MORAVIAN COLLEGE

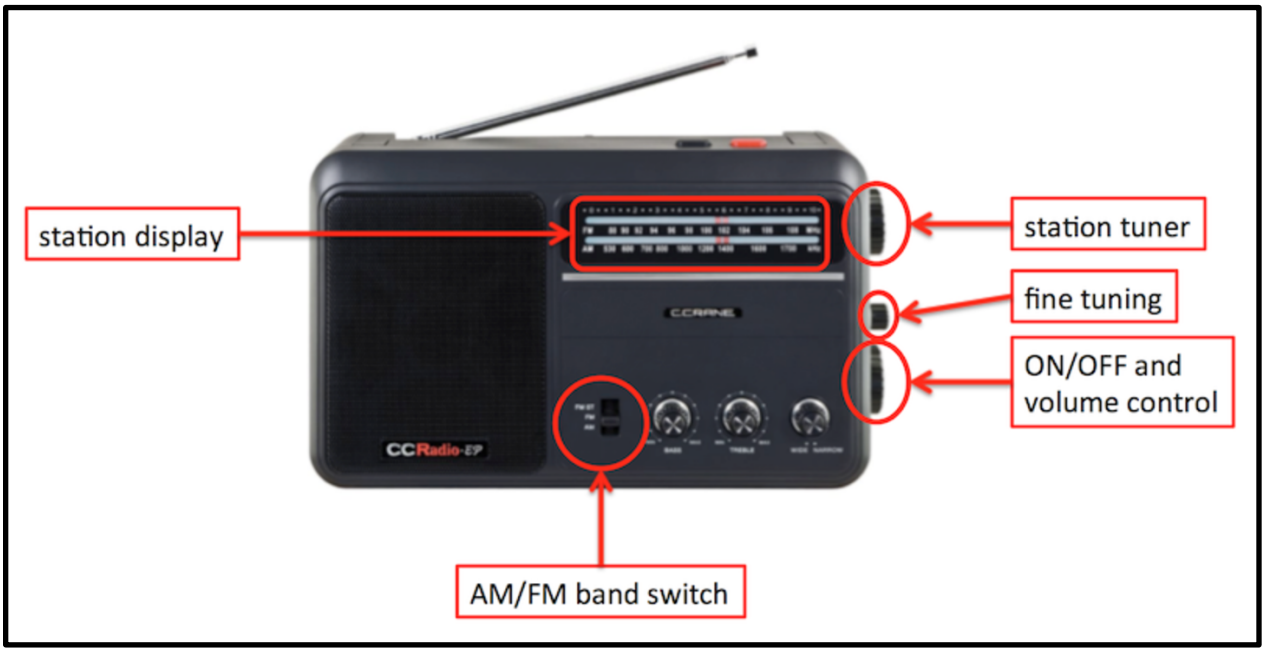
Mathematics & Computer Science Department

Computer Science II - Spring 2020

Lab 03

Introduction

Back in the days, before Spotify times, there was this thing called R-A-D-I-O? Below is how a typical radio looks like, just in case you've never heard of it.



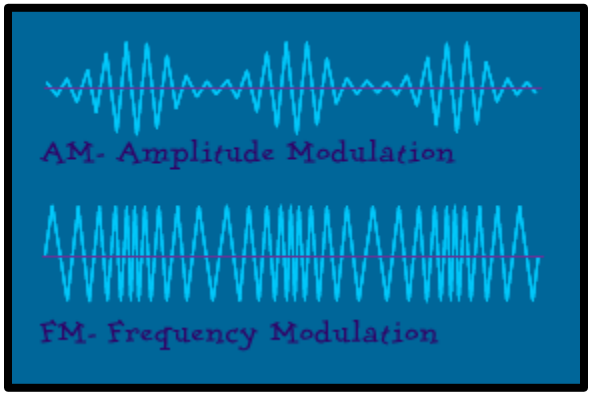
In this lab you will write a class that models a radio. But first let me give you some information about radios to help you write your class. To use a radio you first need to turn it on. There is an ON/OFF knob and when you turn it you hear it click! The same knob is used to control the radios volume, which ranges from 0 up to 82db.

There are typically two radio bands you can choose to listen to: the AM (Amplitude Modulation) and the FM (Frequency Modulation). Modulation defines how sound waves get encoded using electromagnetic waves. The AM modulation works by changing the amplitude (the strength) of the signal when encoding a sound wave, while the FM modulation technique changes its frequency (how many times the signal changes per second).

MORAVIAN COLLEGE

Mathematics & Computer Science Department

Computer Science II - Spring 2020



Source: pbs.org

A radio can be tuned to a station at a specific frequency. A typical radio has a "station tuner" knob that allows changing the current radio station. In order to model the `Radio` class, you need to know what is the range of frequencies reserved for the AM and FM bands. In this lab let's assume that:

- AM band uses frequencies in the 530 to 1,690 KHz (kilohertz) range, while
- FM band uses frequencies in the 88 to 108 MHz (megahertz) range.

As you can see, the FM band uses much higher frequencies compared to the AM band (the prefix Mega means 1,000 Kilo). The higher frequencies provide for a better signal quality. Unfortunately, the higher frequencies attenuate (fade) faster which limits how far the signals can travel. Therefore, FM signals have a shorter range compared to AM signals.

Another thing worth mentioning is that in most (traditional) radios the "station tuner" knob changes both the AM and the FM frequencies as you move the knob forward and backwards. Also, the rate of change is different for the two bands. For example, moving the FM frequency from 88 to 108MHz (therefore moving UP 20MHz) roughly corresponds to moving the AM frequency from 530 to 1,690KHz (moving UP 1,160KHz). In other words, an increase of 0.5MHz on the FM frequency corresponds to an increase of 29KHz on the AM frequency.

MORAVIAN COLLEGE

Mathematics & Computer Science Department
Computer Science II - Spring 2020

Instance Variables

Based on the given description of a radio, identify instance variables and constants you think are needed to model a `Radio` class (use the table below). It is always better to use named constants rather than hard code constant values. As you identify your radio's instance variables and constants, specify the value, or range of values, allowed for each one of them. These values can help you choose types and constraints for each variable.

Identifier	Variable or Constant	Value or Range	Data Type

Before continuing, imagine that you are using a radio with the above identified variables. Can all operations be performed? This may help to identify additional instance variables or constants to add to the list.

Constructors

What would be the initial state of a radio if you were to create one without any prior information? Use the parameterless constructor for this type of initialization. You should also define a *parameterized* constructor to allow the user to set the initial band. Regardless which constructor is used, a radio object should always be created in the OFF state, with the volume set to `0db`, and the initial AM and FM stations set to their minimum possible values.

Methods

Your radio model must allow the user to:

- turn a radio ON and OFF,
- increase/decrease the radio's volume (making sure you respect the minimum and maximum values)
- switch the radio's band from AM to FM and vice-versa,
- select a station by tuning the frequency forward or backwards, and
- display the radio object information, including: whether it is ON or OF, its current volume, band, and (correspondent) station; if the radio is OFF no need to show volume, band, or station. Use the toString() method for this.

Creating and Using a Radio

Now that you have a model of a radio, open the `RadioDriver` class and do the things you are asked to do (instructions are embedded in the source code).

If your program runs correctly, you should get the following output:

```
The radio is OFF
The radio is ON, volume is set to 40dB, tuned to 88.0MHz FM station
The radio is ON, volume is set to 40dB, tuned to 104.5MHz FM station
The radio is ON, volume is set to 40dB, tuned to 1487.0KHz AM station
The radio is ON, volume is set to 30dB, tuned to 1487.0KHz AM station
The radio is ON, volume is set to 30dB, tuned to 994.0KHz AM station
The radio is OFF
```

Rubric

+5	instance variables and constants (+0.5 for each variable)
+1	all instance variables are private
+3	constructors correctly defined (+1.5 for each one)
+2	toString method implemented correctly
+9	all of the other methods defined correctly (+1.5 for each method)
+8	driver class as asked (+1 for each TODO)
+2	correct output is displayed

MORAVIAN COLLEGE

Mathematics & Computer Science Department
Computer Science II - Spring 2020

Hints

Implement one Radio class method at a time and then test it by adding appropriate code to the driver main method.

Implementation order:

1. instance variables and constants
2. default constructor
3. toString method
4. constructor with parameter
5. on your own ...

toString Method output will depend on whether or not the radio is on. If it is off then the output would be "The radio is off.". If radio is on, then output give status of all settings.

Volume moves up and down by 1 for each activation of the method that controls volume. Use private helper methods to move volume control up and down by 1 and checking for appropriate limits. Volume cannot be less than 0 or greater than the given max constant.

Make public methods that allow use to specify a change in volume by an amount specified by an argument.

The FM band is only allowed to move in increments of 0.5 frequency.
How much does the AM band frequency change per 0.5 FM frequency?

When either frequency is changed, both of the frequencies are modified regardless of the band that is currently being used.