Aidan Duffy

MET CS 665

Boston University

Assignment 3, Task 1: Description

Flexibility

In terms of flexibility, it is my goal to ensure that this program is as flexible as possible. Therefore, the program will utilize a factory design pattern. The factory will be for the customers. All of the common code among all of the customers will be placed in a separate Customers abstract class, and the code specific to the individual types of customers will be placed in subclasses that represent these types (ie Newbie.java or Business.java). New customer types can easily be added in this way.

Simplicity and Understandability

I wanted to ensure that the code was modular, was properly marked, well named, and commented so that it was not cluttered, easy for developers to read, and simple enough to understand. All of the classes will be generalized enough so that new objects can easily be created. All of the code, as mentioned above, will be placed in either the common interface or the specialized subclasses so errors can easily be found.

Duplication Avoidance

Given that all of the major objects will have entirely separate classes, and they will have little to no overlap, then there will be little to no opportunity for duplications.

Design Patterns

I utilized the factory design pattern as that made the most logical sense. When a company creates a new message request, it will sendEmail(), learn the context of the message by entering the factory and calling getEmail() on all of the subclasses in the factory, and populate the contents of an Email object. This will all be run by the EmailGenerationSystem. Additionally, for the system, I used a singleton so that there can only be one instance to limit confusion and duplication.

General Overview

Currently, the program operates from a simple factory design. A set of companies exist as well as a subset of their customers, each of various types. The EmaiLGenerationSystem runs the setup, creates a company, who sends messages based on their client types. I added a rudimentary (boolean based, no cryptography) encryption option for business emails.

Running the Program

Simply run the relevant maven commands on the Main.java in the edu.bu.met.cs665 package.