COMP1161 Semester 2, 2018/19

Project Assignment #2 – The Storm Tracker Application

A. PROJECT OVERVIEW

In Assignment #1 you wrote classes for objects that can be used on a map for tracking storms. This assignment will require that you write a class that manages a collection of objects including countries and storms and integrate the new class with a text-based user interface. Assignment #3 will require that you integrate the application with a graphical user interface. The exercise is intended to illustrate an important OOP concept – separation of concerns. Each class in a properly designed program should have its own responsibilities. The object that manages the map objects is not responsible for interacting with the user. Instead, this responsibility is held by class(es) that are constitute the user interface to the program. This principle allows us to change the user interface without making changes to the application itself. Hopefully, you will learn how to build a software application using OOP while having fun doing so 😊.

B. THE DESIGN

Figure 1 is a class diagram for the Storm Tracker application. The design features a class (TextUI) that is responsible for providing the user interface to the application and other classes that implement the functionalities of the storm tracker.

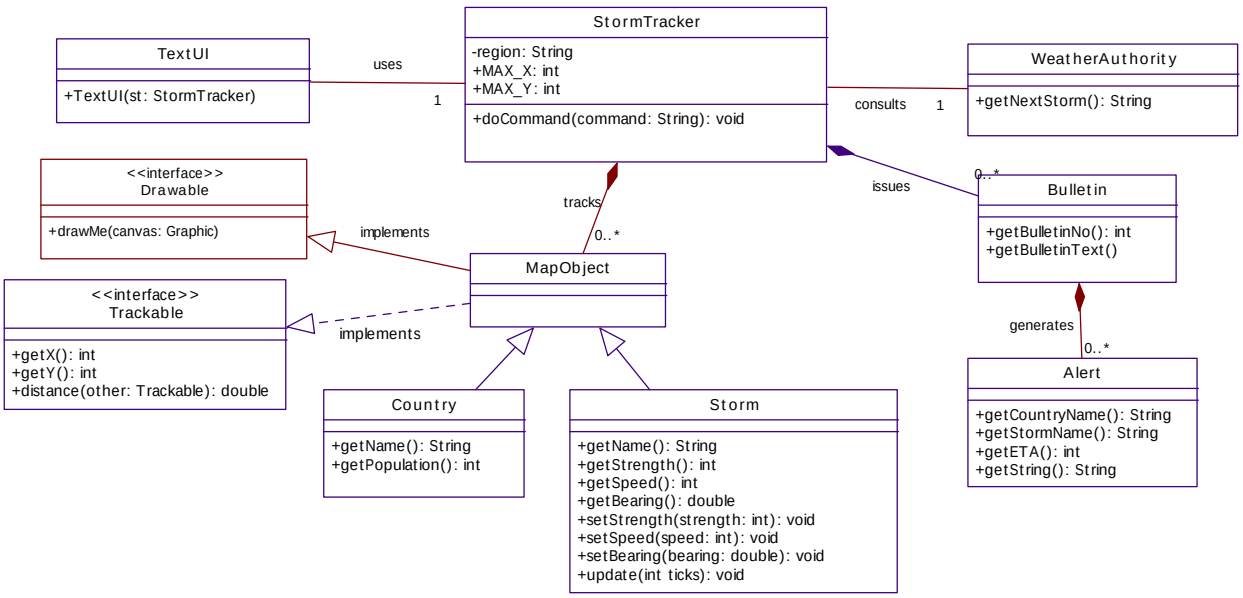

Figure 1: Class Diagram for the Storm Tracker

The following is a description of the classes that are used in the design:

TextUI	This is a text-based User Interface that accepts commands from the user and passes them on to a Storm Tracker object.
StormTracker	This object accepts commands from the user interface and updates the sttus of each storm in response to each command.

do(command:String) accepts a command from the user interface, carries our the requested operation and returns a string as the response. The types of commands that are expected are described later on.

WeatherAuthority — is a special class that keeps a list of storms for each storm season, and provides a unique name for each new storm on request.

getNextStorm() returns the next name for a storm.

Bulletin — a bulletin is a message containing details of all active storms. At the least a bulletin has the name of each storm, its position, strength, bearing, speed.

getBulletinNo() returns the bulletin number.

getBulletinText() returns the messages contained in a bulletin.

Alert — An alert is created for each country that is in the path of an approaching storm if the storm is expected to hit the country within 24 ticks.

getCountry() get the name of the country that a storm is approaching.

getStorm() get the name of the storm that is threatening the country.

getETA() get estimated number of ticks till the storm hits the country.

C. SYSTEM FUNCTIONS

1. On starting up, the application shall prompt the user to enter the name of a file that contains the name of a region, and data about countries in the region.
2. The program shall read the data from the file and use the data to create a StormTracker object.
3. If there is a file with data from a previous simulation then the program shall read the data from that file, and restore the simulation to the status that was saved in the file.
4. The program shall then prompt the user to enter commands (run the simulation). Valid commands are listed in Table 1. These are basic commands. Please feel free to add your own commands

Table 1: A List of Basic Commands for the Storm Tracker

Command	Expected Response
STORM *x y*	Adds a new storm to the map at the given coordinates. For example, the command NEW STORM 30 43 tells the storm tracker that a new storm is to be added at row 43 column 30. The storm tracker should consult with the WeatherAuthority object to get the name of the new storm and shall add the storm to the collection of storm objects.
UPGRADE <name of storm>	The storm tracker shall send an upgrade command to the named storm. The storm shall either increase strength, speed, or change bearing. You can decide what is done when perhaps based on some random value (see the Random class). The string returned shall contain the new status information for the storm.
DOWNGRADE <name of storm>	The storm tracker shall send a downgrade command to the named storm. The storm shall either decrease strength, or

Command	Expected Response
	decrease tracking speed. You can decide what is done when perhaps based on some random value (see the Random class). The string returned shall contain the new status information for the storm.
TICK	The StormTracker object shall run one time period in which the status of each storm is updated based on its current position, direction and strength. A bulletin is created during each tick. Each bulletin contains data about all active storms. If necessary, an alert is created for each country that is threatened by a storm. At the end of the tick the StormTracker shall return a string containing the bulletin and any alerts.
STOP	This command shall stop the simulation and quit the program. If the user wishes to save the simulation then the program shall create a file with all status so that the simulation can be restarted when the program in run again.

D. DESIGN NOTES

1. A simple worksheet with the countries of the Caribbean is provided. You can use the information from this worksheet to create the file of countries that is loaded at atartup.
2. In assignment #1 you created constants for MAX_X, and MAX_Y as static members of the Storm class. These values are now kept by the StormTracker class. You will need to make modifications (without removing the constants from the Storm class) so that the position of a new storm can be set by the StormTracker object.
3. The main method of your program should not be responsible for any logic of the application. In other words, the main method shall be “lean”. Its only task is to create the user interface, and storm tracker objects. This is an important object-oriented design principle.
4. Another important OO design criterion is that each class/object should have a single responsibility. For example, the Storm Tracker object should not try to interact directly wijth the user. That is the responsibility of the user interface. Instead, the Storm Object will send messages to the user interface object which will then decide how to use the message. The importance of this criterion will become clearer when you decide to use a GUI instead of a text-based user interface.

E. SUBMISSION INSTRUCTIONS

1. This assignment may be attempted individually, or by a pair of students.
2. Assignment #2 and Assignment #3 will be marked together by demonstration at the end of the semester. The deadline for submission will be announced then.