Лабораторна робота 3: Лінійні структури даних

Мета роботи:

1) дослідити абстрактні структури даних «Список», «Стек», «Дек», «Черга» та їх реалізації через масив і зв'язаний список
2) опрацювати реалізації абстрактних типів даних «Список», «Стек», «Дек», «Черга» за допомогою одновимірних масивів на мові програмування Java
3) опрацювати реалізації абстрактних типів даних «Список», «Стек», «Дек», «Черга» за допомогою зв'язаних списків на мові програмування Java
4) отримати практичні навички з роботи з класами і методами, які реалізують інтерфейси на мові програмування Java;
5) навчитися обирати абстрактну структуру структури та програмну реалізацію лінійної структури даних відповідно до визначених функціональних вимог та обмежень прикладної задачі.

Інструментальні і програмні засоби:

Java JDK 7 і вище
IDE Eclipse

Склад матеріалів для виконання лабораторної роботи

Пакет **lab3_class_skeleton.stack**
- клас **DLNode**
- клас **SLNode**
- пакет **lab3_class_skeleton.stack**
  - інтерфейс **Stack**
  - клас **ArrayStack**
  - клас **LinkedStack**
- пакет **lab3_class_skeleton.queue**
  - інтерфейс **Queue**
  - клас **ArrayQueue**
  - клас **LinkedQueue**
- пакет **lab3_class_skeleton.list**
  - інтерфейс **List**
  - клас **ArrayList**
  - клас **SinglyLinkedList**
  - клас **DoublyLinkedList**
- пакет **lab3_class_skeleton.deque**
  - інтерфейс **Deque**
  - клас **ArrayDeque**
  - клас **LinkedDeque**

Загальні рекомендації:

Індивідуальні завдання наведені у розділі «Варіанти індивідуальних завдань до лабораторної роботи 3». Робота складається з трьох завдань, які виконуються послідовно.

У першому завданні створюються та оброблюються лінійні структури даних, реалізовані через одновимірний масив: **`ArrayList`**, **`ArrayQueue`**, **`ArrayStack`** або **`ArrayDeque`**. Потрібно реалізувати одну з цих структур даних, яка зазначена у його індивідуальному завданні (табл.1).

У другому завданні створюються та оброблюються лінійні структури даних, реалізовані через зв'язаний список: **`LinkedList`**, **`LinkedQueue`**, **`LinkedStack`** або **`LinkedDeque`**. Потрібно реалізувати одну з цих структур даних, яка зазначена у його індивідуальному завданні (табл.2).

Тип елементів всіх структур даних пакету lab3 (див. «Склад матеріалів для виконання лабораторної роботи») – String. Цей тип обрано як приклад, але при виконанні лабораторної роботи слід обрати тип елементів за індивідуальним завданням (кол. 3 у табл.1 і табл.2). Перед тип, як приступити до виконання першого або другого завдання треба змінити тип елементів в інтерфейсі та класі, що його реалізує. Додаткові обмеження на тип елементів, наприклад додатні цілі числа, слід перевіряти в операціях додавання нового елемента.

У першому та другому завданні обов'язково реалізувати методи, які містять тег //TODO. За необхідності можна написати свої методи, наприклад, для виведення вмісту структури даних, або закритий (private) метод для пошуку вузла за індексом у зв'язаному списку.

Третє завдання використовує вже реалізовані у попередніх завданнях структури даних.

Опис інтерфейсів і класів пакету lab3_class_skeleton

Інтерфейс **`List`** специфікує методи для обробки послідовностей з доступом за індексом:

`boolean add(String e)` – додає елемент **`e`** у кінець списку. Повертає **`true`**, якщо список змінився внаслідок виклику цього методу

`boolean add(int index, String e)`- вставляє елемент **`e`** у позицію **`index`** списку. Повертає **`true`**, якщо список змінився внаслідок виклику цього методу

`String remove(int index)` – вилучає елемент **`e`** з позиції **`index`** списку. Повертає вилучений з цієї позиції елемент.

boolean remove(String e) – вилучає перше входження елемента **e** у список. Повертає **true**, якщо у списку був цей елемент

String get(int index) – повертає елемент списку у позиції **index**

String set(int index, String e) – заміщає елемент в позиції **index** новим значенням **e**

int size() – повертає кількість елементів у списку

boolean isEmpty() – повертає **true**, якщо у списку немає елементів

Інтерфейс **Queue** специфікує методи для обробки черги з політикою FIFO:

boolean enqueue(String e) – додає елемент **e** у хвіст черги. Повертає **true**, якщо черга змінилася внаслідок виклику цього методу

String dequeue() – вилучає і повертає голову черги. Повертає вилучений з цієї позиції елемент, якщо черга не порожня.

String head()– повертає, але не вилучає, голову черги, якщо черга не порожня.

int size() – повертає кількість елементів у черзі

boolean isEmpty() – повертає **true**, якщо у черзі немає елементів

Інтерфейс **Deque** специфікує методи для обробки двосторонньої черги:

boolean addFirst(String e) – додає елемент **e** на початок двосторонньої черги. Повертає **true**, якщо черга змінилася внаслідок виклику цього методу;

boolean addLast(String e) – додає елемент **e** у кінець двосторонньої черги. Повертає **true**, якщо черга змінилася внаслідок виклику цього методу;

String removeFirst() – вилучає і повертає перший елемент двосторонньої черги, якщо черга не порожня;

String removeLast () – вилучає і повертає останній елемент двосторонньої черги, якщо черга не порожня;

String getFirst() – повертає, але не вилучає, перший елемент двосторонньої черги, якщо черга не порожня;

String getLast()– повертає, але не вилучає, останній елемент двосторонньої черги, якщо черга не порожня

int size() – повертає кількість елементів у черзі

boolean isEmpty() – повертає **true**, якщо у черзі немає елементів

Інтерфейс **Stack** специфікує методи для обробки стека:

void push(String item) – додає елемент **item** на верхівку стека;

String pop() – вилучає і повертає елемент з верхівки стека, якщо стек не порожній;

String top() – повертає, але не вилучає, елемент з верхівки стека, якщо стек не порожній;

int size() – повертає кількість елементів у стеку

boolean isEmpty() – повертає **true**, якщо у стеку немає елементів

Завдання

Завдання 3.1. *Реалізувати визначену в індивідуальному завданні лінійну структуру даних через одновимірний масив*

Порядок виконання

1) Описати структуру даних (табл. 1, кол. 2), яка реалізується за допомогою одновимірного масиву. Тип елементів структури даних обрати згідно з варіантом завдання (табл. 1, кол. 3)
2) Створити екземпляр структури даних і продемонструвати працездатність структури даних: вставити і видалити декілька елементів, вивести вміст структури даних. Звернути увагу на такі тестові випадки:
   - вилучення елементу з порожнього списку, черги, деку або стеку;
   - додавання елемента, який не ві
   - додавання елемента, якщо кількість елементів досягла місткості;
   - позиція, куди додається елемент, від'ємна або більше кількості елементів (для списків);
   - позиція, звідки вилучається елемент, від'ємна або більше кількості елементів (для списків);
   - вилучення елементу, якого не міститься у списку.

Завдання 3.2. *Реалізувати визначену в індивідуальному завданні лінійну структуру даних через зв'язний список*

1) Змінити опис поля **`data`** у класі **`SLNode`** або **`DLNode`** (файли **`SLNode.java`** або **`DLNode.java`**) за умовами індивідуального завдання (табл. 2)
2) Реалізувати методи у зв'язаній структурі даних (табл. 1, кол. 2), яку обрати згідно зі своїм варіантом.
3) Створити екземпляр структури даних і продемонструвати працездатність структури даних: вставити і видалити декілька елементів, вивести вміст структури даних. Звернути увагу на такі тестові випадки:
   - вилучення елементу з порожнього списку, деку або стеку;
   - позиція, куди додається елемент, від'ємна або більше кількості елементів (для списків);
   - позиція, звідки вилучається елемент, від'ємна або більше кількості елементів (для списків);
   - вилучення елементу, якого не міститься у списку

Завдання 3.3. *Перенести (або видалити) елементи з однієї структури даних в іншу відповідно до індивідуальному завдання (табл.3), використовуючи реалізації із завдань 3.1 і 3.2.*

1) створити екземпляри першої структур даних
2) вставити елементи до першої структури даних і вивести її вміст
3) сформувати другу структуру даних згідно із завданням (табл., кол. 3);
4) вивести вміст обох структур даних

Питання для самоперевірки:

1. Лінійна структура даних. Класифікація лінійних структур даних. Основні операції над лінійними структурами даних
2. Організація стеку, реалізованого через масив. Операції додавання і видалення елементів
3. Організація черги, реалізованої через масив. Операції додавання і видалення елементів
4. Організація списку, реалізованого через масив. Операції додавання елемента у список
5. Організація списку, реалізованого через масив. Операції вилучення елемента зі списку
6. Організація двосторонньої черги, реалізованої через масив. Операції додавання і видалення елементів
7. Організація стеку, реалізованого через зв’язаний список. Операції додавання і видалення елементів
8. Організація черги, реалізованої через зв’язаний список. Операції додавання і видалення елементів
9. Організація списку, реалізованого через зв’язаний список. Операції додавання і вилучення елемента зі список
10. Організація двосторонньої черги, реалізованої через зв’язаний список. Операції додавання і вилучення елементів
11. Порівняйте реалізації лінійних структур даних структур та швидкодію основних операцій

Таблиця 1. Варіанти індивідуальних завдань лабораторної роботи 3 (частина 1)

Варіант	Тип структури даних	Тип елементів
1	2	3
1	Стек	Строковий (представляє цілі числа)
2	Черга	Цілий
3	Список	Строковий (представляє цілі додатні числа)
4	Стек	Дійсний
5	Список	Цілий
6	Дек	Цілий
7	Черга	Цілий
8	Список	Цілий
9	Черга	Строковий
10	Список	Цілий
11	Стек	Строковий (представляє цілі додатні числа у вісімковій системі числення)
12	Список	Цілий (додатні числа)
13	Стек	Цілий
14	Черга	Цілий
15	Черга	Дійсний
16	Черга	Дійсний
17	Стек	Строковий (містить літери і цифрові символи)
18	Список	Цілий
19	Стек	Цілий
20	Список	Строковий (представляє цілі додатні числа у шістнадцятковій системі числення)
21	Черга	Цілий (додатні числа)
22	Стек	Дійсний
23	Черга	Цілий
24	Дек	Цілий

Таблиця 2. Варіанти індивідуальних завдань лабораторної роботи 3(частина 2)

Варіант	Тип структури даних	Тип елементів
1	2	3
1	Односпрямований список	Цілий
2	Двоспрямований список	Цілий
3	Стек	Цілий
4	Черга	Строковий (представляє цілі додатні числа у вісімковій системі числення)
5	Двоспрямований список	Строковий (представляє цілі числа)
6	Односпрямований список	Строковий (представляє цілі у двійковій системі числення)
7	Двоспрямований список	Строковий (представляє цілі числа)
8	Стек	Строковий (представляє цілі додатні числа у вісімковій системі числення)
9	Стек	Перелічувальний (представляє жіночі імена на парних позиціях, чоловічі – на непарних)
10	Двоспрямований список	Строковий (представляє цілі додатні числа)
11	Односпрямований список	Цілий
12	Дек	Цілий (додатні числа)
13	Односпрямований список	Строковий (представляє цілі додатні числа у шістнадцятковій системі числення
14	Односпрямований список	Строковий (представляє цілі додатні числа у двійковій системі числення)
15	Стек	Строковий (представляє цілі додатні числа у двійковій системі числення)
16	Двоспрямований список	Строковий (представляє цифрові символи)
17	Дек	Цілий (додатні числа)
18	Черга	Строковий (представляє цілі числа у шістнадцятковій системі числення)
19	Односпрямований список	Строковий (представляє цілі додатні числа)
20	Стек	Цілий
21	Двоспрямований список	Цілий
22	Односпрямований список	Строковий (представляє цілі числа у діапазоні від -10 до 10)
23	Односпрямований список	Цілий
24	Двоспрямований список	Цілий

Таблиця 3. Варіанти індивідуальних завдань лабораторної роботи 3 (частина 3)

Варіант	Перша структура даних	Завдання	Друга структура даних
1	2	3	4
1	Стек (табл.1)	Перемістити елементи стеку до списку у такий спосіб: парні елементи додаються до початку списку, а від'ємні – у кінець.	Список (табл.2)
2	Список (табл.2)	Видалити зі списку додатні елементи та перемістити їх у чергу так: кожен елемент черги обчислюється як сума попереднього елемента черги та щойно видаленого елемента списку.	Черга (табл.1)
3	Список (табл.1)	Видалити зі списку елементи, які представляють непарні цілі числа більше за 50, та перемістити їх у стек.	Стек (табл.2)
4	Черга (табл.2)	Перемістити всі елементи черги у стек так, щоб елементи черги, перетворені в десяткову систему числення і більші за 200, – зменшувалися вдвічі, а менші – подвоювалися.	Стек (табл.1)
5	Список (табл.1)	Видалити зі списку парні числа та вставити їх у другий список у відсортованому за зростанням порядку.	Список (табл.2)
6	Дек (табл.1)	З двох елементів початку і кінці деку видаляти найбільший і додавати його в список	Список (табл.2)
7	Список (табл.2)	Видалити зі списку парні числа та перемістити їх у чергу.	Черга (табл.1)
8	Список (табл.1)	Видалити зі списку від'ємні елементи, а решту перетворити у вісімкову систему та перемістити до стеку.	Стек (табл.2)
9	Стек (табл.2)	Перемістити елементи стеку, які представляють жіночі імена до першої черги, а чолові імена – до другої черги.	Черга (табл.1)
10	Список (табл.1)	Видалити зі списку парні додатні числа та вставити їх у список у відсортованому за спаданням порядку.	Список (табл.2)
11	Список (табл.2)	Видалити зі списку від'ємні елементи, інвертувати їх, перетворити у вісімкову систему числення та перемістити у стек.	Стек (табл.1)
12	Список (табл.1)	Видалити зі списку прості числа і додати до деку у такий спосіб: якщо просте число розташовано у списку парній позиції, додати його у кінець деку, на непарній – на початок.	Дек (табл.2)
13	Стек (табл.1)	Перемістити додатні елементи стеку, перетворені у шістнадцяткову систему числення, до початку списку, а від'ємні елементи стеку інвертувати, перетворити в шістнадцяткову систему числення і перемістити в кінець списку.	Список (табл.2)
14	Список (табл.2)	Видалити зі списку непарні елементи, а решту елементів скопіювати до черги в десятковій системі числення.	Черга (табл.1)
15	Черга (табл.1)	Перемісти до стеку додатні заокруглені елементи черги, перетворені у двійкову систему числення.	Стек (табл.2)
16	Список (табл.2)	Видалити зі списку елементи, які містять непарну кількість символів, а решту елементів використати для формування черги так, щоб елемент черги обчислювався як середнє арифметичне цифр елемента списку.	Черга (табл.1)
17	Список (табл.1)	Видалити зі списку парні елементи, а решту скопіювати до черги у шіснадцятковій системі числення.	Черга (табл.2)

Варіант	Перша структура даних	Завдання	Друга структура даних
1	**2**	**3**	**4**
18	Стек (табл.1)	Перемістити зі стеку у дек лише елементи з цифровими символами, які треба перетворити у ціле число. Якщо число просте, додати його на початок деку, в іншому випадку – у кінець.	Дек (табл.2)
19	Список (табл.2)	Перемістити зі списку в стек елементи, які представляють прості числа.	Стек (табл.1)
20	Список (табл.1)	Перетворити елементи списку в десяткову систему числення та обчислити кожен елемент стеку як суму поточного, попереднього та наступного елементів списку.	Стек (табл.2)
21	Список (табл.2)	Видалити зі списку від'ємні елементи, а решту елементів використати для обчислення елементів черги, які є сумою цифр елемента списку.	Черга (табл.1)
22	Список (табл.2)	Перетворити в цілий тип елементи списку і використати їх як степінь, у яку необхідно піднести число 10, щоб отримати значення елементів стеку.	Стек (табл.1)
23	Черга (табл.1)	Перемістити елементи стеку до списку, вставляючи елементи так, щоб однакові числа розташовувалися в списку один за одним, а перший елемент такої послідовності містив їх кількість.	Список (табл.2)
24	Список (табл.2)	Видалити зі списку елементи, які не входять у діапазон від 3 до 10. Решту елементів використати як кількість елементів у послідовностях Фібоначчі, яким утворюють дек. Якщо елемент списку парний, то послідовність Фібоначі такою кількістю, як елемент, додається у кінець деку, в іншому випадку – на початок. Послідовності чисел Фібоначчі, що утворюють дек, упорядковані у деку за зростанням	Дек (табл.1)