passed as argument if the integer is found in the set and returns true, else return false.

   (c) `hasElemement(int):boolean` , returns true if the element passed as argument is found inside the set, false otherwise.

   (d)`intersection(Set):Set` , returns as a Set the elements that are found both inside the set instance passed as argument and the current set.

   (e)`difference(Set):Set` , returns as a set the elements that are part of the current set but are not found inside the set passed as argument to the method.

   (f) `isSuperSet(Set):boolean` , returns true if all the elements of the set passed as argument are also members of the current set.

   (30points)

4. Use the implementation of the binary tree given in class and add the following methods to the class BTree.

   (1) `printReverse():String`, prints the contents of the binary tree but in reverse order, i.e., right subtrees , then the current root then left subtrees.

   (2)`contains(int):boolean`, returns true if the tree contains a node with the same value as the integer passed as an argument to the method.

   (20 points)