Przetwarzanie zamówień

> Autorzy: TS + PW

Wstęp

Celem jest przygotowanie systemu przetwarzania płatności, zawierajcego dwa moduły: `Payments` (płatności) i `Shipment` (dostawa). Oczywiście w praktyce cały kod znajduje się w jednym projekcie, ale podejdź do jego implementacji tak, jakby poszczególne moduły były oddzielnie opracowywane przez odrębne zespoły.

Przetwarzanie zamówień

Dane dotyczące zamówień są przechowywane w dwóch odrębnych bazach danych `LocalOrdersDB` i `GlobalOrdersDB`. Potencjalne zmiany w ich strukturze nie powinny w żaden sposób wpływać na "klienta" (`Program.cs`). Ponadto, system powinien przetwarzać wszystkie zamówienia razem, niezależnie od docelowego kraju. Zamówienie (`Order`) zawiera informację o wybranych metodach płatności: `SelectedPayments`. Z każdą płatnością (`Payment`) jest związana konkretna metoda płatności oraz maksymalna kwota, jaka może zostać zapłacona z jej użyciem. System ma wspierać trzy metody płatności: `PayPal`, `Invoice`, `CreditCard`. Musi jednak być zaprojektowany tak, aby w przyszłości łatwo można było dodać obsługę kolejnych (dodanie nowej metody płatności nie powinno wymagać modyfikacji istniejących). Przetwarzanie płatności polega na próbie zapłaty maksymalnej dopuszczalnej kwoty kolejno dla każdej metody płatności związanej z zamówieniem, dopóki nie zostanie osiągnięta łączna kwota zamówienia. Kolejność metod płatności jest zawsze taka sama: `PayPal` -> `Invoice` -> `CreditCard`.

Próba płatności daną metodą może zakończyć się niepowodzeniem, zaimplementowanym w następujący sposób:

- Co trzecia płatność `Invoice` kończy się niepowodzeniem.
- Każda płatność `PayPal` ma 30% szans na niepowodzenie (`Random` zainicjalizowany ziarnem `1234`).
- Płatności `CreditCard` zawsze kończą się powodzeniem.

W przypadku niepowodzenia, metoda płatności jest pomijana i przetwarzanie przechodzi do kolejnej. Pomyślnie zakończone płatności powinny być dodane do `FinalizedPayments` wraz z zapłaconą kwotą.

Ponadto, w trakcie przetwarzania zamówienia należy odpowiednio ustawić jego `Status`:

- `WaitingForPayment` gdy jest nieopłacone,
- `PaymentProcessing` gdy jest częściowo opłacone,
- `ReadyForShipment` gdy jest opłacone i może zostać dostarczone (trafić do modułu `Shipment`)

Wymagania dot. modułu Payments

- Dodanie nowych metod płatności powinno być możliwe bez zmian istniejących komponentów `PaymentMethods` components, za wyjątkiem enum-a `PaymentMethod`.
- Zmiana jednej metody płatności nie powinna pociągać za sobą jakichkolwiek modyfikacji pozostałych.
- Jeżeli łączna kwota zamówienia została już zapłacona, nie przechodzimy już do kolejnych metod płatności.
- Należy użyć komponentów dostarczonych w zadaniu.

Dostarczanie zamówień

Dodaj filtr, aby uwzględnić tylko opłacone zamówienia (o statusie `ReadyForShipment`) i zaimplementuj dla nich proces dostarczania przesyłek. Wypisz etykiety dla każdego zamówienia i dla każdej przesyłki złożonej z zamówień. System ma wspierać dwóch dostawców: `LocalPost` i `Global`. Musi jednak być zaprojektowany tak, aby w przyszłości łatwo można było dodać obsługę kolejnych (dodanie nowego dostawcy nie powinno wymagać modyfikacji istniejących).

Wybór dostawcy (`ShipmentProvider`):

- `LocalPost` obsługuje wszystkie przesyłki adresowane do Polski.
- `Global` obsługuje wszystkie przesyłki zagraniczne.

Obliczanie podatku:

- Baza danych `TaxRatesDB` zawiera informacje o stawkach podatku VAT w poszczególnych krajach.
- Na potrzeby zadania zakładamy, że jest dostarczana przez zewnętrzną firmę, w związku z czym jej struktura nie jest znana klasie ShipmentProvider (i w przyszłości może się zmienić).
- Podatek jest obliczany jako X procent z `PaidAmount`, gdzie X jest stawką VAT dla danego kraju pobraną z bazy danych.

Etykiety i paczkowanie:

- Poszczególni dostawcy (`ShipmentProvider`) mają własną logikę generowania etykiet oraz w różny sposób łączą zamówienia w przesyłki.
- `LocalPost`: jedna przesyłka (`IParcel`) dla wszystkich zamówień.
- `Global`: oddzielne przesyłki dla każdego kraju (`Recipient.Country`).

Najpierw zarejestruj u właściwego dostawcy wszystkie opłacone i gotowe do wysyłki zamówienia. Dla każdego z nich wypisz etykietę odpowiednią dla dostawcy (`Printer.PrintLabel`). Skorzystaj z interfejsu `ILabelFormatter`, aby zapewnić możliwość ponownego wykorzystania logiki dla wypisywania etykiet w przyszłości. Etykiety dla dostawcy `LocalPost` nie podają docelowego kraju, na przykład:

```
Shipment provider: LocalPost
Janina Osiwiecka
Kamionka 3/34
Lipsko
25-895
```

Etykiety dla dostawcy `Global` uwzględniają kraj adresata, na przykład:

```
Shipment provider: Global
Lea Mathias
9054 Share-wood
Manhattan
90561
USA
```

Następnie, wypisz podsumowanie dla każdej przesyłki (`IParcel`) utworzonej przez dostawców w poprzednim kroku. Preferowany format:

```
Shipment provider: Global
TotalPrice:     246,00, TotalTax:       0,00, TotalPriceWithTax:     246,00
-------------------------------------------------------
------------------------Hawaii-------------------------
 OrderId          Amount             Tax   AmountWithTax
       1          246,00            0,00          246,00
TOTALS:           246,00                          246,00

-------------------------------------------------------
```

Wymagania dot. modułu Shipments

- Dostawcy przesyłek są "ciężkimi" obiektami, więc zastosuj leniwą inicjalizację: dopiero wówczas, gdy pojawi się pierwsze zamówienie do zarejestrowania u danego dostawcy.
- Do wypisania podsumowania przesyłek zastosuj dostarczony interfejs `ISummaryFormatter` wraz z domyślną implementacją `SummaryFormatter`.
- Dodanie nowych dostawców (np. dla poszczególnych krajów) nie powinno wymagać żadnych zmian w "kliencie", czyli w `Program.cs`.
- Dodanie nowych dostawców nie powinno też wiązać się z żadnymi zmianami już istniejących.
- Zmiana struktury bazy stawek podatku VAT nie powinno wymagać żadnych zmian logiki w obrębie modułu Shipments.

Uwagi

- Obydwa moduły powinny wypisywać informacje na konsolę (przykład w pliku `output.txt`).