Training materials

- The Java Tutorials. Trail: Collections.
  Наиболее важные уроки: Interfaces, Implementations.
  В разделе The Collection Interface пропустите подпункт Aggregate Operations в пункте Traversing Collections (на обработку коллекций через стримы будет отдельная задача).
- **Course on learn.epam.com Java. Collections.**
- The Java Tutorials. Lesson: Generics.
- И. Блинов. Глава 10.

Code Exercise

Create the package named **by.epam.lab** for entities classes of two types (see the task inheritance1):

– superclass Purchase that represents a product purchase,

– subclass PricePurchase for a purchase with a price discount.

File in.csv contains a series of **correct** text lines. Every odd line contains a set of values separated by a semicolon and corresponds to a single object of a superclass or a subclass depending on the values number in the line. Every even line contains a weekday of the purchase defined by the previous odd line.

Two purchases are equal if they have the same names and prices.

Define the Runner class in the default package.

The algorithm of the method main( ):

1. load the content of the file in.csv into **the map** where a purchase is a key and a weekday of last purchase is a value;
2. print the map by for–each cycle;
3. load the content of the file in.csv into **the map** where a purchase is a key and a weekday of first purchase is a value;
4. print the map by for–each cycle;
5. find the first and the last weekdays for bread with price 1.55;
6. find the first weekday for bread with price 1.70;
7. remove all entries from the first map where the purchase name is meat;
8. remove all entries from the second map on FRIDAY;
9. print maps by for–each cycle;
10. load instances of the subclass PricePurchase from the file in.csv into **the list** (List<**PricePurchase**>);
11. print the total cost of these purchases;
12. load the content of the file in.csv into **the enumerated map** where a weekday is a key and purchases list for this weekday is a value;
13. print the map by for–each cycle;
14. print the total cost of all purchases for each weekday;
15. find all purchases on MONDAY.

The example of in.csv:

```
bread;155;1;20
MONDAY
```

milk;131;2
SUNDAY
bread;154;3
FRIDAY
bread;155;5
SUNDAY
potato;80;2;10
FRIDAY
butter;270;1
SUNDAY
butter;241;1;50
WEDNESDAY
meat;800;2;90
THURSDAY
potato;80;5;10
THURSDAY
milk;131;2
WEDNESDAY
bread;154;3
THURSDAY
bread;155;5
SUNDAY
meat;900;2;70
MONDAY
potato;80;3;20
MONDAY

Ограничения и замечания к задаче 1

– З**апрещен** функционал **java 8** (стримы, лямбды, появившиеся классы и интерфейсы).

– **Запрещены** raw types. Как следствие, не использовать класс Object. Это задача как по теме Collections framework, так и **Generic Types and Methods (!!!)**.

– З**апрещен** функционал класса **Objects**. Изучите способы реализации методов equals() и hashCode() (например, у Дж. Блоха).

– Создать три мэпа (пункты 1, 3, 12) и лист (пункт 10) за **один** проход по файлу (в одном цикле и без дополнительных структур данных).

– Критерий дня первой/последней (first/last) покупки зависит от очередности появления **в файле**, а не по правилу первая покупка - это та покупка, которая ближе к понедельнику, а последняя - та, которая ближе к воскресенью. Внимательно изучите тестовый пример.

– Создать **единый** статический метод раннера для вывода мэпа на консоль (пункты 2, 4, 13). Формат вывода - последовательность строк вида key => value, см. примеры ниже.

– Создать **единый** статический метод раннера для поиска элемента в мэпе (пункты 5, 6, 15).

– Создать **единый** статический метод раннера для удаления элементов из мэпа (пункты 7, 8).

– Для вычисления суммы покупок по листу создать **единый** статический метод раннера и использовать его для пунктов 11 и 14.

– Не забыть проверить решение, когда нет исходного файла.

– Примеры бессмысленных идентификаторов для коллекций: map1, secondMap, …

– В метод toString() классов покупок добавьте вывод имени класса в начало csv строки, см. примеры ниже.

– После чтения вышеприведенного файла должны быть получены следующие мэпы:

First purchase map:

PricePurchase;butter;2.41;1;0.50;1.91=>WEDNESDAY
Purchase;bread;1.54;3;4.62=>FRIDAY
PricePurchase;bread;1.55;1;0.20;1.35=>MONDAY
PricePurchase;meat;9.00;2;0.70;16.60=>MONDAY
Purchase;milk;1.31;2;2.62=>SUNDAY
PricePurchase;meat;8.00;2;0.90;14.20=>THURSDAY
Purchase;butter;2.70;1;2.70=>SUNDAY
PricePurchase;potato;0.80;2;0.10;1.40=>FRIDAY

Last purchase map:

PricePurchase;butter;2.41;1;0.50;1.91=>WEDNESDAY
Purchase;bread;1.54;3;4.62=>THURSDAY
PricePurchase;bread;1.55;1;0.20;1.35=>SUNDAY
PricePurchase;meat;9.00;2;0.70;16.60=>MONDAY
Purchase;milk;1.31;2;2.62=>WEDNESDAY
PricePurchase;meat;8.00;2;0.90;14.20=>THURSDAY
Purchase;butter;2.70;1;2.70=>SUNDAY
PricePurchase;potato;0.80;2;0.10;1.40=>MONDAY

Enumerated map:

MONDAY=>[PricePurchase;bread;1.55;1;0.20;1.35, PricePurchase;meat;9.00;2;0.70;16.60,
PricePurchase;potato;0.80;3;0.20;1.80]
WEDNESDAY=>[PricePurchase;butter;2.41;1;0.50;1.91, Purchase;milk;1.31;2;2.62]
THURSDAY=>[PricePurchase;meat;8.00;2;0.90;14.20, PricePurchase;potato;0.80;5;0.10;3.50,
Purchase;bread;1.54;3;4.62]
FRIDAY=>[Purchase;bread;1.54;3;4.62, PricePurchase;potato;0.80;2;0.10;1.40]
SUNDAY=>[Purchase;milk;1.31;2;2.62, Purchase;bread;1.55;5;7.75,
Purchase;butter;2.70;1;2.70, Purchase;bread;1.55;5;7.75]

– Создание тестов опционально.