МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ»

Г.Э. Вощинская, Е.М. Лещенко

СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ

Часть 2

Учебно-методическое пособие

Воронеж
Издательский дом ВГУ
2019

Содержание

Также необходимо разработать серию примеров, демонстрирующих основные аспекты работы с данной библиотекой списков.

2. Кольцевой список

Разработать библиотеку обобщенных классов для работы с кольцевыми списками данных. В структуру классов входят как минимум:

- IList<T>: IEnumerable<T> – базовый интерфейс для всех кольцевых списков;
  - методы:
    - int Add(T value);
    - void Clear();
    - bool Contains(T value);
    - int IndexOf(T value);
    - void Insert(int index, T value);
    - void Remove(T value);
    - void RemoveAt(int index);
    - IList<T> subList(int fromIndex, int toIndex);
  - свойства:
    - int Count;
    - T this[int index];
- ListException – класс, описывающий исключения, которые могут происходить в ходе работы с кольцевым списком (также можно написать ряд наследников);
- ArrayList<T>: IList<T> – класс кольцевого списка на основе массива;
- LinkedList<T>: IList<T> – класс кольцевого списка на основе связного списка;

- UnmutableList<T>: IList<T> – класс неизменяющегося кольцевого списка, является оберткой над любым существующим списком (должен кидаться исключениями на вызов любого метода, изменяющего список);
- ListUtils – класс различных операций над кольцевым списком;
    - методы:
        - static bool Exists<T>(IList<T>, CheckDelegate<T>);
        - static T Find<T>(IList<T>, CheckDelegate<T>);
        - static T FindLast<T>(IList<T>, CheckDelegate<T>);
        - static int FindIndex<T>(IList<T>, CheckDelegate<T>);
        - static int FindLastIndex<T>(IList<T>, CheckDelegate<T>);
        - static IList<T> FindAll<T>(IList<T>, CheckDelegate<T>, ListConstructorDelegate<T>);
        - static IList<TO> ConvertAll<TI, TO>(IList<TI>, ConvertDelegate<TI, TO>, ListConstructorDelegate<TO>);
        - static void ForEach(IList<T>, ActionDelegate<T>);
        - static void Sort(IList<T>, CompareDelegate<T>);
        - static bool CheckForAll<T>(IList<T>, CheckDelegate<T>);
    - свойства:
        - static readonly ListConstructorDelegate<T> ArrayListConstructor;
        - static readonly ListConstructorDelegate<T> LinkedListConstructor;

Также необходимо разработать серию примеров, демонстрирующих основные аспекты работы с данной библиотекой кольцевых списков.

3. Двусвязный список

Разработать библиотеку обобщенных классов для работы с двусвязными списками данных. В структуру классов входят как минимум:

ŋqqdqyeɎ bɆ dedⱻbɎɜe d ɛɑɒcɎ ɟqqdɎqyè

- IList<T>: IEnumerable<T> – базовый интерфейс для всех двусвязных списков;
  - методы:
    - int Add(T value);
    - void Clear();
    - bool Contains(T value);
    - int IndexOf(T value);
    - void Insert(int index, T value);
    - void Remove(T value);
    - void RemoveAt(int index);
    - IList<T> subList(int fromIndex, int toIndex);
  - свойства:
    - int Count;
    - T this[int index];
- ListException – класс, описывающий исключения, которые могут происходить в ходе работы с двусвязным списком (также можно написать ряд наследников);
- ArrayList<T>: IList<T> – класс двусвязного списка на основе массива;
- LinkedList<T>: IList<T> – класс двусвязного списка на основе связного списка;
- UnmutableList<T>: IList<T> – класс неизменяющегося двусвязного списка, является оберткой над любым существующим списком (должен кидаться исключениями на вызов любого метода, изменяющего список);
- ListUtils – класс различных операций над двусвязным списком;
  - методы:
    - static bool Exists<T>(IList<T>, CheckDelegate<T>);

- static T Find<T>(IList<T>, CheckDelegate<T>);
- static T FindLast<T>(IList<T>, CheckDelegate<T>);
- static int FindIndex<T>(IList<T>, CheckDelegate<T>);
- static int FindLastIndex<T>(IList<T>, CheckDelegate<T>);
- static IList<T> FindAll<T>(IList<T>, CheckDelegate<T>, ListConstructorDelegate<T>);
- static IList<TO> ConvertAll<TI, TO>(IList<TI>, ConvertDelegate<TI, TO>, ListConstructorDelegate<TO>);
- static void ForEach(IList<T>, ActionDelegate<T>);
- static void Sort(IList<T>, CompareDelegate<T>);
- static bool CheckForAll<T>(IList<T>, CheckDelegate<T>);

  - свойства:
    - static readonly ListConstructorDelegate<T> ArrayListConstructor;
    - static readonly ListConstructorDelegate<T> LinkedListConstructor;

Также необходимо разработать серию примеров, демонстрирующих основные аспекты работы с данной библиотекой двусвязных списков.

4. Упорядоченный список

Разработать библиотеку обобщенных классов для работы с упорядоченными списками данных. В структуру классов входят как минимум:

- IList<T>: IEnumerable<T> – базовый интерфейс для всех упорядоченных списков;
  - методы:
    - int Add(T value);
    - void Clear();

        - bool Contains(T value);
        - int IndexOf(T value);
        - void Insert(int index, T value);
        - void Remove(T value);
        - void RemoveAt(int index);
        - IList<T> subList(int fromIndex, int toIndex);
    - свойства:
        - int Count;
        - T this[int index];
- ListException – класс, описывающий исключения, которые могут происходить в ходе работы с упорядоченным списком (также можно написать ряд наследников);
- ArrayList<T>: IList<T> – класс упорядоченного списка на основе массива;
- LinkedList<T>: IList<T> – класс упорядоченного списка на основе связного списка;
- UnmutableList<T>: IList<T> – класс неизменяющегося упорядоченного списка, является оберткой над любым существующим списком (должен кидаться исключениями на вызов любого метода, изменяющего список);
- ListUtils – класс различных операций над упорядоченным списком;
    - методы:
        - static bool Exists<T>(IList<T>, CheckDelegate<T>);
        - static T Find<T>(IList<T>, CheckDelegate<T>);
        - static T FindLast<T>(IList<T>, CheckDelegate<T>);
        - static int FindIndex<T>(IList<T>, CheckDelegate<T>);
        - static int FindLastIndex<T>(IList<T>, CheckDelegate<T>);