Лабораторна робота №4.

Мета:

- Вивчення принципів параметризації в Java .
- Розробка параметризованих класів та методів.
- Розширення функціональності параметризованих класів.

1. Вимоги

1. Створити власний клас-контейнер, що параметризується (Generic Type), на основі зв'язних списків для реалізації колекції domain-об'єктів з лабораторної роботи №10 (Прикладні задачі. Список №2. 20 варіантів)

2. Для розроблених класів-контейнерів забезпечити можливість використання їх об'єктів у циклі foreach в якості джерела даних.

3. Забезпечити можливість збереження та відновлення колекції об'єктів: 1) за допомогою стандартної серіалізації; 2) не використовуючи протокол серіалізації.

4. Продемонструвати розроблену функціональність: створення контейнера, додавання елементів, видалення елементів, очищення контейнера, перетворення у масив, перетворення у рядок, перевірку на наявність елементів.

5. Забороняється використання контейнерів (колекцій) з Java Collections Framework .

6. Розробити параметризовані методи (Generic Methods) для обробки колекцій об'єктів згідно (Прикладні задачі. Список №2. 20 варіантів).

7. Продемонструвати розроблену функціональність (створення, управління та обробку власних контейнерів) в діалоговому та автоматичному режимах. a. Автоматичний режим виконання програми задається параметром командного рядка -auto . Наприклад, java ClassName -auto . b. В автоматичному режимі діалог з користувачем відсутній, необхідні данні генеруються, або зчитуються з файлу.

Розробник

Левицький Богдан, КН-108

1.1 Задача

Захід: дата, час початку і тривалість; місце проведення; опис; учасники (кількість не обмежена)

2 Опис програми

3 Програма містить інформацію про заходи. Можна додавати, чистити, виписувати, серіалізувати та десеріалізувати ваші заходи у файл вибраний користувачем

3.1 Засоби ООП

Були використані різні класи та методи, структури даних та модифікатори доступу.

3.2 Ієрархія та структура класів

1. Клас `Main`, який викликає всі методи та класи створенні користувачем
2. Клас `Event` - domain-об’єкт
3. Клас `PathMover` повертає шлях збереження файлу та демонструє вміст
4. Клас `Date` під-клас класу
5. Клас `Time` під-клас класу
6. Клас `Members` під-клас класу
7. Клас `LinkedList` спеціально розроблений за для виконання умови «Лабораторна №4». Представляє собою типовий зв’язний список.

3.3 Важливі фрагменти програми.

1) PathMover
2) LinkedList

New from laba3: sort, remove, search,LinkedList added.

PathMover

```java
import java.io.File;

public class PathMover
{
 StringBuilder path;

 public PathMover()
 {
      path = new StringBuilder(System.getProperty("user.dir"));
      path.append("\\");

 }

 private int index()
 {
     for(int i = path.length()-2;i >= 0;i--)
     {
         if(path.charAt(i) == '\\')
             return i;
     }
     return path.length()-1;
 }
private void dir()
{
 File files = new File(path.toString());

 for(File list:  files.listFiles())
 {
  if(list.isFile())
  {
   System.out.println("File: " + list.getName());
  }
  else
   System.out.println("Directory: " + list.getName());
 }
}
```

```
private void cd()
{
    Scanner in = new Scanner(System.in);
    String input;
    while(true)
    {
        System.out.println("$ " + path + "\t\n");
        dir();
        input = in.nextLine();
        if(input.substring(0,3).equals("cd "))
        {
            if(input.substring(3,5).equals(".."))
            {
                path = new StringBuilder(path.substring(0, index()+1));
            }
            else
            {
                System.out.println(input.substring(3,input.length()));
                path.append(input.substring(3,input.length()));
                path.append("\\");
            }

        }
        else
        {
            path.append(input);
            break;
        }
        System.out.println("$ " + path + "\t\n");
    }
}
```

```
public String directory()
{
    cd();
    return path.toString();
}
```

LinkedList with Node

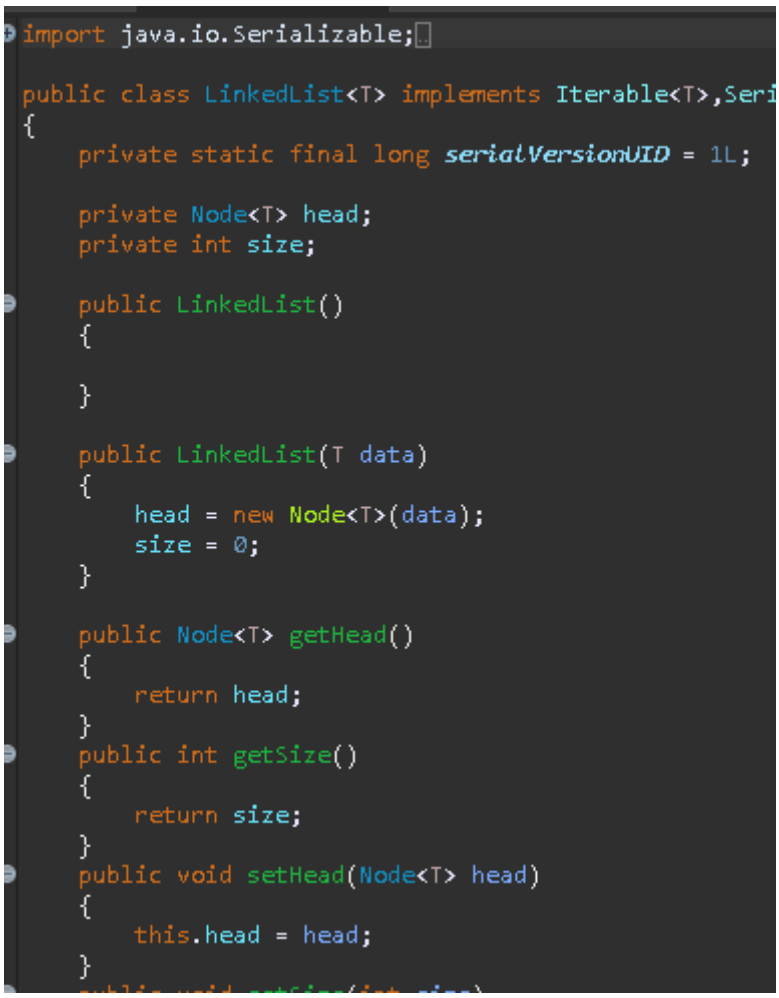
```java
public class Node<T> implements Serializable
    {
         /**
     *
     */
    private static final long serialVersionUID = 1L;

        private T data;
        private Node<T> next;

        public Node()
        {

        }

        public Node(T data)
        {
            this.data = data;
        }
        public Node<T> getNext()
        {
            return next;
        }
        public T getData()
        {
            return data;
        }
        public void setNext(Node<T> next)
        {
            this.next = next;
        }
        public void setData(T data)
        {
            this.data = data;
```



```java
import java.io.Serializable;

public class LinkedList<T> implements Iterable<T>,Seri
{
    private static final long serialVersionUID = 1L;

    private Node<T> head;
    private int size;

    public LinkedList()
    {

    }

    public LinkedList(T data)
    {
        head = new Node<T>(data);
        size = 0;
    }

    public Node<T> getHead()
    {
        return head;
    }
    public int getSize()
    {
        return size;
    }
    public void setHead(Node<T> head)
    {
        this.head = head;
    }
```

```java
public void setSize(int size)
{
    this.size = size;
}

public void addFirst(T data)
{
    size++;
    if(head == null)
    {
        head = new Node<T>(data);
        return;
    }
    Node<T> tmp = new Node<T>(data);
    tmp.setNext(head);
    head = tmp;
}

public void addLast(T data)
{
    size++;
    Node<T> tmp = new Node<T>(data);
    if(head == null)
    {
        head = tmp;
        return;
    }
    Node<T> current = head;
    while(current.getNext()!= null)
    {
        current = current.getNext();
    }
    current.setNext(tmp);
}
```

```java
    public void clear()
    {
        head = null;
        size = 0;
    }

    public T[] toArray()
    {
        T[] arr = (T[]) new Object[size];
        int count = 0;
        Node<T> current = head;
        while(current != null)
        {
            arr[count++] = current.getData();
            current = current.getNext();
        }
        return arr;
    }

    public String toString()
    {
        StringBuilder res = new StringBuilder();
        Node<T> current = head;
        while(current != null)
        {
            res.append(current.getData().toString());
            res.append(" ");
            current = current.getNext();
        }
        return res.toString();
    }

public boolean isClear()
{
    return head == null;
}

public T getElement(int index)
{
    if(head == null || index >= size || index < 0)
        return null;
    Node<T> current = head;
    int count = 0;
    while(current != null && count++ < index)
    {
        current = current.getNext();
    }
    if(current == null)
        return null;
    return current.getData();
}
```

```java
public boolean remove(int index)
{
    if(index >= size || index < 0)
        return false;
    Node<T> current = head;
    int count = 0;
    while(count < index-1)
    {
        current = current.getNext();
        count++;
    }
    if(current == null)
        return false;
    if(index == 0)
    {
        head = current.getNext();
        size--;
        return true;
    }
    current.setNext(current.getNext().getNext());
    size--;
    return true;
}
```

```java
@Override
public Iterator iterator() {
    return new Iterator()
            {
                int count = 0;
                @Override
                public boolean hasNext() {
                    return count < size;
                }

                @Override
                public Object next() {
                    if(!hasNext())
                    {
                        throw new NoSuchElementException();
                    }
                    return getElement(count++);
                }

            };
}
}
```

8. Варіанти використання

Можна використовувати для зберігання важливих заходів

ВИСНОВКИ

У ході роботи ми вивчили принципи параметризації в Java та розробили параметризований LinkedList, також розширили його функціональність