Создайте пакет с именем by.gsu.epamlab для класса «Покупка», который представляет собой оптовую покупку того же продукта с той же ценой в течение недели.

Класс Purchase должен реализовать интерфейс Comparable, параметризованный этим классом.

Поля класса:

•наименование товара,

• цена (в BYN),

• количество купленных единиц,

• дисконтные проценты,

• недельный день (создание перечисления WeekDay).

Конструкторы:

• конструктор no-arg,

• параметризованный конструктор.

Методы:

• геттеры / сеттеры;

• getCost () - возвращает стоимость покупки:

цена \* номер \* (100%) / 100, округленная до 1,00 BYN;

• toString () - возвращает строковое представление покупки в формате csv: каждое не постоянное поле и стоимость покупки, разделенные символом «;»; символ);

• compareTo (Покупка покупки) - сравнивает количество купленных единиц покупок и возвращает отрицательное целое число, ноль или положительное целое число, так как этот объект меньше, равен или больше указанной покупки.

Файл src \ in.txt состоит из 11 строк с правильными данными.

Первая строка содержит номер PURCHASES\_NUMBER от 0 до 10. Следующие 10 строк содержат информацию о 10 действительных покупках.

Значения в строке разделяются пробелами.

Значение дня недели задается цифрами от 0 до 6, где 0 - воскресенье, ..., 6 - суббота.

Определите класс Runner в пакете по умолчанию, где:

1. Создайте массив для покупок PURCHASES\_NUMBER.

2. Инициализируйте этот массив данными файла.

3. Вывести содержимое массива на консоль в следующем формате:

константы класса

purchase [0]

...

purchase [PURCHASES\_NUMBER - 1]

4. Рассчитайте среднюю стоимость всех покупок (3 цифры после точки), общую стоимость всех покупок в понедельник, день с максимальной покупной стоимостью. Выведите их на консоль.

5. Сортируйте массив по номеру поля в порядке возрастания методом sort () класса Arrays.

6. Выведите содержимое массива на консоль в формате выше.

7. Найдите покупку с номером, равным 5, методом binarySearch () класса Arrays и выведите его.

Create the package named by.gsu.epamlab for a class called Purchase that represents a wholesale purchase of the **same** product of the **same** price during a week. The class Purchase must implement the interface Comparable parameterized by this class.

**Class fields**:

* product name,
* **price** (in BYN),
* **number** of purchased units,
* discount **percent**,
* week day (create **enumeration** WeekDay).

**Constructors**:

* no-arg constructor,
* parameterized constructor.

**Methods**:

* getters/setters;
* getCost( ) – returns the purchase cost:   
  price \* number \* (100 – percent) / 100, **rounded to 1.00 BYN**;
* toString( ) – returns a string representation of a purchase in the csv–format: each **non constant** field and the purchase cost, separated by the ";" symbol);
* compareTo(Purchase purchase) – compares **numbers** of purchased units of purchases and returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified purchase.

File src\in.txt consists of 11 lines **with correct data**.

The first line contains the number PURCHASES\_NUMBER from 0 to 10. Next 10 lines contain information about 10 valid purchases. Values within a line are separated by spaces. The value for a weekday is given by numbers from 0 to 6, where 0 is Sunday, … , 6 is Saturday.

Define the Runner class in the **default** package, where:

1. Create an array for PURCHASES\_NUMBER purchases.

2. Initialize this array by the file data.

3. Output the array content to the console in the following format:

class constants

purchase[0]

…

purchase[PURCHASES\_NUMBER - 1]

4. Calculate the mean cost of all purchases (3 digits after the point), the total cost of all purchases on Monday, the day with the maximum purchase cost. Output them to the console.

5. Sort the array by the field number in the ascending order by the method sort( ) of the class Arrays.

6. Output the array content to the console in the format above.

7. Find some purchase with number equaled to 5 by the method binarySearch( ) of the class Arrays and output it.

**Замечания и ограничения**

– В данной задаче также запрещаю создавать класс для финансовой величины.

– Пункт 2 выполнить в отдельном цикле.

– Пункт 4 выполнить в одном цикле (без счетчика!).

– Класс Purchase должен реализовывать интерфейс Comparable, параметризованный объектами этого же класса. Следовательно, структура класса Purchase такая:

public class Purchase implements Comparable<Purchase> {

…

public int compareTo(Purchase purchase) {

if(number < purchase.number) {

return –1;

}

//и так далее.

//Можно возвращать ЛЮБОЕ отрицательное число.

**//Подумайте, как обойтись без операторов if.**

}

}

   При реализации механизма сравнения объектов класса через метод   
compareTo( ) интерфейса Comparable массив объектов этого класса сортируется следующим образом:

//объявление и инициализация массива

final int PURCHASES\_NUMBER = sc.nextInt();

Purchase[ ] purchases = new Purchase[PURCHASES\_NUMBER];

//цикл инициализации массива из файла

…

//пункты 3, 4

...

//сортировка

**Arrays.sort(purchases);**

– Для дня недели создать перечисление. Причем не нужно его прятать внутрь класса покупки. Оно должно быть внешним, как следствие, в отдельном файле. Элементы перечисления – константы, что требует соответствующего именования.

– Иногда ошибочно интерпретируют фразу пункта 3 задачи 3: «Calculate …, the day with the maximum purchase cost». Т.е. речь идет о дне, когда была зарегистрирована покупка (в единственном числе) с максимальной стоимостью. А вот если бы покупки были во множественном числе (the day with the maximum of purchases costs), тогда следовало бы найти суммарные стоимости за каждый день, а потом среди них максимум.

– Начальные значения константам класса задать в самом классе, остальные данные ввести из файла.

– Читать файл с помощью функционала класса Scanner. См. пример ниже, а также И. Блинов стр. 242. Установить программно англоязычную локаль. По умолчанию она использует точку в качестве разделителя целой и дробной частей вещественного числа (в кирилличных локалях –  запятая).

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.util.Scanner;

public class ScannerSample {

public static void main (String[] args) {

Scanner sc = null; //единственная переменная, которая объявляется до блока try!

//уменьшайте область видимости локальных данных метода.

try {

sc = new Scanner(new FileReader("src/in.txt"));

            sc.useLocale(Locale.ENGLISH); //установка локали

String str = sc.next(); //чтение строки,

int n = sc.nextInt(); //целого,

double x = sc.nextDouble(); //вещественного

System.out.println(str + ";" + n + ";" + x);

} catch (FileNotFoundException e) {

System.err.println("Input file is not found");

} finally {

if(sc != null) {

sc.close();

}

}

}

}

– По условию файл корректный. Не надо проверять вводимые значения. Если встретится любая ошибка ввода, то приложение должно **завершить работу с рантаймовым исключением**. Например, если задать в первой строке значение -1 (или 11, или строку), в следующих строках для дня недели 7 или строку и т.д.

– Обязательно проверить работу раннера на крайних значениях, т.е. когда в первой строке файла находится значение 0 или 10.

Если в первой строке файла находится значение 0, то на консоль нужно вывести:  
Mean cost = 0.000

The total cost on Monday = 0.00

The day with the maximum cost purchase is null

Required purchase is not found

– В джаве, в отличие от С++, можно создавать массивы длиной 0, что часто практикуется. См. детали у Дж. Блоха “Эффективное программирование”, статья 27.

– Исходный файл корректный. Однако он может отсутствовать. И этот  случай приложение должно обрабатывать корректно.

Правильно расположите блок catch (FileNotFoundException e) и обязательно протестируйте работу приложения на отсутствующем файле.

– Этапы выполнения задачи 1

Этап 1. Реализация авторского решения.

Создать архив с решением **classes3.jar** и выложить в свою папку jars.

Этап 2. Анализ правильного решения.

После появления jar-файла скрипт через некоторое время откроет доступ к текстовому документу, где описан процесс реализации задачи.

Исправьте присутствующие в Вашем решении недостатки и выложите его в архиве **classes3\_2.jar**.

Этап 3. Рецензия.

Я выложу рецензию на исправленное решение.