Lab 4: Structural Patterns

1 Introduction

As we've discussed in class, design patterns come in a few varieties including Structural, Behavioural and Creational. This week we'll be working with a Structural pattern named the Decorator pattern.

To begin, log into MarkUs and add the started files for "Lab 4" to your repository; then pull them to your local machine. In the repository you should find these java files: *MunicipalTree.java, WhiteAsh.java, Beech.java, TreeDecorator.java, BarkDecorator.java, FoliageDecorator.java and Main.java.* Some sanity tests have also been included to help you get started.

2 Programming Task

This week, your starter project once again contains code inspired by urban forests. It contains classes for two types of tree, both of which implement the MunicipalTree abstract class: Beech and White Ashes. In this lab, we will revisit our calculations of the carbon content within these trees.

As you may remember, the carbon content of trees can be calculated using allometric equations, which can take the following form:

$$B = a * (p * diameter^q)$$

where a is the mean wood carbon fraction for tree species' surveyed by the USDA Forest Service Forest Inventory and Analysis. The variables p and q are called allometric coefficeints, and they are used to relate the *diameter* of tree parts to biomass.

What we did not discuss in our last lab is the fact that different parts of each tree (i.e. the wood, the branches, the leaves and the bark) have different allometric coefficeints. Coefficients for different parts of North American trees can be found here:

https://www.researchgate.net/publication/
249535320_Canadian_national_tree_aboveground_biomass_equations

In our lab this week, we will use the Decorator pattern to "decorate" estimates of the carbon that lies in the wood of trees with estimates of the carbon in trees' leaves and bark. You will notice that the MunicipalTree class contains a default implementation of *calculateCarbonContent* that calculates carbon based

on the allometric coefficients of wood. We want to enhance these calculations with calculations of carbon in leaves and bark. But we do not want to change our original classes!

To do this you will write:

1. An abstract class called "TreeDecorator", which we will use to "decorate" the *calculateCarbonContent* method of Municipal Trees.
2. Two concrete TreeDecorators: a FoliageDecorator and a BarkDecorator.
3. A municipal tree that is "decorated" with a FoliageDecorator will override the tree's *calculateCarbonContent* method. The "decorated" method should add carbon content stored in the tree's leaves to the total calculated by the overriden method.
4. A municipal tree that is "decorated" with a BarkDecorator will override the tree's *calculateCarbonContent* method. The "decorated" method should add carbon content stored in the tree's bark to the total calculated by the overriden method.
5. It should be possible to decorate a FoliageDecorator with a BarkDecorator and vice versa.
6. Make sure that your "decorations" do not change any existing classes!

3 What to Submit

TreeDecorators.java, FoliageDecorator.java and BarkDecorator.java

HAVE FUN AND GOOD LUCK!