PZWP – Zasady SOLID

Zadanie 1

1. Stworzyć projekt w Eclipse i zaimportować klasy z pliku restauracja.zip.
2. Przeanalizować i uruchomić kod.
3. Jakich zasad nie spełnia podany kod?
4. Żeby to zobaczyć dokonaj w nim następujących modyfikacji:
    a. Dodać możliwość podania zupy z ryżem zamiast makaronu.
    b. Dodać możliwość wysyłania obiadu pod wskazany adres (na wynos) zamiast podawania do stolika.
    c. Dodać możliwość gotowania nowej zupy pomidorowej, w której zamiast kostki rosołowej używany jest koncentrat pomidorowy.
    d. Napisać nowy kod testujący (metody przygotujZupePomidorowaZRyzemNaWynos() i przygotujZupePomidorowaZMakaronem() w klasie Test)
5. Które klasy i jak należało zmienić? Jakich zasad nie spełnia klasa Zupa?

6. Zaproponować inną implementację problemu pozwalająca na łatwiejszą rozbudowę programu i uwzględniającą zasady Open Close, Dependency Inversion, Single Responsibility. *(Wskazówka: zupę przygotowuje Kucharz a podaje Kelner, zupa ma tylko nazwę i składniki)*

7. Sprawdzić czy zaproponowane rozwiązanie można rzeczywiście łatwo rozbudowywać zgodnie z punktami 4a-4d. Tym razem zasady powinny być spełnione.

8. Stworzyć w klasie test metodę przygotuj(Kucharz, Kelner), która tworzy zupę, dodaje do niej składniki, przekazuje kucharzowi do przygotowania i kelnerowi do podania. Aby kod działał dla dowolnych kucharzy i kelnerów, konieczne jest stworzenie odpowiednich interfejsów (np. IKelner i IKucharz).

9. Uzupełnić interfejs IKelner o metody: nakryjStol(), zbierzNaczynia() i uruchamiaj te metody w metodzie przygotuj(). Co trzeba zrobić z klasą Kurier?

10. Zaproponuj rozwiązanie zgodne z zasadą Interface Segregation.

11. Kurier, aby móc zawieźć zupę musi znać adres. Dodać do niego metodę setAdres() i czytanie adresu w metodzie podaj().

12. Jakiej zasady nie spełnia teraz kod i dlaczego? Jak to poprawić?

PZWP – Zasady SOLID

Zadanie 2

1. Zaprojektować aplikację dla firmy kurierskiej. Mamy kilka miejscowości w których znajdują się magazyny paczek. Operator aplikacji może wykonać następujące operacje:
    a. Nadanie paczki – nowa paczka umieszczana jest w magazynie jednej z miejscowości z informacją o adresacie.
    b. Przewóz paczek z jednej miejscowości do drugiej – przewozy dokonywane są samochodami, samochód znajdujący się aktualnie w miejscowości jest ładowany paczkami (każdy samochód może zmieścić tylko określoną liczbę paczek) i przemieszcza się do innej miejscowości.
    c. Odbiór paczki – paczka jest wydobywana z magazynu w określonej miejscowości.

   Zaproponować konieczne klasy, metody, interfejsy tak, aby dało się zrealizować podaną funkcjonalność i aby kod zgodny był z zasadami SOLID, a więc odporny na zmiany i rozbudowę, które mogą nastąpić.

2. Zapoznać się i ocenić kod podany w pakiecie kurier1.zip
3. Dokonać następujących zmian:
    a. Każda paczka powinna mieć ciężar i na tej podstawie powinna być liczona ładowność samochodu.
    b. Należy się zabezpieczyć przed nieprawidłowymi wartościami w metodzie przewieź – gdy samochód nie jest w punkcie startowym lub paczki nie ma w punkcie startowym.
    c. Jest nowy typ samochodów – otwarte, które nie mogą przewozić paczek żywnościowych.
    d. Pojemność magazynów w miejscowościach jest ograniczona.

4. Zapoznać się i ocenić kod podany w pakiecie kurier2.zip
5. Spróbować dokonać tych samych zmian, co wcześniej dla kurier1 i opisać, co trzeba było zmienić.
6. Porównać oba rozwiązania.