Домашнє завдання

Інтерфейси. Реалізація стандартних інтерфейсів `IComparable, IComparer, IEnumerable. Ітератори(yield).`

Завдання 1. **Виконати на вибір одне із завдань**

1.1.

- Визначити інтерфейси `IWalk, IEat, ISleep`. Реалізувати ці інтерфейси для класів різних тварин зоопарку(`Bear, Parrot, …`).
- Визначити інтерфейс `IWatch(`спостерігати), який реалізувати для `класів VideoCamera` та класу `ZooWorker(`працівника зоопарку).
- Створити клас `Zoo,` що міститиме масиви(або колекції `List<>, ArrayList`) тварин, робітників, відеокамер.

Змоделювати роботу зоопарку:

- Тварини можуть їсти, прогулюватися, тощо. ○ За цим слідкують відеокамери, та можуть слідкувати працівники.
- Працівники можуть годувати тварин.

1.2.*

Реалізувати класи `House, Basement, Walls,` Door`, Window, Roof, Worker, TeamLeader, Team` та інтерфейси `IWorker, IPart`.

Всі частини будинку повинні бути класами та реалізувати інтерфейс `IPart`, для робочих і бригадира надається базовий інтерфейс `IWorker.`

Бригада будівельників (`Team`) будує будинок (`House`). Будинок складається з фундаменту (`Basement`), стін (`Wall`), вікон (`Window`), дверей (`Door`), даху (`Roof`).

Згідно проекту, в будинку повинно бути 1 фундамент, 4 стіна, 1 двері, 4 вікна і 1 дах.

Врахуйте також, що двері бувають різні: скляні, металеві, дерев'яні, профільні тощо. Фундамент також буває різного виду: стрічковий, збірний, стовпчастий, суцільний, зварний тощо. Реалізувати це за допомогою перелічувальної константи (`enum`).

Бригада починає роботу, і будівельники послідовно будують будинок, починаючи з фундаменту. Кожен будівельник не знає наперед, на чому завершився попередній етап будівництва, тому він "перевіряє", що вже побудовано і продовжує роботу.

Якщо в гру вступає бригадир (`TeamLeader`), він не будує, а формує звіт, що вже побудовано і яка частина роботи виконана.

В кінцевому рахунку на консоль виводиться повідомлення, що будівництво будинку завершено і відображається "малюнок будинку" (циклами).

Завдання 2.

Визначити клас Напій(`Drink`)

з полями ○ назва напою,

- тип напою(власного перелічувального типу)

- виробник ○ кількість ккал ○ ціна

методами ○

конструктори

- перевизначити метод `ToString()` ○ реалізувати інтерфейс `IComparable`(як метод класу `int CompareTo(object)`) :
  порівнювати напої за типом, потім за назвою
- реалізувати інтерфейс `IComparable< >`(як метод класу int CompareTo(Drink)) :
  порівнювати за назвою напою

Визначити 3 класи компараторів, які реалізують інтерфейс `IComparer`, тобто метод `int Compare(object, object)` для ○ порівняння напоїв за ціною(за зростанням) ○ порівняння напоїв за енергетичною цінністю, ккал(спаданням) ○ порівняння за виробником(за зростанням)

Визначити 3 класи компараторів, які реалізують інтерфейс `IComparer<Drink>`, тобто метод `int Compare(Drink, Drink)` для ○ порівняння за ціною(за спаданням) ○ порівняння за ккал(зростанням)

- порівняння за виробником(зростання)

Реалізувати інтерфейс `IEquatable<>` Перевірити роботу класу.
Підготувати 2 колекції: `ArrayList`, `List<Drink>`.
Перевірити роботу реалізованих інтерфейсів за допомогою сортування колекцій(`Sort`), пошуку(`IndexOf`).

Завдання 3.
Доповнити один(!) з створених раніше класів(`Student`, `Department` чи ін.) реалізацією інтерфейсу `IEnumerable`.

- для класу `Student` при використанні `foreach`-ом повинен проходитися рваний масив з оцінок
- для класу `Department` при використанні `foreach` повинен проходитися масив(чи список) з працівників