Лабораторна робота 3: Зв'язані списки

Мета роботи:

1) набути практичних навичок зі створення, ініціювання, обходу одно- та двоспрямованих зв'язаних списків на мові програмування Java;
2) опанувати алгоритми додавання і видалення вузлів зв'язаних списків;
3) навчитися писати та викликати статичні методи для роботи зі списками на мові програмування Java;
4) навчитися локалізувати та виправляти помилки часу виконання за допомогою вбудованих засобів відлагодження IDE Eclipse:
5) навчитися аналізувати результати обробки списків на різних наборах тестових даних.

Необхідні теоретичні знання мови Java

1. Визначення статичних методів
2. Виклик методів
3. Перевантаження методів
4. Створення об'єктів
5. Виклик методів для створеного об'єкту

Інструментальні і програмні засоби:

Java JDK 8
IDE Eclipse

Проект з каркасом програми Пакет ***lab_1_3***

Каркас програми: **`Lab3_Main.java`**
Клас для роботи з текстовим файлом: **`FileAssistant.java`**
Текстовий файл: **`input.dat`**

Загальні рекомендації:

Індивідуальні завдання наведені у розділі «Варіанти індивідуальних завдань до лабораторної роботи 3».

Дані для роботи програми – цілі числа. Вони записуються у файл за допомогою будь-якого текстового редактора або засобами IDE Eclipse. Числа розташовуються в один або декілька рядків і розділяються одним або декількома пробілами. Наприклад:

```
56 7 -5 4 2
345 6 7
1
```

0 67 445 3 465

У складі проекту такий файл називається **"input.dat"** і розташований він у директорії **"data"** директорії поточного проекту. Для того, щоб перевірити в IDE Eclipse шлях до поточної директорії, перейдіть до назви проекту у **Package Explorer** та оберіть меню **File -> Properties -> Resources -> поле Location**.

Перед виконанням роботи треба створити директорію **"data"**, створити файл **"input.dat"** і записати до файлу цілі числа. Назву файлу і його розташування можна змінити у цих рядках:

```
private static String fileName = "input.dat";
 private static String currenDir = System.getProperty("user.dir") + File.separatorChar
+ "data";
```

Робота складається з двох завдань, які виконуються послідовно. Метод main() в класі **Lab3_Main** (файл **Lab3_Main.java**) містить послідовність викликів методів для виконання програми. Метод main() змінювати ***не треба***

Програма виконується так. Спочатку читається вміст файлу **"input.dat"**. За умови успішності операції читання розпочинається цикл, в якому із файлу послідовно зчитуються цілі дані, а слова у файлі, які не відповідають формату цілого числа, тобто містять букви або символи, ігноруються. Ознакою завершення читання буде код помилки **FileAssistant.*ERROR_CODE.***
У циклі за допомогою метода **createDLNode** створюється новий вузол двоспрямованого зв'язаного списку. Поле **data** нового вузла зберігатиме ціле число, зчитане з файлу. Новий вузол додається до першого списку методом **addNode**.

Після завершення циклу виводиться розмір першого списку (метод **size**) та дані, які зберігаються у його вузлах (метод **print**).

Потім перший список передається у метод **createSecondList** для того, щоб із даних, що зберігаються у вузлах першого двоспрямованого зв'язаного списку, створити новий односпрямований список. Після такого перетворення виводиться кількість вузлів та їх дані обох списків.

Якщо дані з фала не прочиталися або у файлі не міститься цілих чисел, у консольне вікно виводяться повідомлення, що розмір списків дорівнює 0 і списки порожні.

Клас **Lab3_Main** (файл **Lab3_Main.java**) містить методи, і яких є тег // **TODO**. Це означає, що у цьому місці ви маєте написати власний код. Сигнатуру (тип значення, що повертається, і параметри) існуючих методів змінювати ***не треба***. Але, звичайно, нові методи для реалізації програми можна дописати власноруч. Для нових методів потрібно створити документуючи коментарі з анотаціями @param @return. Щоб ав-

томатично створити згенерувати заготовку коментарів для метода, необхідно стати на заголовок метода та обрати пункт меню **Source → Generate Element Comment**

Опис класів пакету lab3

Клас **SLNode** представляє вузол односпрямованого зв'язаного списку. Містить поля:

data – ціле число, що зберігає вузол
next – посилання на наступний вузол

Клас **DLNode** представляє вузол двоспрямованого зв'язаного списку. Містить поля:

data – ціле число, що зберігає вузол
next – посилання на наступний вузол
prev – посилання на попередній вузол

Клас **Main** є головним класом програми. Містить поля:

fileName – назва файлу, звідки зчитуються дані
currenDir – назва директорії, де знаходиться файл

Метод **main** становить порядок виконання програми. Цей порядок докладно описано у розділі «Загальні рекомендації».

Опис статичних методів класу Main

SLNode createSLNode(int data) – створює новий вузол односпрямованого зв'язаного списку, який зберігає **data** у відповідному полі

DLNode createDLNode(int data) – створює новий вузол двоспрямованого зв'язаного списку, який зберігає **data** у відповідному полі

DLNode addNode(DLNode head, DLNode node) – вставляє новий вузол **node** у двоспрямований зв'язаний список, який передається у метод через його перший вузол **head**. Метод містить тег **TODO**, тобто код методу треба написати згідно із завданням 3.1.

SLNode addNode(SLNode head, DLNode node) – вставляє новий вузол **node** в односпрямований зв'язаний список, який передається у метод через його перший вузол **head**. Метод містить тег **TODO**, тобто код методу треба написати згідно із завданням 3.2. Метод викликається в методі **createSecondList** при створенні другого списку.

void printList(SLNode list) – виводить у стандартний потік виведення (консольне вікно) вміст односпрямованого зв'язаного списку **list**. Метод містить тег **TODO**, тобто код методу треба написати самостійно

`void printList(DLNode list)` – виводить у стандартний потік виведення (консольне вікно) вміст двоспрямованого зв'язаного списку **`list`**. Метод містить тег **`TODO`**, тобто код методу треба написати самостійно

`int size(SLNode list)` – обчислює і повертає кількість вузлів в односпрямованому зв'язаному списку. Метод містить тег **`TODO`**, тобто код методу треба написати самостійно

`int size(DLNode list)` – обчислює і повертає кількість вузлів в двоспрямованому зв'язаному списку. Метод містить тег **`TODO`**, тобто код методу треба написати самостійно

`SLNode createSecondList(DLNode dlHead)` – створює новий односпрямований зв'язаний список, вузли якого містять дані вузлів, видалених з двоспрямованого списку. Метод містить тег **`TODO`**, тобто код методу треба написати згідно із завданням 3.2.

Сигнатуру (параметри і тип значення, що повертається) наведених вище методів змінювати ***не треба***. За потреби в класі можна описати додаткові методи, які викликатимуться з тих, що описані. Таким додатковими методами можуть бути методи перевірки на порожність, пошук вузла з мінімальним значенням, метод видалення вузла і т.і.

***Завдання 3.1**: створити двоспрямований зв'язаний список, додаючи новий вузол у позицію, визначену в пункті a) індивідуального варіанта завдання.*

Для виконання завдання 3.1 слід написати програмний код в методі
`DLNode addNode(DLNode head, DLNode node)`

Якщо у двоспрямовааний список вузол додається у порожній список, він стає першим елементом. В іншому випадку,спочатку треба визначити місце, куди буде додаватися вузол. Таке місце може визначатися або вузлом, за яким треба поставити новий вузол, а перед яким. Додавання нового вузла означає, що в ньому треба змінити поля next і prev.

Працездатність програми перевіряється на різних тестових наборах, які можна перезаписувати в файл **`"input.dat"`**, або створити декілька файлів для кожного з тестових випадків.

***Завдання 3.2**: видалити з двоспрямованого зв'язаного списку,які підпадають під умову, визначену в пункті b) індивідуального варіанта завдання, і створити новий односпрямований список, додаючи новий вузол у позицію, визначену в пункті c) індивідуального варіанта завдання.*

Для виконання завдання 3.2 слід написати програмний код в методі
`SLNode createSecondList(DLNode dlHead)`

Спочатку треба визначити, які вузли двоспрямованого списку підпадають під критерій видалення. Критерій задано в пункті b) індивідуального завдання. Залежно від завдання може організовуватися цикл для обходу списку, або визначатися певна позиція таких вузлів. Знайдені вузли один за одним вилучаються з двоспрямованого списку, а дані поля data кожного вилученого вузла використовуються, щоб створити нові вузли і додати їх в односпрямований список. Код такої операції треба розмістити в методі

`SLNode addNode(SLNode head, DLNode node)`

Місце вставки нового вузла у другий список задано в пункті b) індивідуального завдання.

Оскільки програмний код завдання 3.2 буде громіздким і великим за обсягом, треба написати декілька допоміжних методів. Які саме методи будуть у нагоді залежить від індивідуального завдання

При тестування слід звернути увагу на випадок, коли жоден вузол першого списку не підпадає під критерій видалення. В результаті другий список не може сформуватися. Або випадок, коли з першого списку треба видалити всі вузли і всі вони переміщуються у другий список.

Після виконання завдань треба вивести кількість елементів у списках. Має виконуватися постумова:
{після завдання 3.1} size (List1) = {після завдання 3.2} (size (List1) + size (List2))

Варіанти індивідуальних завдань до лабораторної роботи 3

Варіант 1

a) після вузла з мінімальним значенням
b) вузли на парних позиціях
c) після останнього вузла

Варіант 2

a) перед першим вузлом
b) вузли зі значенням 5
c) початок другої половини списку

Варіант 3

a) після останнього вузла
b) вузли зі значеннями, які не входять до діапазону [-5; 5]
c) парні – перед першим вузлом, непарні – після останнього

Варіант 4

a) після другого вузла
b) вузли зі значеннями, кратним 3
c) середина списку, якщо парна кількість елементів, інакше – на початок списку

Варіант 5

a) перед останнім вузлом
b) вузли зі значеннями, кратним 5
c) після вузла з рівним значенням або на початок списку, якщо вузла з рівним значенням немає у списку

Варіант 6

a) перед першим вузлом
b) вузли з парними значеннями
c) після вузла з найбільшим значенням

Варіант 7

a) після першого елемента
b) вузли з непарними значеннями
c) додатні елементи – перед першим вузлом, від'ємні – після останнього вузла

Варіант 8

a) початок другої половини списку
b) вузли з додатними значеннями
c) перед останнім вузлом

Варіант 9

a) кінець першої половини списку

b) кожен третій вузол
c) після першого елемента

Варіант 10

a) середина списку, якщо парна кількість елементів, інакше – на початок списку
b) вузли другої половини списку
c) перед першим елементом

Варіант 11

a) після останнього вузла
b) вузли другої половини списку
c) після вузла з найменшим значенням

Варіант 12

a) після вузла з найменшим значенням, якщо значення нового вузла більше за нього або перед ним у іншому випадку
b) вузли другої половини списку
c) перед першим вузлом

Варіант 13

a) після останнього вузла
b) вузли з повторюваними значеннями
c) унікальні вузли перед першим вузлом

Варіант 14

a) після першого вузла
b) вузли зі значеннями у діапазоні між найменшим і найбільшим значеннями
c) після останнього вузла

Варіант 15

a) після першого вузла
b) вузли зі значеннями у діапазоні між найменшим і найбільшим значеннями
c) після останнього вузла

Варіант 16

a) після вузла з мінімальним значенням
b) вузли з парними значеннями
c) в кінець першої половини списку

Варіант 17

a) після вузла з мінімальним значенням у першій половині списку
b) вузли на непарних позиціях початкового списку
c) після останнього вузла

Варіант 18

a) після першого елементу
b) вузли з додатними значеннями
c) після вузла з найбільшим значенням

Варіант 19

a) перед вузлом з найбільшим значенням
b) вузли на непарних позиціях початкового списку
c) після останнього вузла першої половини списку

Варіант 20

a) перед останнім вузлом
b) вузли зі значеннями, більше за середнє арифметичне значення елементів
c) перед першим вузлом другої половини списку

Варіант 21

a) після вузла з найбільшим парним значенням
b) вузли на парних позиціях
c) перед першим вузлом

Варіант 22

a) перед першим вузлом
b) вузли зі значенням, меншим 5
c) після вузла з мінімальним значенням

Варіант 23

a) після вузла з останнім додатним значенням
b) вузли зі значеннями у діапазоні [-5; 5]
c) парні – перед першим вузлом, непарні – після останнього

Варіант 24

a) після другого вузла
b) вузли зі значеннями, кратним 3
c) середина списку, якщо парна кількість елементів, інакше – на початок списку

Варіант 25

a) перед останнім вузлом
b) вузли зі значеннями, кратним 3
c) перед вузлом з тим же значенням

Варіант 26

a) перед першим вузлом
b) вузли з парними значеннями
c) після вузла з максимальним значенням