CSCI221 Fall 2016
Programming Assignment 1, Due date to be announced, at least 2 days in advance.
Instructor: McCauley

Collaboration Policy: individual Assignment 100-Points

Program Requirements

You work for LinkedIn. At LinkedIn, we need to store information about companies that people work for or have created. We will create a `Company` class. Each instance of class Company will represent a particular company at LinkedIn.

Program Design and Specification

Class Company

`Company` has several attributes that are used to describe it, as well as methods that operate on them. Here are the attributes for `Company` that you must implement as instance variables. All of these instance variables must be private. You decide the name of each instance variable[1].

- name of the company (`String`)
- type of company (`char`): `D' for domestic, `O' for overseas, or `I' for international
- year founded (`int`)
- number of employees (`int`)
- parent company (`Company`) : the company that owns this company
- id number (`int`) : a number used to reference this company in a database under development.

More information about the Company instance variables:

- It is ok for two companies to have the same name. This happens a lot.
- The year founded is something like 1997 or 2005. Do not worry about invalid dates, i.e. do not write code that checks if dates are valid – assume it is valid.
- Number of employees is always non-negative. It does not make sense to have < 0 employees.
- The id number is just a number >= 0. Assume all given ids are unique, i.e. do not write code to check whether a company with the given id number already exists. If a company hasn't been assigned an id number yet, this attribute should be -1.
- The parent is a Company object that owns this company, i.e. it is not a String; it is a Company reference data type. The parent is null if no company owns this company.

Accompanying the declaration of each instance variable must be a comment that describes what it means, i.e. a description of the data stored in it. For example, the type instance variable stores one of three character values (D, O, I) that represents what kind of company it is, etc.

Whenever you write a method, define the pre- and post-conditions, i.e. what is expected before and after the method is called (see page 300 in the course textbook). Do this for each method your write and test in the `Company` class.

Add one assertion (assert in Java) to each method (except constructors) you design and implement. An assertion is a statement of what must be true at that point in the code (see page 343, 243, and 244 in the course textbook). When you run the program with assertions

[1] Note: value in parentheses indicate instance variable data type

on, the JVM will help you test your methods by paying attention to the assertions as your test code runs.

Company Methods

The `Company` class has the following methods. Play close attention to the parameters and return values of each method. The descriptions, while informal, are complete. When you write a constructor body ensure all instance variables are set to the appropriate values.

Constructor	Description and Preconditions (if any)
`Company( String name, int year, int employees, char type )`	Constructor: a new company. Parameters are, in order, the name of the company, the year the company was founded, number of employees, and what type of company it is. The company has no id number and no parent company. Precondition: All parameters are valid as described.
`Company( String name, char type, Company parent, int year, int employees, int id )`	Constructor: a new company. Parameters are, in order, the name of the company, its type, its parent, year founded, number of employees, and id number. Precondition: All parameters are valid as described, the parent is not null, and the id number is >= 0.

Accessor (i.e. Get) Methods	Description and Preconditions (if any)
`getName()`	Name of the company. Returns a String.
`getType()`	Type of company. Returns a char.
`getYearFounded()`	Year company was founded. Returns a int.
`getNumberOfEmployees()`	Number of employees in this company. Returns a int.
`getParent()`	The parent company. Returns a Company
`getId()`	Id number of company. Returns a int.
`getDescription()`	A String representation of this company. Its precise specification is discussed below. Returns a String.

Mutator (i.e. Set) Methods	Description and Preconditions (if any)
`setName( String s )`	Name of the company. Accepts a String.
`setType( char t )`	Type of company. Accepts a char that meets the precondition: t is one of 'D', 'O', and 'I'
`setYearFounded( int y )`	Year company was founded. Accepts a int.
`setNumberOfEmployees( int e )`	Number of employees in this company. Accepts a int that meets the precondition: e

	>0
`setParent( Company p )`	The parent company. Accepts a Company that meets the precondition: p is not null
`setId( int i )`	Id number of company. Accepts a int that meets the precondition: i >= 0, if not the id number is -1

Comparison Methods	**Description and Preconditions (if any)**
`isBigger( Company c )`	c is not null, and this company has more employees than c has. Returns a boolean
`isBigger( Company c1, Company c2 )`	Static method. c1 and c2 are not null, and c1 has more employees than c2 has. Returns a boolean.

Make sure the names of your methods match those listed above **exactly**, including capitalization. The number of parameters and their order must also match. Our testing will expect those method names, so any mismatch will fail during our testing. Parameter names will not be tested – you may change the parameter names if you wish to make them more descriptive.

The pre-condition should not be tested by the method; it is the responsibility of the caller of the method to ensure the pre-conditions are met. For example, it is a mistake to call setParent method with null for the parent argument.

Additional specifications:

- Use a package named `hw2.csci221` for all the classes you write for this assignment.

getDescription Method Format

Below is an example that demonstrates the required format of the String returned by this method. Your format must match the provided example.

"D company SuperMaster. Id 34. Founded 2005. Has 1000 employees"

Here are some points about the format.

- Exactly one blank space separates each piece of information and the periods are necessary.
- The `D' at the beginning means it is a domestic company. Use `O' for overseas and `I' for international.
- The name of the company follows the word "company"
- If the id number is unknown (i.e. -1) omit it entirely.
- The parent is not described in the provided format.

TestCompany Class

Build a suite of test cases in the main method defined in the TestCompany class. The main method will create an instance of the Company class, then proceed to test the object by calling methods to ensure the constructed Company works correctly.

Make sure that your test suite adheres to the following principals:

- Every method in the Company class needs at least one test case in the test suite.
- The more interesting or complex a method is, the more test cases you should have for it. What makes a method interesting or complex is the combinations of inputs the method can have, the number of different results the method can have when run several times, the different results that can arise when other methods are called before and after this method, and so on …
- Here is one important point. If an argument of a method can be null, there should be a test case that has null for that argument.

Program Documentation

Refer to the documentation standard posted in the content section on OAKS. Each method MUST be preceded by an appropriate Java doc comment. Use Eclipse short cuts (e.g. /** - return key) to generate comment stubs above class definition and methods. Follow the example given in class, or setup an appointment with instructor or TA if require more direction.

Program Submission

Create a ZIP file that only contains .java files, that is, no byte-code files (i.e. .class files) will be submitted. Furthermore, please do not include a file hierarchy (i.e. the package structure) – just one or more .java files. For the ZIP file you **must** use the naming convention:

<*lastname*>.zip
e.g., Munsell.zip

If the assignment is not submitted in the correct format – it will not be accepted – no exceptions! Submit the ZIP file via OAKS in the Dropbox that corresponds to the assignment. Resubmit as many times as you like, the newest submission will be the graded submission.

Grading Rubric

20 Points	Style: Comments and Indentation
80 Points	Functionality: • Program compiles (10) • Program runs (10) • For given inputs, program produces correct output (60)

If the submitted program does not compile: 0 of 80 points
If the submitted program compiles but does not run: 10 of 80 points
If the submitted program compiles and runs: 20 of 80 points
If the submitted program compiles, runs, and produces correct output: 80 of 80 points

The correctness of your program will be evaluated using test cases developed by the instructor. Late assignments will not be accepted – no exceptions (please do not email me your assignment after the due date, I will not accept it).

Please feel free to setup an appointment to discuss the assigned problem. I'll be more than happy to listen to your approach and make suggestions. However, I cannot tell you how to code the solution. Furthermore, code debugging is your job; please do not come to my office asking me to fix your code.