For this week's programming activity, we are going to play around with Maps! Unlike this week's lab, this activity will use Maps in a more standard (1-dimensional) way with a less directed system of hashing. You may also work in pairs for this activity.

The Background

The search for non-human life in the universe has been a very old project that has tasked even the scientific minds of scholars such as Nikola Tesla. It's such a prevalent area of research that it has its own acronym: SETI (Search for ExtraTerrestrial Intelligence).

This area of research challenges itself to seek out detectable waves from space that could be interpreted as an intelligent communication from another intelligent species that has created the technological means to broadcast information across the stars (potentially important communiques, potentially their version of Star Trek or CSI: Miami).

To detect these very delicate signals, large radio dishes are created and maintained to point at different parts of the cosmos in the hopes that something sensical can be picked up in amongst the noise of space (and humanity stomping about the place and creating their own noises).

To mitigate these problems, research stations pick out of the place areas elevated away from the troubles of humanity and utilize many radio dishes spaced out so that they can catch the cosmic waves from slight variations in angle and better identify the signal they are receiving.

(It's complicated and I'm neither an astrologist nor someone deeply invested in SETI)

The Premise

Now, these stations can have a problem: they're very complicated machines that need to be incredibly precise to work properly. A little bit out of alignment and suddenly what you're hearing has nothing to do with what you're aiming at, and you're going to think that a football game being sent to your portable radio is an important space message (though if you made some unwise bets it may be a *very* important message to receive).

So, you are going to write a program that would help an aspiring SETI research station manage their radio dishes! And we are going to use a map to do this as well for quickly checking in on how well calibrated our dishes are.

We're also going to make this into a bit of a game.

The Program - General

Your program should model a research station that is composed of radio dishes. Each radio dish has an identifying name (you may use a system of naming such as "Alpha", "Beta", "Charlie", etc. or some other) and a calibration percentage (value in the range [0.0, 100.0]) that denotes how well honed-in it is on a space signal that it is interpreting.

That calibration will influence how we interpret a message from space. As we are not actually talking to aliens (or cosmic radiation), you will create a test phrase that each radio dish is trying to listen to and interpret. The radio dish's current calibration is how likely it is to interpret each character of that message correctly!

For example, if we had a message such as "Today is a good day to sleep in", and our radio dish were at 50% calibration, the message may end up being garbled as "T##d#y is a###od#da##t# sleep#in", where the '#' symbol denotes a failure to interpret the character.

Recall that statistics owe us nothing, so even though we are at a 50% probability for each character to be interpreted we could see a statistically low outcome of most of the message being readable or unreadable. You do not have to ensure that the message it interprets is exactly a 50% correct interpretation.

You should have a series of different messages available to be heard by the radio dishes that are a secret from the user to create some variety in what they may receive from their radio dishes. As we want this to be a bit of a game, make sure there is enough variety that someone replaying the game could not immediately guess at the message from the first few characters (or dynamically generate the message? Your call!)

The Program – Workflow: Setup

The way your program will work is it will allow a user to create a research station and add a number of radio dishes to it that are named and start at 100% calibration. These radio dishes should be stored in a map of its name (a String) as the key to a radio dish object as the value.

This program will be terminal-based, so the user may say something like 'add station Bravado' to add a station with that name. They should be able to also put in a command such as 'check station Bravado' to get a print-out of that station's name and its current calibration level, or a command such as 'check stations' to get a print out of all of the station's names and calibration levels.

You have been given a MapADT interface to implement your own map class. Note that your Map implementation should use generic types for both the Key and Value, and not be customized specifically to your foreknowledge of it being used for a String and radio dish.

When ready, the user can tell the program to begin 'listening' to a signal (perhaps with a command like 'start processing' or 'start listening'). This should pick a random option of the prepared messages you have created and begin displaying the process of each radio dish interpreting the message.

The Program – Workflow: Listening Process

We will think of this interpretation as a turn-based process that treats each individual character being 'heard' by all the radio dishes as one turn. On each turn the user will be shown the current progress for each radio dish on listening to the message and be able to act.

After each display of the current progress of message interpretation, the user will have several options but only be able to perform one of them per 'turn':

- Check in on the current calibration of their radio dishes.
- Fix the calibration of a specific radio dish (no fixing all of them at the same time!)
    - This will recover 20% of its calibration per ensuing turn (during which it only produces '#' characters) until it is back to 100%.
    - This repair process is automatic once it is begun, so the user may still take an action next turn, including telling another radio dish to fix itself.
- Try to interpret the message.
    - The user will input a string and be told what characters are correct (so the display of the current progress of interpretation should include overall what they have correctly interpreted thus far between guesses).
- Give up on the current message entirely and go back to pre-listening setup.
    - This would allow them to add or remove radio dishes to change their research station setup, and then restart the interpretation with another randomly selected message once they initiate it again.

After the user takes an action, reduce the calibration of each radio dish by a random amount between 0% and (100% / # of radio dishes). So, if there are 5 radio dishes, they each will lose between 0% and 20% calibration.

Remember to inform the user of what commands they can take; the person grading this will not be the same person who wrote it!

Once the user has correctly identified the message or given up on it, they will return to the initial phase of the program where they can change the setup of their radio dishes. If they correctly identified the message, tell the user how many guesses it took them (how many 'try to interpret' actions they took).

Make Sure To Plan

As this program has a few moving parts and different 'phases' of behavior, we will be planning ahead and making a UML diagram/documentation of our plans to get things sorted out before we start programming. The focus of this design process is preparation more than explanation.

What To Submit:

- The UML Diagram
- Your planning documentation
- Your program implementation

You will be assessed based on the quality of your design preparations (give these serious consideration!), how well your program works according to the descriptions given in this handout, and the quality of your code (recall that this incorporates class architecture now).

Include as an addendum to your planning documentation a brief reflection on how what you ended up doing compared to the plans you had made. If you worked with a partner for this activity, include a brief analysis of the contributions you each made.

Due: November 25’th, 2023 @ 9 PM

Late submissions are not accepted without prior discussion to extend the deadline.

I will not answer questions about the assignment asked after November 24’th @ 9 PM.

This activity may be completed in pairs or solo. You may discuss course concepts with your fellow students and how to approach a problem, but your work should be that of yourself or just the two of you who have decided to work together. Include in your comments any websites that you consulted during your work and what contributions they made.

Information gathering is a valuable skill and knowing how to find information online is important to be able to do; letting me know what information you needed to seek out allows me to know what to focus on teaching in the future.