ASSIGNMENT 4

COMP-202, Fall 2013, All Sections

Due: December 6^{th}, 2013 (23:59)

Please read the entire PDF before starting.

You must do this assignment individually and, unless otherwise specified, you must follow all the general instructions and regulations for assignments. Graders have the discretion to deduct up to 10% of the value of this assignment for deviations from the general instructions and regulations. These regulations are posted on the course website. Be sure to read them before starting.

Part 1:	0 points
Part 2, Question 1:	35 points
Part 2, Question 2:	30 points
Part 2, Question 3:	35 points
	100 points total

It is very important that you follow the directions as closely as possible. The directions, while perhaps tedious, are designed to make it as easy as possible for the TAs to mark the assignments by letting them run your assignment through automated tests. While these tests will not determine your entire grade, it will speed up the process significantly, which will allow the TAs to provide better feedback and not waste time on administrative details. Plus, if the TA is in a good mood while he or she is grading, then that increases the chance of them giving out partial marks :)

Part 1 (0 points): Warm-up

Do **NOT** *submit this part, as it will not be graded. However, doing these exercises might help you to do the second part of the assignment, which will be graded. If you have difficulties with the questions of Part 1, then we suggest that you consult the TAs during their office hours; they can help you and work with you through the warm-up questions.*

Warm-up Question 1 (0 points)

In this question you are to use the text files provided to you on the course website. Each text file refers to a particular theme. Use your research skills to find documentation about reading a text file in Java. A good class to look at is `BufferedReader`.

Now write a Java program to read one of the `.txt` files and print out every entry. It is important to handle exceptions.

Warm-up Question 2 (0 points)

Extend the program from warm-up question 2 to use an ArrayList which will hold Strings. As you read each entry from the `.txt` file, insert the entry into the ArrayList.

Then, create a method printList() to print out all the entries in the ArrayList. Also, create a method pickRandom() that returns a random String in the ArrayList.

Part 2

The questions in this part of the assignment will be graded.

Question 1: WordList (35 points)

Write a class `WordList`. A `WordList` should have one private property, an `ArrayList<String> words`. In addition, it should have two methods defined in it:

1. A constructor that takes as input a `String filename`. Your constructor should initialize `words` to be a new ArrayList and, using a `BufferedReader`, read all of the lines in the file `filename` and store it into the private ArrayList. Note that every line of the file should be considered as one "word".

   Because your method reads from a file, it is possible you will encounter an exception. This happens if the file does not exist for example. In this case, your method should NOT catch the exception and should pass it on to the calling function. Remember you can specify you want to do this by adding `throws IOException` in your constructor header.

   The reason to throw an exception, rather than catching the exception, is there is no good choice the constructor could make in the case the file was not found. Only the function *using* the constructor knows what to do in this case.

2. A method `getRandomWord` which takes nothing as input and returns a random `String` from `words`. Remember that you can use the `Random` class to do this. As in assignment 3, you should declare a `static` variable of type `Random` outside of any method and initialize it. This will avoid you getting the same words every time.

Question 2: Defining an Agent (30 points)

In this part of the assignment you will create an **Agent** class with different attributes, with the values for these attributes selected at random. **The agent class must store all of this information in private properties. It is up to you to choose a good data representation. Since they are all private properties we do not care what method you use to store the data but you MUST have your agent represent all these things.**

For example, the gender of the agent could be represented in several ways:

1. A boolean `isFemale` which is true or false based on gender.
2. A char where you store an abbreviation such as M or F
3. A String where you store the whole string "male" or "female"
4. etc.

For each of these, you will have to think what the best way to store them is. You should base these decisions on how you use these variables later on.

- Gender - Chosen at random (male or female)
- Birthday - Chosen at random
- Name - Depends on agent's gender and comes from WordList.java
- City where they were born - Randomly picked from WordList.java
- City where they live now - Randomly picked from WordList.java
- Current major - Randomly picked from WordList.java

In addition to these properties, you must provide several methods:

1. getCityNow() : Returns the city the Agent currently lives in
2. getCityBorn() : Returns the city the Agent was born in

3. getName() : Returns the name of the Agent
4. getGender() : Returns the gender of the Agent
5. getMajor() : Returns the major of the Agent
6. getBirthday() : Returns a String representing the date the Agent was born (e.g. 20th of July, 1985)

Moreover, since this is an agent, you have to give them a voice and make them say things. You have to provide us with these public methods:

- sayHello() - `println(this.name + " says:  Hello my dear")`
- sayCityBorn() - `println(this.name + " says:  I am from "+ this.cityBorn)`
- howOldAreYou() - This one has to be based on real time, not hard coded.

  `Calendar.getInstance().get(Calendar.YEAR)`
- sayGender()
- sayCityNow()
- sayMajor()

In order to test your code, you are to include a public method `whoAreYou()` that prints the content of all of your lists. `whoAreYou()` that would output a biography of your agent. We are expecting to know their name, gender, age, current city, where they is born, and their current major.

```
Output example from calling whoAreYou():
Jean says: "Bonjour, my name is Jean!"
Jean says: "I am from Baie-Comeau."
Jean says: "I am 101 years old."
Jean says: "I am majoring in Computer Science."
Jean says: "I live in Montreal."
```

Finally, you should provide a `static` method `generateAgent` that returns a "random" `Agent`. It should do this by first creating four different `WordList` objects based on the files for cities, majors, female names, and male names. These files are on the course webpage as examples (or you can make up your own, but please use the same file names).

You then use these four WordLists to create a random Agent:

1. Choose a gender at random
2. Choose a birthday at random (hint: you need to choose 3 random ints for this)
3. Choose a name at random, looking at either the female or male names (using the WordList) based on the gender of the `Agent`
4. Choose a major, a city born in, and a city currently in at random using the appropriate `WordList`

If any call to the WordList constructor throws an `IOException` you should initialize the appropriate entries to be default values. For example, if the city file failed to load, you should store a default value (a hard coded String) into the from city and current city properties. All other properties should work as normal.

Note that you should NOT do any file reading directly in the Agent class. You should only use the `WordList` objects.

Hint: Since you are writing a static method, you will be operating from a "static context" unless you provide an instance of an object to operate on. This means your code should look something like the following:

```
Agent randomAgent = new Agent();
.....//set values of properties.
// Note that since you are inside the same class you may access private properties
// by writing randomAgent.someProperty
...
return randomAgent;
```

Summary - The class should have:

1. Six member variables describing the agent
2. Getter methods for these variables
3. Six methods to let your Agent speak (such as sayGender() and sayName())
4. `whoAreYou()` method presenting the agent
5. One static method to generate a random agent.

Question 3: Agents chatting (35 points)

For this question you will have to create `DiscussionDirector.java`[1] where you will create two instances of your agent and make them speak to each other. In order to accomplish this you may need to create additional methods in `Agent.java` such as `sayHelloTo(String name)` which would give the following output,

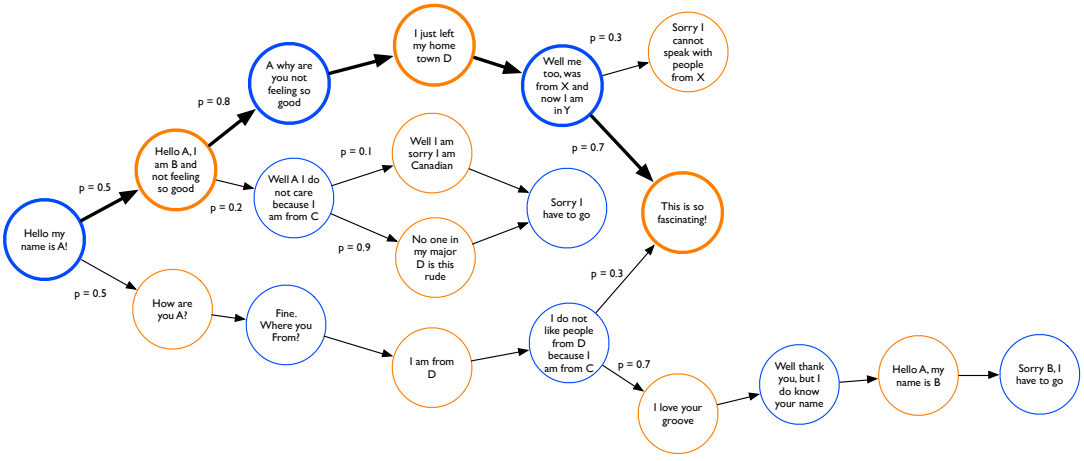
```
Paul says: "Hi there, my name is Paul!"
Alice says: "Hello Paul, nice to meet you, my name is Alice, how are you?"
```

In order to start a discussion, you will have to create the method `discuss` in your DiscussionDirector class. This method will take two Agents, and will print out a discussion between them.

The discussion that you will create will have to follow the structure represented below. It represents the possible conversations the two agents can have, the blue circle is agent *Jean* and the orange circle is agent *Lea*.

Start at the left portion of the graph with the circle saying "Hello my name is A!" (using Agent A's specific methods.) Then randomly choose with probability of 50% whether to respond with "Hello A, I am B and not feeling so good" or "How are you A?". Then continue to follow the flow chart until you reach the right side of the chart, making decisions based on the probability in each split in the chart.

The actual text for each line is not important, but you must have the same structure. Try to make the text make sense and not offensive.



For example, one discuss() call might follow the bold line in the chart, and give the following output:

[1] Check out `http://goo.gl/hnUc2p` for a presentation of Valve's discussion director

```
Jean says: "Bonjour, my name is Jean!"
Lea says: "Welcome Jean, nice to meet you, my name is Lea and I am not feeling so well."
Jean says: "Hoo Alice, why are you feeling so low?"
Lea says: "Well I just left my home town, Natasquan."
Jean says: "Me too, Natasquan is a fantastic town, I just left my home town Calgary as well."
Lea says: "This is fascinating, but really I have to go, see you Jean!"
...
```

If you want to be creative, in addition to implementing the chart below, you may choose to have other model discussions.

In this case you should add other methods to your class (e.g. `discuss2`). Five bonus points will be given for a flow chart and discussion of your own, it has to be comparable in length to the presented one.

Note that only the main chart will be marked.

What To Submit

You should submit your assignment on MyCourses. In order to do this, you will need to make a `zip` of the file. You can do this on windows by following the instructions at this link: `http://condor.depaul.edu/slytinen/instructions/zip.html`. On a mac or linux, you can find instructions at `http://osxdaily.com/2012/01/10/how-to-zip-files-in-mac-os-x/`

You should submit a zip file called `Assignment4.zip` with the following files inside of it.

`WordList.java`	
`Agent.java`	has to include method `void whoAreYou()`
`DiscussionDirector.java`	has to include `void discuss()`

`Confession.txt` (optional) In this file, you can tell the TA about any issues you ran into doing this assignment. If you point out an error that you know occurs in your problem, it may lead the TA to give you more partial credit. On the other hand, it also may lead the TA to notice something that otherwise he or she would not.

Marking Scheme

Up to 30 points can be removed from bad indentation of your code as well as omitting comments, or missing files. Marks will also be removed if your DAG is non readable. Marks will be removed as well if the class names are not respected.

Question 1

Used throw try-catch properly	10	points
Code handles error and tells the user of missing file	5	points
Read files and creates ArrayLists	5	points
Have methods that returns random elements from a list	5	points
Static random variable	5	points
Closes files after reading and correctly used the BufferReader	5	points
	35	**points**

Question 2

Creates randomly an agent and handles exceptions	10	points
`whoAreYou()` fully presents the agent	5	points
The agent is presented through calling a set of methods	5	points
The agent is stored within class private variables	5	points
Agents are created within the constructor and it is robust (no errors)	5	points
	30	**points**

Question 3

The DAG and the code is correct	10	points
The code is robust, *e.g.* lets the user know which file is missing	10	points
Information is passed and saved through the discussion	5	points
Calling `discuss()` leads to different discussions	5	points
New methods were created in `Agent.java`	5	points
Original discussion and diagram	5	**Bonus points**
	35	**points**