**Home exercise 9**

**Course name: Object oriented programming and design for engineering**

**Course number: 157109**

**Subject:** **Design patterns – Observer, Template method, Singleton**

General:

In this exercise we will develop a **weather monitoring system.** The weather station will have different components which provide information related to the weather. This information will be available to anyone who wishes to observe it. When developing this system we will use useful design patterns.

We will build the system in stages. This assignment is the first stage.

In each stage, you should present the **UML** and write the **code**. In some cases, code will be provided as assistance.

**Stage A: Temperature, pressure, and pressure trend [TemplateMethod, Observer]**

The weather monitoring system contains two sensors which measure the temperature and pressure and a component for measuring the pressure trend. The temperature is measured in Celsius, the pressure in millibars and the pressure trend has one of the following optional values: rising, falling, stable. The sensor components have an abstract function called *read* which performs the measurement and is implemented by sensors of version Nimbus1. The Nimbus1 sensors simulate the measurment by returning a random number which indicates the temperature (between 0 and 40) or the pressure (between 950 and 1050). The pressure trend component is updated every time the air pressure changes. This component saves the three last values of the pressure - a, b, c. This component returns “rising” if b<c && a<b, “falling” if b>c && a>b, and "stable" in all other cases.

The information of the temperature, pressure and pressure trend components is distributed to anyone who wishes to view it. The information distribution mechanism is implemented by using the Observer-Observable pattern, and the sensors are the information providers.

The information consumers (the observers) are components called MonitoringScreen and Log which present the information they receive from the weather station. MonitoringScreen presents information from the temperature and pressure. Log presents the information on pressure and pressure trend. These components have designated functions for presenting the information. Printing the information will be done in the following format:

<MonitoringScreen/Log>: <temperature/pressure/pressure trend> = <data/RISING/FALLING/STABLE> <Celsius/millibars>

Examples:

MonitoringScreen: temperature = 30 Celsius

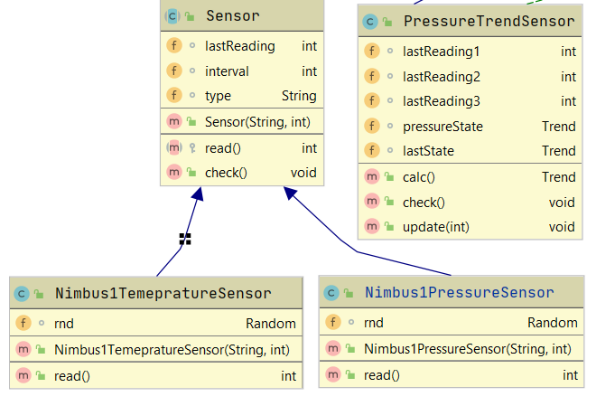
Log: pressure = 1010 millibars

The measurements are scheduled by a clock mechanism (the code for the clock is provided, make sure that the sensors are registered to it). Each sensor determines its interval between measurements, when it is registered to a clock (the temperature sensor should be registered with an interval of 700 and the air pressure should have an interval of 1100). When a sensor is registered to the clock with an interval, it also provides the clock with an AlarmListener object which contains a wakeup function in order to activate its measurement.

Activating the clock is implemented by calling the ‘check’ function which calls the ‘read’ function. the information from the reading is distributed to all observers only if there is a change in value from a previous measurement (as was learned in class).

The Nimbus sensors for pressure and temperature contain a joint behavior and so they should inherit from the Sensor class which contains the TemplateMethod, activating the abstract *read* method.

We provide the following partial UML diagram to assist you:



Note! When grading the exercise, we will replace the Nimbus classes with the grading version. Therefore, it is important that they implement the diagram precisely (it is especially essential that they contain a *read* function).

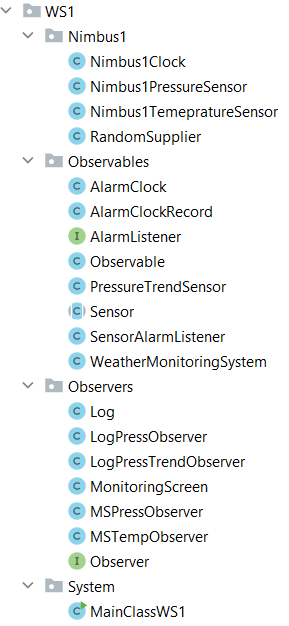
The weather station component is a singleton class. This component provides functions which allow the observers to register to the temperature, pressure and pressure trend components.

The following messages should be part of the output:

1. Creating objects, in the format: <object name> was created
2. Registering as an observer to a weather station component, in the format: <object name> observes <sensor name>
3. Registering to a clock,in the format: <object name> registered to clock
4. Printing the values read by the observers. The printing format is defined above.

For the print order to be deterministic, the temperature sensor is registered to the clock first, followed by the air pressure sensor. In addition, the order of observer registration to the pressure sensor is: The pressure trend sensor goes first, followed by the MonitoringScreen and then the Log. All variables of type ‘random’ will be copies of the object managed by the RandomSupplier class (which is provided to you).

The supplied classes assume the following packages structure (in the submitted code there should not be any package structure due to the checker limitations):



**Assistance material:**

The weather station tutorial found in Moodle, pages 5-12.

Course presentations: weatherStation1

In addition, you are provided the code of Nimbus2Clock, Alarmclock (partial, you should complete the missing parts), RandomSupplier and the main class.