Task 1

Create the package named **by.gsu.pms** for the superclass and subclasses.

Define the superclass, describing the commodity purchase.

Superclass fields:

- commodity name,
- price in belarusian rubles,
- number of purchased units.

Constructors:

- default constructor,
- general-purpose constructor.

Methods:

- getters/setters;
- getCost( ) – calculating the purchase cost;
- toString( ) – converting of an object to a string in the following format: each field and the purchase cost, separated by the ";" symbol);
- equals( ) – comparing of purchases (equal if name and price are the same).

Define the first subclass for the purchase with a price discount and override necessary methods.

Define the second subclass for the purchase with a discount to be presented if the number of purchased units is greater than the given subclass constant. A discount rate is given by the percent from the purchase cost. Override necessary methods.

File src\in.txt consists of 6 lines with correct data. Every line contains needed data separated by spaces for 1 object of the superclass or the first subclass or the second one. Every line begins with some identifier of the purchase class, then other data follow.

The line example for the superclass object:

```
GENERAL_PURCHASE  Milk  2500  3
```

Define the Runner class in the default package, where:

1. Create an array for 6 objects.
2. Input data from the given file into array.
3. Print the array content to the console (one element per line).
4. Print the maximum cost purchase.
5. Determine whether all purchases are equal.

Realize subtasks 2–5 by the single cycle.

Замечания

– Подкласс в своем имени содержит имя (по крайней мере ключевое слово) суперкласса. Как правило, ключевое слово последнее.

– Подклассы не должны изменять поля базового класса. Можно вводить новые поля и переопределять методы!

– Примеры расчета стоимости в подклассах:

1) пусть скидка в цене 50, цена 300, количество 2;
тогда стоимость = (300 – 50) * 2 = 500.

2) пусть процент скидки 5.825, цена 500, количество 20, количество, которое надо превысить 15.
т.к. 20 > 15, то стоимость = 500 * 20 * (1 – 5.825/100) = 9418 (стоимость по-прежнему целая!).

– В toString( ) подклассов использовать вызов toString( ) суперкласса.

– Строки, как и экземпляры других классов, сравниваются через метод equals( ).

– Для создания объектов применить шаблон Factory (стр. 115–116, а также стр. 87–88, но хуже). Схема фабричного класса:

```
public class PurchasesFactory {
        private enum PurchasesKinds {
                GENERAL_PURCHASE, …
        }
        public Purchase getClassFromFactory(Scanner sc) {
                String id = sc.next();
                …
                PurchasesKinds kind = PurchasesKinds.valueOf(id);
                switch(kind) {
                        case GENERAL_PURCHASE :
                                return new Purchase(…);
                        case … :
                        …
                }
        }
}
```

– В задаче 2 пункты 3 и 5 совпадают. Следовательно, реализацию нужно выносить в статический метод. Не забывайте, что в раннере тип доступа public только у метода main( ). Все остальные private!

Task 2

Create the package named **by.gsu.pms** for the entities classes.
Define the class Commodity, describing the commodity.

Superclass fields:

- commodity name,
- price in belarusian rubles.

Constructors:

- default constructor,
- general-purpose constructor.

Methods:

- getters/setters;

- toString( ) – converting of an object to a string in the csv–format: each field and the purchase cost, separated by the ";" symbol.

Define the superclass AbstaractPurchase, describing the commodity purchase and implementing interface Comparable<AbstaractPurchase>.

Superclass fields:

- commodity,
- number of purchased units.

Constructors:

- default constructor,
- general-purpose constructor.

Methods:

- getters/setters;
- getCost( ) – calculating the purchase cost;
- toString( ) – converting of an object to a string in the csv–format: each field and the purchase cost, separated by the ";" symbol.
- compareTo(AbstaractPurchase purchase) – comparing purchases by the cost *decreasing*.

Define the first subclass for the purchase with a price discount and override necessary methods.

Define the second subclass for the purchase with a percent discount and override necessary methods.

Define the third subclass for the purchase with an addition for transport expenses and override necessary methods.

Define the Runner class in the default package, where:

1. Create unique commodity for purchasing.
2. Create an array for 6 objects (2 – of every subclass).
3. Print the array content to the console (one element per line).
4. Sort an array on the cost *decreasing* by the method sort( ) of the class Arrays.
5. Print the array content to the console (one element per line).
6. Print purchase with minimum cost.
7. Perform an additional task.

!!!! Для выводы реализовать минимум один интерфейс

Реализовать задачу согласно приведенной схеме