# COMP 111

# Lab 2 – Winter 2019

# Scenario

Sam Snake, the CIO of Snake Oil, is beaming as you enter his office. His greeting is warm, “You are off to a great start!” He ushers you to a chair across from his desk. As you settle in, you nervously position a folder on his desk that contains your ideas for the next version of the station locator project.

“Here’s a token of our appreciation!” he laughs as he takes a neat, white box from his desk drawer and slides it towards you. “Actually it’s the latest giveaway item at our ‘SOUP’ stations.”

You are not sure whether to open Snake’s gift, but since he continues talking, you decide to place the box next to your folder atop the desk.

“I was impressed with your coding of ‘version 1’ and was intrigued by your suggestions for a design improvement. I really like the idea of separating out the location stuff from the Station class.”

You had noticed the Station class your predecessor Grace designed had both data and methods specific to Snake Oil stations (like the description, price per gallon, and calculation of the price per GGE) combined with data and methods associated with locations (like latitude, longitude, and the calculation of distance between two locations). It occurred to you that, not only do stations have locations, so, too, do customers – and that, by principles of object-oriented design, it makes sense to propose a separate Location class.

“I think this is brilliant.” Mr. Snake is pleased. “So the Location class will have latitude and longitude as instance variables. And a method to calculate the distance between two locations.” Then the CIO pauses, deep in thought. Out of the corner of your eye, you notice the gift box you placed on the desk seems to have moved. You’re not certain, but you thought you had carefully placed it right next to your folder.

Mr. Snake picks up the conversation, “And the latitude and longitude instance variables of the Station class are replaced by a single instance variable of type Location.”

“Yes,” you acknowledge.

“I like this design,” the CIO exclaims, “Is this what they call clipping and adhesion?” he inquires.

“Coupling and cohesion?” you respond.

A blush crosses Mr. Snake’s thin face, “Yes. I would like to better understand ‘coupling and cohesion’ some time. But I can see this is good use of abstraction! Clearly you are well prepared – where did you go to school?”

You simply reply, “Franklin.” Sam Snake nods knowingly – there was no need to explain you meant Franklin University.

Mr. Snake shifts gears. “The calculation of the distance in miles between two locations needs to be improved in this next version. Do you have suggestions?”

You open your folder and pull out a page of notes and hand them to Mr. Snake.

1. **First convert latitude and longitude (in degrees) to latitude and longitude in radians  
   NOTE: use the Math.toRadians() method**
2. **Use the formula  
   3959.0 \* ArcCos[ sin(lat1) \*  sin(lat2) + cos(lat1) \* cos(lat2) \* cos(lon2 - lon1) ]**

**Use Math.acos() for ArcCos, Math.cos() for cosine, Math.sin() for sin  
3959.0 is the approximate radius of the earth in miles**

As you do so, you notice the gift box seems to have moved even further from the folder. You explain, “To get greater accuracy, we need to use the *Great Circle Distance Formula* – it uses some spherical geometry.” You are pleased with yourself and the research you have done.

Mr. Snake has fallen silent studying the formula. The box. What is going on with the box? You slowly inch your hand towards the gift box on the desk. As your fingers make contact, you feel an intermittent vibration from within the container. As you lean in, you are sure you hear an almost imperceptible sound emanating from the box.

Without looking up, Snake interrupts your exploration. “OK. Let’s implement this *Great Circle Distance Formula* in the next version. And let me confirm that will be done in a method of the new Location class.”

You nod and add, “The implementation of the calcDistance method of the Station class must be modified to call on the calcDistance method of the Location class.”

“And you will update the test method in the StationTest class to reflect use of this new formula?” Snake adds.

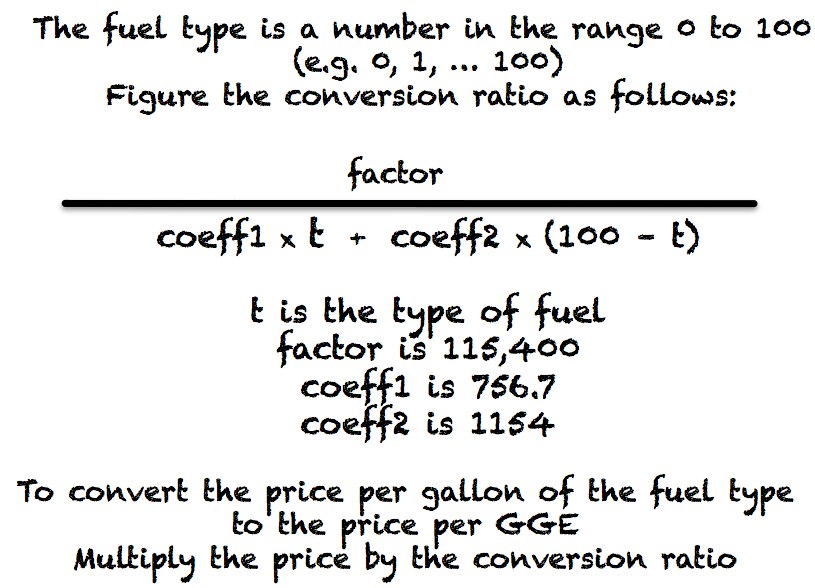
You reply, “I have already done that – I created a spreadsheet to generate expected values for test cases for calculating the distance using the *Great Circle Distance Formula*.”

“Spreadsheet? Why, if you are coding it” Snake shoots back.

“The ‘test cases’ in TDD should provide an independent test of calculations a programmer has implemented. We pre-calculate an expected value – using a manual approach or alternative source – and then check what we expect against what our code actually produces.”

Your comment stimulated Snake’s memory – you could see it in his face.

“I have also added new test methods to reflect other changes made to the Station class,” you continue. You pull your preliminary skeleton class definitions from your folder and hand it to Mr. Snake to review. But you take care to check the location of the gift box before you reach!

“One more thing,” Mr. Snake intones, as he scribbles on a note pad, as he scratches his head. “Grace oversimplified the conversion of the price per gallon of an alternative fuel to a price per gasoline gas equivalent. I don’t have time to explain the details now, but it’s more like this.” He tears off a sheet from the pad and sends it spinning across the table towards you.

You study his notes (see them at the right).

“Can you incorporate this fuel type idea and change the calculation of price per GGE?” Snake peers at you anxiously.

After a few seconds, you nod, “Yep. A new instance variable, a change to one of the constructors, and a new implementation of calcPricePerGGE.”

Mr. Snake looks pleased, but warns, “I am not absolutely sure about my factors and coefficients, so make it easy to change those.”

You make a mental note to use final class variables for those constants.

Mr. Snake summarizes, “So you will be rewriting several methods of the Station class to use the new Location class, adding support for a fuel type, and completing all the methods of the Location class.” Seeing you nod, the CIO inquires, “How are you coming along with Test Driven Development?”

“I think I get it,” you say. “And I can pattern what I do in LocationTest after what Grace did in the StationTest class.”

Mr. Snake stands and warmly extends his hand. “Splendid. You are doing great work. Let’s have the next version in two weeks!”

Your anxiety about the contents of the box mixes with the reality of another two-week deadline. You collect your folder, nervously pick up the box, and turn towards the door. You hope that the CIO doesn’t notice you are sweating!

Mr. Snake calls out with a chuckle, “I hope you enjoy the new Snake Oil giveaway! It’s a toy snake with a battery-operated rattler. I think you’ll find it is pretty authentic.”

You manage a weak, “Thanks for the gift,” as you leave the room.

# Details

As you return to your desk, you think of some additional questions to ask the CIO. You will send him a quick text message – you don’t plan to go back to his office any sooner than necessary! You ask him about initial values to assign instance variables for the constructor that has no parameters. His quick response confirms your assumptions:

* Latitude, longitude and price per gallon should be 0.0
* The station description should be set to “NO DESCRIPTION PROVIDED”
* The station fuel type should initially be 0

# Assignment

1. A BlueJ *starter* project will be provided to you. It will contain five classes – Location, LocationTest, Station, StationTest, and StationDemo. The version of the classes provided have many of the methods completed to help get you started and provide the syntax and approach to use for the methods you will code. You need to complete and code methods – following any guiding comments within the code and meeting requirements described in this write up.
2. You must use the provided BlueJ starter project and develop your solution using the BlueJ IDE.
3. Do NOT change the name, return type, or parameter order/type of any of the class or test class methods. These are used by Web-CAT to grade your submission, and you will end up losing points if they are changed.

Each class (in this assignment, Station and Location) constructor and method should be tested by a unit test method in the test class (in this assignment, StationTest and LocationTest). Besides serving to verify that your code is accomplishing what you intended, examining the unit test methods will help you understand how the objects should behave (how the class methods create, access, and mutate the object). That is, first writing the test methods helps guide your class coding efforts. Follow these steps to complete this assignment:

* 1. Find and correct any syntax errors so that the classes in the project compile cleanly.
  2. Review the first JUnit test method, completing the code if necessary. Note that proper testing includes making manual calculations to verify that expected values for a method action are equivalent to actual results.
  3. Once you feel the unit test method is a valid and robust test of the method’s expected behavior, review the corresponding class constructor or method and complete the coding if necessary.
  4. Run the JUnit test for the method. If it does not pass, review and correct the code for this method.
  5. When the test passes, continue in like manner for the next and subsequent methods, iteratively completing tests and developing your class or classes. All class methods must be tested.

1. All classes must compile cleanly.
2. Check your programming style using the “Check Style” tool provided within BlueJ (Tools 🡪 Checkstyle).
3. Document the overall project in the README file as directed in the Action Items for the Lab.
4. Submit your completed lab to Web-CAT using BlueJ (Tools 🡪 Submit) and review the Web-CAT results for errors. Repeat the above steps as needed to resolve any errors.
5. The class StationDemo included in the project file demonstrates the use of the Station and Location classes, calling on its constructors and methods and displaying results. While it supports your understanding of the lab, it is not part of the actual solution. You can modify it if that is helpful to you, but the driver class will not be examined by Web-CAT nor graded by your instructor.
6. An excel spreadsheet is included in the *starter project* folder. You can use it to create your test cases for the calcDistance methods. Creating spreadsheets to test complicated calculations is a practice you might want to use in this and future labs.