Streamy - projekt

Wyobraźmy sobie, że mamy dane ze sklepu internetowego nazwanego... (w sumie to nazwij go sobie jak masz ochotę ☺).

Dane te są dostarczone w postaci obiektu klasy `DataFactory`, która fabrykuje takie dane.

Fabrykujemy te dane w taki sposób, bo nie poznaliśmy jeszcze metod na odczyt danych z pliku, bądź też z bazy danych.

W naszym sklepie będziemy operować na schemacie klas, który jest przedstawiony poniżej.

Klasa Purchase oraz enumy Delivery, Payment oraz Status

```
public class Purchase {

    private final Client buyer;

    private final Product product;

    private final long quantity;

    private final Delivery delivery;

    private final Payment payment;

    private final LocalDate when;

    private Status status = Status.PAID;

    // konstruktory, gettery itp

    public enum Delivery {
        IN_POST,
        UPS,
        DHL
    }

    public enum Payment {
        CASH,
        BLIK,
        CREDIT_CARD
    }

    public enum Status {
        PAID,
        SENT,
        DONE
    }
}
```

Klasa Client

```
public class Client implements Comparable<Client> {
```

```
    private final String id;

    private final String name;

    private final String surname;

    private final BigInteger pesel;

    private final String city;

    // konstruktory, gettery itp

    @Override
    public int compareTo(final Client o) {
        return this.id.compareTo(o.id);
    }

    @Override
    public boolean equals(final Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        final Client client = (Client) o;
        return Objects.equals(pesel, client.pesel);
    }

    @Override
    public int hashCode() {
        return Objects.hash(pesel);
    }
}
```

Klasa Product oraz enum Category

```
public class Product implements Comparable<Product> {

    private final String id;

    private final String name;

    private final Category category;

    private final Money price;

    // konstruktory, gettery itp

    @Override
    public int compareTo(final Product o) {
        int thisNumber = Integer.parseInt(id.substring(7));
        int otherNumber = Integer.parseInt(o.id.substring(7));
        return thisNumber - otherNumber;
    }

    public enum Category {
        HOBBY,
        CLOTHES,
        GARDEN,
        AUTOMOTIVE
    }
}
```

Klasa Money

```
public class Money {

    private final BigDecimal value;

    private final Currency currency;

    // konstruktory, gettery itp

    public enum Currency {
        PLN,
        EUR
    }
}
```

Mając dane jakie dostarcza nam klasa `DataFactory`, wykonaj polecenia wymienione poniżej. Wszędzie stosuj podejście funkcyjne i Streamy.

Najpierw zapoznaj się z poleceniami do wykonania. Kod klasy `DataFactory` tworzącej dane do wykorzystania znajdziesz w formie kodu źródłowego umieszczonego w poście z zadaniem.

W poleceniach wyróżniamy 3 poziomy skomplikowania zagadnienia: pierwszy, drugi oraz trzeci 😉.

Zadania z poziomu pierwszego:

1. Oblicz jaka ilość klientów dokonała zakupu w naszym sklepie.
2. Oblicz jaka ilość klientów płaciła Blikiem.
3. Oblicz jaka ilość klientów płaciła kartą kredytową.
4. Oblicz jaka ilość zakupów została wykonana w walucie `EUR`.
5. Oblicz ile unikalnych kupionych produktów zostało zakupionych w `EUR`.

Zadania z poziomu drugiego:

1. Oblicz ile PLN wydała w sklepie każda z osób, które dokonały u nas zakupu. Uwzględnij tylko zakupy dokonane w PLN.
2. Przygotuj metodę, która przyjmie konkretną kategorię i dla tej kategorii zwróci mapę, gdzie kluczem będzie `id` klienta, a wartością ilość kupionych przez niego produktów z podanej kategorii (weź pod uwagę tylko te transakcje, w których ilość kupionych produktów jest większa niż 1).
3. Każde zamówienie początkowo ma status `PAID`. Zaktualizuj status wszystkich zamówień, wykorzystując sprawdzenie statusu każdego konkretnego zamówienia poprzez kod klasy `OrderService` podany poniżej. Aby sprawdzić status każdego zamówienia wykorzystaj kod klasy `OrderService` podany poniżej. Na koniec oblicz ile zamówień zostało przetworzonych, czyli mają status `DONE`.

W rzeczywistości, takie rzeczy sprawdzałoby się przykładowo w innym systemie zewnętrznym wywołując jego API. Tutaj natomiast, zmiana takiego statusu zamówienia jest "na sztywno" określana przez metodę `checkOrderStatus()`.

Klasa OrderService

```
import java.util.Set;

public class OrderService {

    private static final Set<String> PAID_STATUSES
        = Set.of("0", "5", "12", "15", "18");

    private static final Set<String> SENT_STATUSES
        = Set.of("1", "2", "4", "7", "8", "11", "13", "16", "19", "21", "25");

    public static Purchase.Status checkOrderStatus(final Purchase purchase) {
        if (PAID_STATUSES.contains(purchase.getProduct().getId().replace("product", ""))) {
            return Purchase.Status.PAID;
        } else if (SENT_STATUSES.contains(purchase.getProduct().getId().replace("product", ""))) {
            return Purchase.Status.SENT;
        } else {
            return Purchase.Status.DONE;
        }
    }
}
```

4. Oblicz ilu unikalnych klientów kupiło produkt wyceniony w `EUR` (klienci nie mogą się powtarzać, pamiętaj, że jeden mógł kupić kilka produktów). Dodatkowo stwórz mapę w której kluczem jest id klienta, a wartością lista zakupów produktów tego klienta w `EUR`.

5. Przygotuj mapę, gdzie kluczem będzie rocznik klienta, a wartościami, lista wszystkich produktów jakie klient z danego rocznika kupił. Rocznik weź z numeru PESEL, nie musi być to pełne 1987, może być przykładowo 87. Posortuj mapę po kluczu rosnąco.
6. Stwórz mapę, gdzie kluczem będą roczniki, a wartością unikalny zestaw kategorii produktów kupionych przez dany rocznik.
7. Jaki jest drugi najczęściej kupowany produkt? Jeżeli kilka produktów jest kupionych w takiej samej ilości, posortuj je alfabetycznie po `id`, i nadal weź drugi. Czyli sortujesz najpierw po największej ilości wystąpień danego produktu, a potem po `id`.

Zadania z poziomu trzeciego:

1. Dla ludzi starszych niż 50 lat stwórz strukturę, w której zawrzesz informacje: rocznik, najmniej popularna kategoria dla danego rocznika, ilość transakcji dla danego rocznika w obrębie danej kategorii. Mówiąc najmniej popularna mamy na myśli, najmniejszą ilość dokonanych zakupów w obrębie danej kategorii. Np: "rocznik: 62, najmniej popularna kategoria: GARDEN, transakcje: 5".
2. Który rocznik kupił najwięcej produktów?