Name: ____ Lab Section: 05

CS161 Fall 2015 Programming Exam

Write your name above and return this paper to your lab instructor when you submit your exam. You must finish and submit your work using Check-In by the end of this lab.

Do your own work. Only look at your own computer screen. Do not browse, email, or use cell phones and other digital technology other than the computers in the lab.

During the exam you may use your Java text book (or ebook version if you let your instructor know), the Java API documentation website and the course website, but no other web pages.

Your grade will be based on:

- Correctness of each method
- Clarity of code and comments. Your code should be easy to read, and if there are non-obvious steps, they should be explained with comments.
- Object-oriented design: Does your code capture the idea of encapsulation
- Testing: Does your main method adequately test all aspects of your code?

You will write five java classes for keeping track of projects for a company. The names of the classes are Company, Project, Hire, Design, and Construction. The classes Hire, Design, and Construction must extend the class Project, to allow an instance of Company to use a single ArrayList to hold instances of Hire, Design and Construction.

Your classes must implement the following methods.

Company: class that is the primary application that you will run.

[**4 points**] **constructor:** Creates an empty ArrayList for holding items of type Project.

[**4 points**] **add:** Add a new Project, either a Hire, Design, or Construction.

[**4 points**] **toString:** Returns a string representing all Projects that have been added, by implicitly calling the toString method of each. Use `"\n"` in the String to make a new line for each Project.

[**3 points**] **main:** A main method that performs the test specified below.

[**4 points**] **Project:** parent class of Hire, Design, and Construction.

[**4 points**] **constructor:** Has two String arguments, a Project name and an expected completion date. Assigns these values to protected instance variables. It also creates an empty ArrayList of Strings in a protected instance variable for keeping track of employee names who are on the team to complete this project.

[**4 points**] **getName:** Returns name.

[**4 points**] **setName:** Changes the value of name.

[**4 points**] **getDate:** Returns the completion date.

[**4 points**] **setDate:** Changes the value of the completion date.

[**4 points**] **addToTeam:** Adds a person, given as a String, to this project's team.

[**4 points**] **addToTeamFrom:** Accepts another Project and adds all team members from that Project to this project.

[**3 points**] **toString:** Returns a string like `Project: DoStuff 5/31/2015 Jim Bill` if Jim and Bill are the team.

[**3 points**] **main:** A main method that tests each method in this class.

[**4 points**] **Hire:** child class of Project

[**3 points**] **constructor:** Has three arguments, two Strings for the project name and expected completion date, and a double for the salary. Call the parent class's constructor to assign the first two values, then assign the salary to another protected instance variable.

[**3 points**] **toString:** Returns a string like `Hire: Accounting 2/1/2016 45000  Jim Bill` if Jim and Bill are the team.

[**2 points**] **main:** A main method that tests each method in this class.

[**4 points**] **Design:** child class of Project

[**3 points**] **constructor:** Has two String arguments, for the name and the expected completion date. Just call the parent class's constructor to do all the work.

[**3 points**] **toString:** Returns a string like `Design: Shed 6/1/2016 Jim Bill` if Jim and Bill are the team.

[**2 points**] **main:** A main method that tests each method in this class.

[**4 points**] **Construction:** child class of Project

[**3 points**] **constructor:** Has two String arguments, for the name and the expected completion date. Just call the parent class's constructor to do all the work.

[**3 points**] **toString:** Returns a string like `Construction:  Shed 6/1/2016 Jim Bill` if Jim and Bill are the team.

[**2 points**] **main:** A main method that tests each method in this class.

Remember, you are required to implement a `main` method in every class that tests the methods of that class. After your tests in the main methods of each class work, change the main method in your Company class to perform the following steps.

[**4 points**] Compiles.

1. Add these to your Company in the order given.
   - A Hire project named "Accounting" with expected completion date "12/1/2015", a salary of 60000.0, and team members "Janet" and "Jim".
   - A Construction project named "Warehouse" with expected completion date "7/31/2016", and team members "Bonnie" and "Bill".
   - A Design project named "Penthouse" with expected completion date "3/1/2016", and team member "Art".

[**3 points**] 2. Print the company projects using the toString method in Company.

[**3 points**] 3. Add to the "Penthouse" design project all of the team members from the "Warehouse" consruciton project.

[**1 points**] 4. Print the company projects again using the toString method in Company.

Here are some reminders about how to use ArrayLists.

- Remember to `import java.utils.ArrayList;`
- Constructing an instance: `ArrayList<Type> list = new ArrayList<Type>();`
- Add an object `o` of type Type doing `list.add(o)`
- Get element at index `i` by doing `list.get(i)`
- The `indexOf(x)` method returns the index of the first occurrence of object `x` in the list, determined by the `equals` method for the type of `x`. It returns -1 if it does not occur in the list.

When you are done, combine your files into a **jar file** named `company.jar`

```
jar cvf company.jar *.java
```

and check-in your jar file using the Checkin page at CS161 web site. And you must turn in this exam form with your name on it to your instructor before you leave recitation.