У предпринимателя имеется паром, который перевозит грузы и людей. Поля для класса Паром: грузоподъемность и общий массив для грузов и людей.

Виды транспортировки грузов:

– прямоугольные контейнеры для однородного твердого материала (масса на основе плотности материала, параметров контейнера и массы самого контейнера);

– платформы для любого груза (масса на основе массы груза; платформа является частью парома, т.е. ее масса принимается равной 0);

– цилиндрические цистерны для жидкости (масса на основе плотности жидкости, параметров цистерны и массы самой цистерны).

Спроектировать объектную модель.

В методе *Main* создать необходимые объекты (ничего не вводить, использовать вызовы конструкторов с константными значениями).

Вывести содержимое общего массива.

Отсортировать содержимое массива по людям и видам транспортировки грузам (сначала люди, затем грузы в контейнерах, на платформах, в цистернах).

Вывести содержимое общего массива.

Определить, может ли перевезти паром заданные грузы и людей (*да*/*нет*).

Ограничение: не делать общей иерархии классов для людей и грузов (это не логично). Hint: данные классы должны реализовывать одинаковый интерфейс, через которые их можно объединить в массив.

Замечания

– Сортировку массива реализовать с помощью создания компаратора. Пример сортировки компаратором:

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication2
{

  class Test
  {

    private string name;
    public string Name
    {
      get {return name;}
    }
    public Test(string name)
    {
      this.name = name;
    }

    public override string ToString()
    {
      return name;
    }

  }

//создать класс реализующий интерфейс IComparer или наследованный от Comparer
//принцип, как и для интерфейса IComparable,
//только два аргумента у метода, задающего принцип сравнения
```

```
  class TestComaprer : Comparer<Test>
  {

    public override int Compare(Test x, Test y)
    {
      return (x.Name.CompareTo(y.Name));
    }

  }


  class Program
  {
    static void Main(string[] args)
    {
      Test[] tArr = {
                      new Test("John"),
                      new Test("Mikhael"),
                      new Test("Tod"),
                      new Test("Alex"),
                      new Test("Sam")
                    };
      Console.WriteLine("Source:");
      foreach (var t in tArr)
        Console.WriteLine(t);

      //передать в метод объект-компаратор
      Array.Sort(tArr,new TestComaprer());
      Console.WriteLine("Sorted2:");
      foreach (var t in tArr)
        Console.WriteLine(t);

    }
  }
}
```

– Есть несколько приемлемых реализаций заданной сортировки.
1. Решение «*в лоб*»: в методе Compare( ) класса компаратора использование операции is и if-ов. При вдумчивом подходе ифов не так и много.
2. Определить в необходимых классах метод GetId( ) (или GetOrdinal( ) по аналогии с методом перечисления; перечисление из сортируемых сущностей очень даже будет к месту). Но это все же лишний метод в классах.
3. Самое красивое решение – совместное использование перечисления и рефлексии.

– Дополнение к code conventions.
Идентификаторы булевских методов и переменных имеют вид IsXxx( ), где Xxx – прилагательное или причастие.
Пример.

```
 public boolean IsPassed()
 {
   return true;
 }
```

– Ради удобства решения задачи человека хочется приравнять к грузу. Однако это неестественно с точки зрения житейской логики. Поэтому ООП-гуру не советуют так делать. Всегда нужно думать о развитии проекта. Если возникнет у заказчика желание получить новый функционал (что есть обычная практика

при его платежеспособности!), то раньше или позже НЕЕСТЕСТВЕННАЯ ИЕРАРХИЯ с большой вероятностью вылезет боком.
Таким образом, ограничение о разной иерархии классов для людей и грузов является разумным.
Почитайте на http://skipy.ru/architecture/module_design.html.