Promocje dla metod płatności

Płatności

W naszym internetowym supermarkecie oferujemy możliwość płatności za zamówienia przy użyciu tradycyjnych metod (np. kart płatniczych, szybkich przelewów obsługiwanych przez banki, itp.) oraz punktów lojalnościowych. Każde zamówienie można opłacić:

- w całości za pomocą jednej tradycyjnej metody płatności (np. jednej karty),
- w całości punktami lojalnościowymi,
- częściowo punktami i częściowo jedną tradycyjną metodą płatności.

Image by freepik

Rabaty

Na podstawie umów z wybranymi bankami oferujemy rabaty, jeśli zamówienie zostanie opłacone **w całości** kartą konkretnego banku. Każde zamówienie ma przypisany podzbiór promocji, odpowiadających metodom płatności, które mogą zostać zaaplikowane **tylko wtedy**, gdy płatność zostanie dokonana odpowiednią metodą (np. konkretną kartą). System **nie wspiera** stosowania tych rabatów przy częściowych płatnościach kartą.

Dodatkowo, aby zachęcić klientów do korzystania z punktów lojalnościowych, oferujemy rabaty za częściową płatność punktami lojalnościowymi. Promocja za użycie punktów nie jest przypisana do zamówień, ponieważ można ją zastosować zawsze. W przypadku zastosowania, wyklucza możliwość aplikacji dodatkowego rabatu przez użycie karty. Podsumowanie zasad:

1. Do każdego zamówienia przypisany jest wybrany podzbiór możliwych do zaaplikowania promocji związanych z metodą płatności.
2. Jeśli całe zamówienie zostanie opłacone kartą banku, z którym mamy podpisaną umowę, naliczany jest rabat procentowy określony w definicji danej metody płatności.
3. Jeśli klient opłaci **co najmniej 10%** wartości zamówienia (przed rabatem) punktami lojalnościowymi, sklep nalicza dodatkowy rabat w wysokości **10% na całe zamówienie.**
4. Jeśli całe zamówienie zostanie opłacone punktami lojalnościowymi, należy zastosować rabat zdefiniowany dla metody “PUNKTY”, zamiast rabatu 10% za częściową płatność punktami.

Cel zadania

Celem zadania jest opracowanie algorytmu, który dla zadanej listy zamówień, dostępnych promocji oraz portfela klienta (zawierającego metody płatności, limity oraz rabaty) dobierze **optymalny sposób płatności dla każdego zamówienia**, maksymalizując łączny rabat przy spełnieniu wszystkich ograniczeń. Należy tak dobrać płatności, aby wszystkie zamówienia były w pełni opłacone. Algorytm powinien **minimalizować płatności kartami**, preferując wydawanie punktów, jeśli nie zmniejszy to należnego rabatu.

Dane wejściowe

Do uruchomienia aplikacji wymagane będą dwa pliki. Oba w formacie JSON, tak jak w przykładach. Uruchamiając aplikację musimy mieć możliwość określenia **bezwzględnych ścieżek** do obu z tych plików jako argumenty w wierszu poleceń.

Zamówienia

Pierwszy plik to lista, do 10000, zamówień (np. `orders.json`):

```
Unset
[
  {
    "id": "ORDER1",
    "value": "150.00",
    "promotions": ["mZysk", "BosBankrut"]
  },
  ...
]
```

Gdzie:

- `id` - identyfikator zamówienia,
- `value` - kwota zamówienia z dokładnością do dwóch miejsc po przecinku,
- `promotions` - lista identyfikatorów promocji zależnych od metod płatności, które mogą być zaaplikowane do tego zamówienia. Nazwy tych promocji są tożsame z nazwami metod płatności.
    - Lista ta jest opcjonalna, a gdy nie jest zdefiniowana, można stosować wyłącznie promocje za wydane punkty - patrz Rabaty, zasady 3 i 4.
    - Nawet kiedy karta z dostępnych metod płatności nie jest zdefiniowana w liście “promotions”, wciąż można jej użyć do opłacenia zamówienia - w takiej sytuacji nie można jedynie zastosować związanej z nią promocją.

Metody płatności

Drugi plik to lista, do 1000, posiadanych przez klienta metod płatności wraz z przysługującymi im zniżkami i limitem środków do wykorzystania (np. `paymentmethods.json`):

```
Unset
[
  {
    "id": "PUNKTY",
    "discount": "15",
    "limit": "120.00"
  },
  {
    "id": "mZysk",
    "discount": "10",
    "limit": "180.00"
```

```
  },
  ...
]
```

Gdzie:

- `id` – nazwa metody płatności / promocji; dla punktów lojalnościowych zawsze `PUNKTY`
- `discount` – liczba całkowita określająca procentowy rabat,
- `limit` – maksymalna kwota dostępna w danej metodzie płatności (z dokładnością do dwóch miejsc po przecinku).

Uruchomienie aplikacji i spodziewany wynik

Aplikację będziemy uruchamiać używając Javy 21:

```
Unset
java -jar /home/…/app.jar /home/…/orders.json /home/…/paymentmethods.json
```

Na wyjściu program powinien wypisać, na standardowym wyjściu, **wartość wydanych środków z rozbiciem na konkretne metody płatności** (w dowolnej kolejności, sumarycznie, nie per zamówienie):

```
Unset
<id_metody_1> <wydana_kwota>
<id_metody_2> <wydana_kwota>
...
```

Przykład

```
Unset
mZysk 62.17
PUNKTY 100.00
```

Kompletny przykład

`orders.json`

```
Unset
[
 {"id": "ORDER1", "value": "100.00", "promotions": ["mZysk"]},
 {"id": "ORDER2", "value": "200.00", "promotions": ["BosBankrut"]},
 {"id": "ORDER3", "value": "150.00", "promotions": ["mZysk", "BosBankrut"]},
 {"id": "ORDER4", "value": "50.00"}
]
```

`paymentmethods.json`

```
Unset
[
  {"id": "PUNKTY", "discount": "15", "limit": "100.00"},
  {"id": "mZysk", "discount": "10", "limit": "180.00"},
  {"id": "BosBankrut", "discount": "5", "limit": "200.00"}
]
```

Wynik

```
Unset
mZysk 165.00
BosBankrut 190.00
PUNKTY 100.00
```

Wyjaśnienie

- Rabat banku `mZysk` można wykorzystać na zamówieniu `ORDER1` lub `ORDER3`. Na `ORDER3` daje większy rabat (15.00 vs 10.00), więc klient płaci za `ORDER3` 150 - 10% = **135.00** kartą `mZysk`.
- Rabat `BosBankrut` można użyć na `ORDER2` lub `ORDER3`, ale `ORDER3` zostało już przypisane do `mZysk`, a `BosBankrut` daje większy rabat na `ORDER2` niż na `ORDER3` (10.00 vs 7.50), więc klient płaci za `ORDER2` 200 - 5% = **190.00**.
- `ORDER1` może być w całości opłacony punktami: 100 - 15% = **85.00**.
- Dla zamówienia `ORDER4` klient może zapłacić **15.00** punktami (czyli ≥10%), więc przysługuje mu rabat 10%: 50 - 10% = 45.00. Pozostałe **30.00** dopłaca kartą `mZysk`.
- Uwaga: nie potwierdzaliśmy optymalności tego rozwiązania. Będziemy również przyznawać punkty za nieoptymalne rozwiązania, ale niepełne.

Nasze oczekiwania

- W odpowiedzi na maila z zadaniem spodziewamy się archiwum .zip lub .tar.gz z:
  - kodem źródłowym aplikacji;
  - plikiem .jar ze zbudowaną aplikacją
- Można korzystać z dowolnych bibliotek, należy jednak wiedzieć, co robią. Uwaga! Prosimy aby jar zawierał wszystkie wymagane zależności aplikacji (tzw. fat-jar).
- Aplikacja powinna być napisana w Javie 17 lub 21
- To, na co zwracamy uwagę:
  - poprawność wyników;
  - czytelność kodu;
  - jakość projektu - chcielibyśmy, aby kod, a przynajmniej jego kluczowe fragmenty, były pokryte testami
- Możesz uzyskać dodatkowe punkty za użycie narzędzia do budowania projektu oraz stworzenie pliku README wraz z kluczowymi informacjami.