HW5 – Inheritance and Aggregation

Estimated time: 10-14 hours

Learning Objectives

- Gain experience with localizing design decisions using inheritance
- Gain experience with polymorphism
- Gain experience with reuse via inheritance
- Gain experience with localizing design decisions using aggregation
- Gain more experience with encapsulation in the presents of aggregation
- Gain experience with reuse via aggregation
- Gain more experience with communicating design using UML Class Diagrams
- Gain more experience with simple testing techniques
- Gain more experience with debugging skills

Overview

In this homework assignment, you will complete a program, called *GeoRegions*, that manages a hierarchy of geographical regions. The regions can be the world, nations, states, county, and cities. For this purpose of this assignment, the world includes nations; nations include states; states include counties or cities that are not in a county; and counties can include cities. Every geographical region must have a name, a population, and area (measured in square kilometers). Also, to help the user select regions in the user interface, every region has a unique identifier.

To complete this assignment successfully, you understand and use both inheritance. Most of the design and implementation is already done, but you need to understand those parts and implementing the incomplete parts. Regardless of the additions or changes that you make, the program needs exhibits good localization of design decision, encapsulation, abstraction, and it needs to use inheritance and aggregation in appropriate ways.

Requirements

Below are the functional requirements for your

1. When *GeoRegions* starts up, it should automatically load “Nations.txt”, if it exists, into a set of *Nations*.
    a. If “nations.csv” doesn’t exist or is empty, then *GeoRegions* should start off with an empty World region.
    b. Before the program exits, it should save all the world region to “Nations.txt”, such that every region is a line of text, formatted as

        Region type,Name,Population,Area

following by its subregions and then a "^^^" on its own line. See the files in Testing/SampleData for examples.

2. *GeoRegions* should provide the user with an interface for manipulating the regions and their sub-regions.
    a. *GeoRegion* should show a menu that contains options applicable the current context, which must be one the following: the world, a nation, a state/district, a county, or city.
    b. *GeoRegion* should always show the current context before the menu.

3. When the current context is the world, the menu needs to include the following options:
    a. Create new nation, which is a top-level region.
    b. List all nations without that the sub-regions that they contain
    c. Edit a nation
    d. Delete a nation
    e. Print a full list report of the region an outline format that shows each region or sub-region's name, population, area, and persons/km^2.
    f. Change contexts to a nation
    g. Quit the program

4. When the current context is a nation, state or county, the menu needs to include the following options
    a. Create a new sub-region.
        i. When then current context is a nation, the valid sub-regions are states.
        ii. When the current context is a state/district, the valid sub-regions are counties or cities.
        iii. When the current context is a county, the valid sub-regions are cities.
    b. List the sub-regions (not including those sub-regions' sub-regions)
    c. Edit a sub-region.
        i. If the user selects this option, *GeoRegions* should 1) ask for the identifier of the sub-region, 2) make it the new context, and 3) present the menu for that context.
    d. Print a full list report of the region an outline format that shows each region or sub-region's name, population, area, and persons/km^2.
    e. Change contexts to a sub-region, except when in the context of a county
    f. Back up to previous context and menu

5. When creating a region, *GeoRegion* should gather the following data from the user and then create a new region and add it to the current context.
    a. Name of the region
    b. Population, which is the number people living in the region not counting those living sub-regions

c. Area in km^2

6. When editing a region, GeoRegion should display the region's name, population, and area and allow the user to change those values.

7. GeoRegions should be able to compute a region's total population, which it's population plus the sum of the total populations for all of its sub-regions.

8. GeoRegions should allow the user to print all out a full report, outline format, of any context. For each region in the context, the report needs to show the region's name, total population, area, and persons/km^2.

Instructions

For HW5, you must complete the following

1. Use Git to pull down the latest copy the cs1440s17-shared repository.

2. Find the *GeoRegions* folder in the cs1440s17 folder and copy that folder to your own cs1440 repository.

3. Study the initial design and CLion project. There is a significant amount of existing code. Walk through every part in the debugger to make sure you understand what it is doing.

4. Study the *CMakeLists.txt* file to understand how it specifies two targets: *GeoRegions* and *Test*. If you add new source files, you will need edit this file.

5. Complete the design so it describes the special kinds of regions and how regions are going to contain sub-regions.

6. Look for all occurrence of TODO comments in the code and implement those snippets of code.

7. Finish the incomplete test cases for Region (see TODO comments)

8. Finally, push all your materials to your Git repository, by doing the following

    a. Open a terminal window or command prompt

    b. Change your working directory to your repo directory

        $ cd <your repo directory>

c. Add all files to your repo

```
git add .
```

d. Commit a new version, with the message “Final version of HW5”

```
git commit –m “Final version of HW5”
```

e. Push to your remote repository

```
git push
```

9. Submit the URL for your Git repository to Canvas

Grading Criteria

Requirements	**Max Points**
Proper setup and submission of HW4 using Git	5
A complete design (documented with a UML Class Diagram) that exhibits good localization of design decisions, encapsulation, abstraction, inheritance, and aggregation	15
An implementation that is correct with respect to the design	10
Meaningful test cases for the Region class	15
The GeoRegions target runs as expected according to the requirements	5

Penalties	**Deduction**
Incomplete or inaccessible Git repository	0 to -50
Doesn’t compile	0 to -20
Poor code readability	0 to -10
Warnings in the code that could have been easily corrected	0 to -10
Weak parameter or method declarations (things could have be declare as “const” were not)	0 to -5