Assign-05

Learning From Your Elders …

In this assignment, you will be modifying your Assign-03 code to use *inheritance*. I need you to create a class called ***PioneerCarRadio*** that inherits its radio operation functionality from the AmfmRadio class (so you could say that a *PioneerCarRadio* **is an** *AmFmRadio*).

One of the features that PioneerCarRadio adds is a user interface based on buttons (represented by keypresses obtained through a `getch()` function call) as follows. There is no more menu driven input!

Keypresses:

On/Off	o (lower case "O")
Volume up by 1	+
Volume down by 1	_ (underscore)
Scan up	=
Scan down	- (minus sign)
Switch band (e.g. am to fm to am)	b
Choose button 1 to 5	1 to 5
Set button 1 to 5	Shift+1 to Shift+5

- You don't have to worry about key repeat (i.e. whatever keys you receive, process).

Display

The user should never see the keystrokes that they enter (which is why you're using `getch()` instead of `getche()`). **After each <u>valid</u> keystroke**, you should display the information in the following format (including spacing):

```
Pioneer XS440
Radio is on
Volume: 4
Current Station:  800 AM
AM Buttons:
1:  1000, 2:  1210, 3:   800, 4:   700, 5: 1300
FM Buttons:
1: 102.1, 2: 104.3, 3: 100.1, 4:  99.3, 5: 89.7
```

and (when the radio is off)

```
Pioneer XS440
Radio is off
```

Assign-05

Learning From Your Elders …

Guidelines:

- Start with the radio off.
- Do nothing for invalid keystrokes.
- Exit the program only when the user presses 'x'.
- Move all keystroke handling code **out of `main()`** and into the PioneerCarRadio class.
    - To do this, you'll be developing a new method within the PioneerCarRadio class
- Do not display any kind of instructional menu saying what the keys do.
- The output should look **exactly** as above
    - I expect you to do this **without changing** the ShowCurrentSettings() method within the AmFmRadio class
    - Also remember to have a print statement within your PioneerCarRadio's destructor indicating that the PioneerCarRadio is being destroyed
- There should be **no output** from the **AmfmRadio** class
    - This is why you put in the code to turn off output in the previous assignment.
- Do **not** make or promote any data members of the AmfmRadio class to be protected and/or public.
- Put the source code for your new class in *PioneerCarRadio.cpp* and *PioneerCarRadio.h*.
    - Try to avoid changing the contents of AmFmRadio.cpp and AmFmRadio.h
        - the exception to this is fixing things that you did wrong in the previous assignment
- Put the test harness code for your new class in a source module named *carDriver.cpp*.

Style and Convention Requirements

- Make any new data items that you create private.
- Your code must not unnecessarily duplicate similar code segments.
- We are going to continue to use the *Classic* commenting style in this assignment (i.e. not DOXygen)
    - Add full class header comments in each of your source files (even the .h file). For the descriptions of each, describe what the purpose of the class is, what the class can do (generally), what its purpose is, etc.
    - Add full method header comments on each method you develop and don't forget your inline comments as well
    - As discussed in class – also make sure to comment your private data members and any constants (private or public) as well
    - Make sure that all comments that exist in the example are correct.

Other Stuff

- ZIP up all the source files : AmFmRadio.h, AmFmRadio.cpp, PioneerCarRadio.cpp, PioneerCarRadio.h and carDriver.cpp. Submit the ZIP to the appropriate eConestoga Dropbox by the deadline.