Robert Rose

12/8/2017

Object Oriented Programming

Final Project Report

For my final project, I decided to use an abstract factory to create Car objects, from two different companies, Ford and Audi. Then, after creating them and putting them in outlets, I test them with a Façade object named Inspector, which utilizes three different classes to test the cars to see if they will pass an inspection. The inspection rules state that each Car must have a tire pressure (tireP) within the range one and ten, anything less than one and greater than ten failing the inspection. In addition, the software must be installed and the oil must be changed on each car (softwareInst and oilChanged respectively), and if either of those Boolean values are false then the car fails the inspection. All of these field must be true in order to pass.

The way the Inspector works, is that it uses three other classes as sort of “checks.” These classes are tPressure, SoftwareCheck, and OilChecker, and they check the pressure, if the software is installed, and oil respectively with the check function. The check function returns a Boolean if the inspection rules follow true for each thing the classes test. Inspector itself calls a function called inspect, which calls each check function of the smaller classes. If each check returns true, then inspect returns true.

The factory itself that returns the cars is an abstract, generic class called Factory. It basically lays the groundwork for each car Factory class, and I specifically made it generic just in case I wanted to use it again for some other object. This abstract Factory class has an abstract method called get. The objects that extend the Factory are Ford and Audi, and each of them call the get method. When audi calls the get method, it returns a Car subclass called AudiR, which is created with a random tire pressure between -2 and 8. This ensures that there is some chance of failure, because not all cars that come from factories actually pass inspection. The Ford factory subclass also returns a similar car subclass called FordTruck, but it also can return a broken car called FordBroken. The FordBroken car specifically designed to fail the inspection, having a tireP range of negative ten to zero, and both oilChanged and softwareInst values are set to false. This was made to test Inspector and make sure it worked properly.

These Factory subclasses are then used to create a generic class called Outlet (Outlet being generic for the same reason I made Factory generic). The Outlet uses a Factory to create a car, which is stored in the Thing variable. This variable is returned when we want to check to see if the class passes inspection.

The finalDriver class is only the main method, and this is where we test the OOM. First we declare and initialize all the Factory subclasses and Outlets, the different Cars, and then the inspector class. Then, we print out the rules so the user understands what passes and does not pass. After, we print the result of fordOutlet.get() while simultaneously storing it in fordCar, then we test fordCar. If it passes, we print that the car passes, else we print it failed the inspection. Then we repeat the same steps for the Audi variables, completing the OOM test.