Task 1

Разработать *базовый класс*, определяющий покупку товара:

Поля:

- название товара;
- цена в рублях;
- кол-во единиц товара.

Конструкторы:

- по умолчанию;
- с параметрами.

Методы:

- установки/считывания полей (можно использовать свойства, в т.ч. автоматические);
- GetCost( ) – вычисляет стоимость покупки;
- ToString( ) – переводит объект в строку с разделителями «;»;
- Equals( ) – сравнивает две продажи (считаются равными, если совпадают *название* и *цена*).

Разработать *первый производный класс*, для покупки товара с фиксированной скидкой от цены и переопределить нужные методы (GetCost( ) и ToString( )).

Разработать *второй производный класс* (от базового) со скидкой к цене, если количество единиц товара не меньше некоторой константы подкласса. Переопределить нужные методы.

Разработать консольное приложение, выполняющее следующее:

1 Создать массив из шести объектов (2 – базового класса, по 2 – каждого производного класса).

2 Вывести массив на консоль.

3 Найти покупку с максимальной стоимостью и вывести эту покупку после цикла.

4 Определить, являются ли все покупки *равными* (да/нет).

Задачи 2–4 реализовать в *одном общем цикле*.

Task 2

Разработать *базовый класс* Commodity, описывающий товар.

Поля:

- название товара;
- цена в рублях.

Конструкторы:

- по умолчанию;
- с параметрами.

Методы:

- установки/считывания полей;
- ToString( ) – переводит объект в строку с разделителями «;».

Разработать *абстрактный базовый класс* AbstractPurchase, описывающий покупку товара и реализующий интерфейс IComparable<AbstractPurchase>

Поля:

- товар Commodity;
- количество единиц товара.

Конструкторы:

- по умолчанию;
- с параметрами.

Методы:

- установки/считывания полей;
- GetCost( ) – стоимость покупки, абстрактный метод;
- ToString( ) – переводит объект в строку с разделителями «;»;
- CompareTo(AbstractPurchase purchase) – сравнивает покупки по ***убыванию*** стоимости.

Разработать *первый производный класс* от AbstractPurchase, в котором продажа товара осуществляется со скидкой от цены. Переопределить нужные методы.

Разработать *второй производный класс* от AbstractPurchase, в котором скидка присутствует, если кол-во единиц товара не меньше некоторого числа, заданного константой в классе. Переопределить нужные методы.

Разработать *третий производный класс* от AbstractPurchase, в котором к стоимости товара добавляются дополнительные транспортные расходы. Переопределить и добавить нужные методы.

Разработать консольное приложение, в котором:

1 Создать массив из шести объектов (по два каждого производного класса).

2 Вывести объекты на консоль через ToString().

3 Отсортировать объекты по *убыванию* с использованием метода Sort( ) класса Array( ).

4 Вывести объекты на консоль через ToString().

Замечания

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

– 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() базового класса.

– В задаче 2 2 и 4 совпадают. Следовательно, реализацию нужно выносить в статический метод. В раннере (класс Program) тип доступа public только у метода Main( ). Все остальные private.