Sarah MacEwan

Sienna Collette

COMP 3721

Lightning Week Assignment 1

Design Report

We decided to use the Observer design pattern for our implementation of this problem. This seemed like the best choice, since there were objects (Mirelle and Ursule) that wanted updates about the state of other objects (blossoming trees), but not all the time (only while they were on vacation, presumably). The Observer design pattern is a good choice for handling problems like this because it ensures that the objects that want observations about the states of other objects do not know too much (are not tightly coupled to) the objects they are observing, and likewise for the subjects of observation.

In our implementation, we included a TreeBlossomReport class that worked as a go-between for the Observer and Subject (in our case, FloweringTrees) classes. The TreeBlossomReport could query information about the FloweringTrees and could keep track of which observers were “subscribed” to get updates about the trees. Then, whenever a tree changed from blooming to not blooming, notifySubscribers() was called on the TreeBlossomReport, which would send the Observer the updated list of trees that were in bloom. We used an abstract class, TreeBlossomObserver, to define the behavior of the observer so we could make subclasses if different types of observers needed to use the information in different ways.