UNIVERSITY OF JOHANNESBURG

Practical 4 (due 2023-08-18 @ 09:00)

The purpose of this practical is for you to become familiar with the basics of inheritance and polymorphism.

You may use the memo for Practical 3 in the creation of Practical 4. This will not be considered as plagiarism if you include a comment indicating you have done so.

Create a class called `IIoTScanner`

- This class includes a pure virtual public void member function called `scan()`
- The class includes a private property `_scanLevel` of type `ScanLevel`, an `enum` with `HIGH` and `LOW` values.
- The member function must take an `IIoTMonitor` reference as a parameter

Create classes `IIoTSimpleScanner` and `IIoTAdvancedScanner` that inherit from `IIoTScanner`

- The two classes must redefine the scan function differently.
  - The `IIoTSimpleScanner` class function must loop through all the sensors in the array and check if any of the sensors have a reading of 0 for either the temperature, humidity, pressure or light intensity. If any of the readings is 0 for a particular sensor, the sensor must be set as inactive.
  - The `IIoTAdvancedScanner` class function must loop through all the sensors in the array and check if all sensors have a reading of 0 for either the temperature, humidity, pressure or light intensity. If all of the readings are 0, the sensor must be set as inactive.
  - The constructor sets the `_scanLevel` to `HIGH` for the `IIoTAdvancedScanner` and `LOW` for the `IIoTSimpleScanner`.
- The `IIoTMonitor` must be updated so that
  - it **has-a**(n) `IIoTScanner` whose lifecycle it manages directly (it must be instantiated from the free-store when the `IIoTMonitor` is created and de-allocated when the `IIoTMonitor` is destroyed).
  - it has a `scan()` member function that delegates the scan task to the correct class (`IIoTAdvancedScanner` or `IIoTSimpleScanner`).
  - The `toString()` member function must be updated so that inactive sensors are represented by the character '**X**'.
  - The constructor must take in an additional property, a boolean which is true creates an `IIoTAdvancedScanner` and if `false`, `IIoTSimpleScanner`.
- The output will vary depending on the type of `IIoTScanner` used.

Note: The creation of virtual functions will be shown during tutorials next week.

Upload and submission

- Create a PDF design document named **Design.pdf**
- When your program is working and you have created a design document, you must add your work to an archive file in the **zip** compression format. The name of the archive must be in the following format:

  `SURNAME_INITIALS_STUDENTNUMBER_CSC01B1_2023_PX.zip`

  e.g. for a student called Anne Student with student number 123456789

  `STUDENT_A_123456789_CSC01B1_2023_PX.zip`
- The archive must contain the following directories / folders:
  - **src** – containing the C++ source code needed to compile your program
  - **bin** – containing an executable generated from your source code
  - **doc** – containing your design document
- Submissions that do not compile will be capped at 40%.

Mark sheet		
	Design	10
	IIoTScanner base class	10
	Appropriate member visibility	10
	scan pure virtual member function	10
	Derived classes	10
	Constructor chaining	10
	Overriden scan function in IIoTSimpleScanner	10
	Overriden scan function in IIoTAdvancedScanner	10
	Scan function in IIoTMonitor that calls scan polymorphically	10
	Demonstration of functionality in a main function	10
	Total	/100