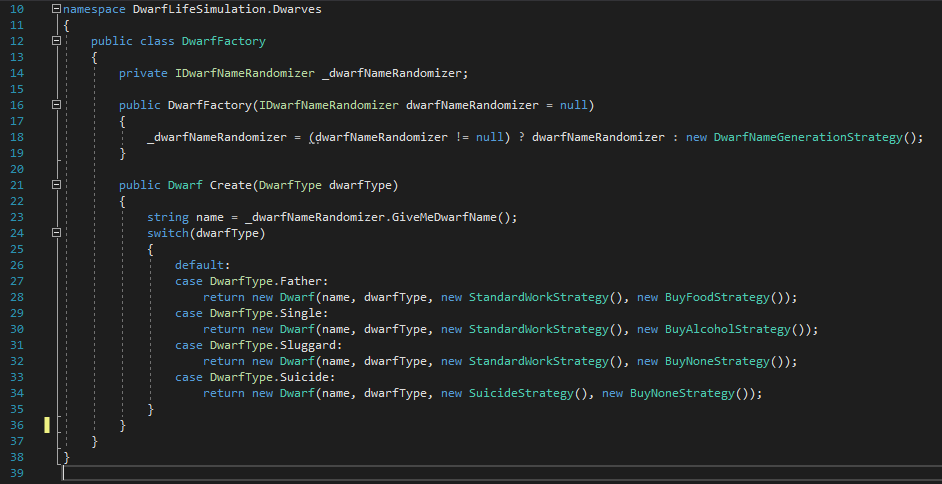
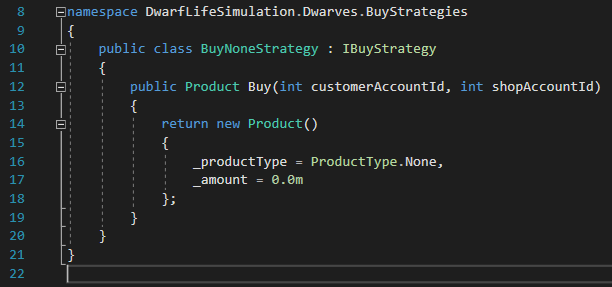
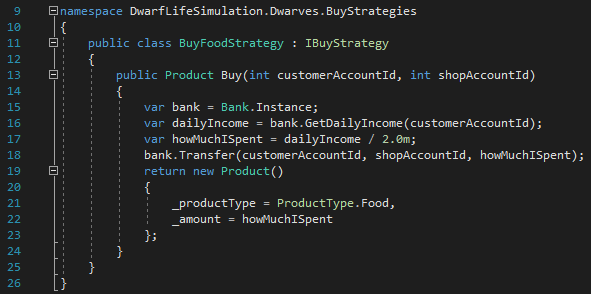
Paweł Howaniec

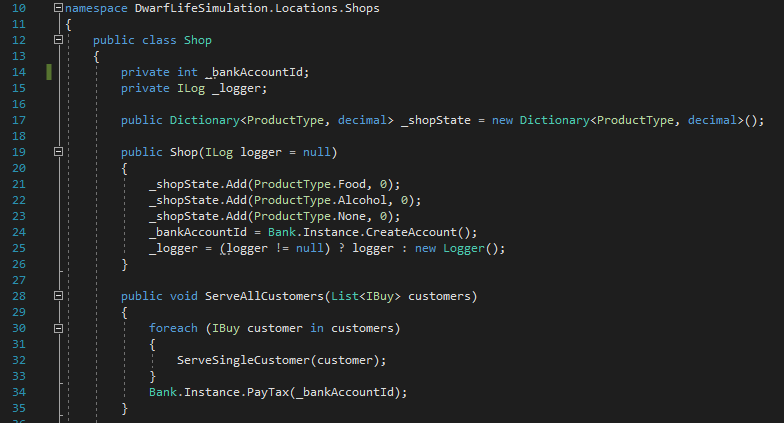
Wybrane fragmenty kodu są częścią aplikacji – symulacji górniczego miasta krasnoludów.

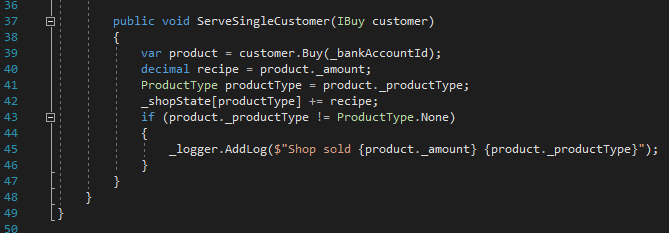


Pierwszą klasą wyróżnioną jest DwarfFactory, użyłem tutaj wzorca fabryki a jedynym celem tej klasy jest tworzenie krasnoludów według wybranego typu krasnoluda (enum DwarfType). Istotne dla mnie było aby w zależności od wybranego typu zwracany był odpowiednio stworzony obiekt krasnoluda z losowanym imieniem z puli imion oraz z poszczególnymi strategiami kupowania oraz pracy według narzuconej domeny. Użyłem factory, aby kod był przejrzysty oraz łatwo rozszerzalny.



Następnie dwie przykładowe klasy/strategie kupowania, które implementują interfejs IBuyStrategy, istotne dla mnie było aby to krasnoludy wiedzą co oraz ile czego kupują, według domeny krasnolud typu Father kupuje Food, Single kupuje Alcohol, a krasnoludy typu Sluggard oraz Suicide nie kupują nic. Metoda Buy zwraca obiekt typu Product z dwoma polami \_productType oraz \_amout aby wiedzieć ile i jakiego produktu zakupił krasnolud, wykonywany jest w tej metodzie „przelew” z konta krasnoluda na konto sklepu w wysokości połowy dziennego urobku krasnoluda. Po analizie tego kodu, inaczej bym to teraz napisał, gdyż jest to strategia, która powinna być bezstanowa a używam w niej zmiennych prywatnych i teraz bym zrobił to na zasadzie input -> transform -> output.





Ostatnią klasą, powiązaną z poprzednimi, którą chciałbym pokazać jest klasa Shop.

Chciałem aby sklep przyjmował tylko obiekty implementujące interfejs IBuy, a nie tylko obiekty typu Dwarf, stąd też pomysł na strategie, aby nie miał dostępu do niczego poza tym co implementuje interfejs IBuy, jak również dzięki temu zabiegowi aplikacja jest łatwo rozszerzalna. Sklep posiada Dictionary przechowujący ilość sprzedanych towarów danego typu, posiada loggera do wyświetlania na bieżąco co i za ile zostało sprzedane. Posiada metodę do „obsługi” pojedynczego klienta typu IBuy oraz metodę do „obsługi” wszystkich klientów z podanej listy, na koniec zgodnie z domeną, płacąc podatek do banku.

A na koniec całkowity test sklepu, sprawdzający całość logiki i działania strategii kupowania oraz sklepu.

