Лабораторна робота №9

Параметризація в Java

Мета:

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

1 Вимоги

- Створити власний клас-контейнер, що параметризується (Generic Type), на основі зв’язних списків для реалізації колекції domain-об’єктів лабораторної роботи №7.

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

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

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

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

1.1 Розробник

Інформація про розробника:

• Безпалий М.Л.;

• 1КІТ118г;

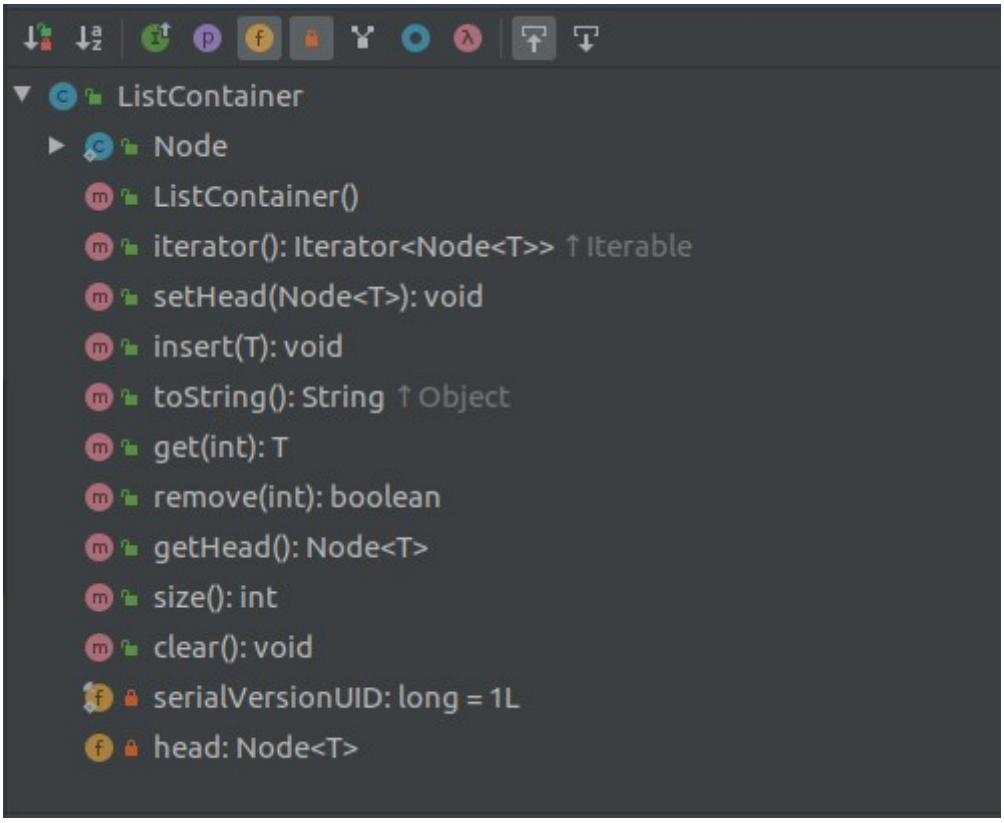
• варіант No3.

ОПИС ПРОГРАМИ

Дана програма містить два класа з методами.

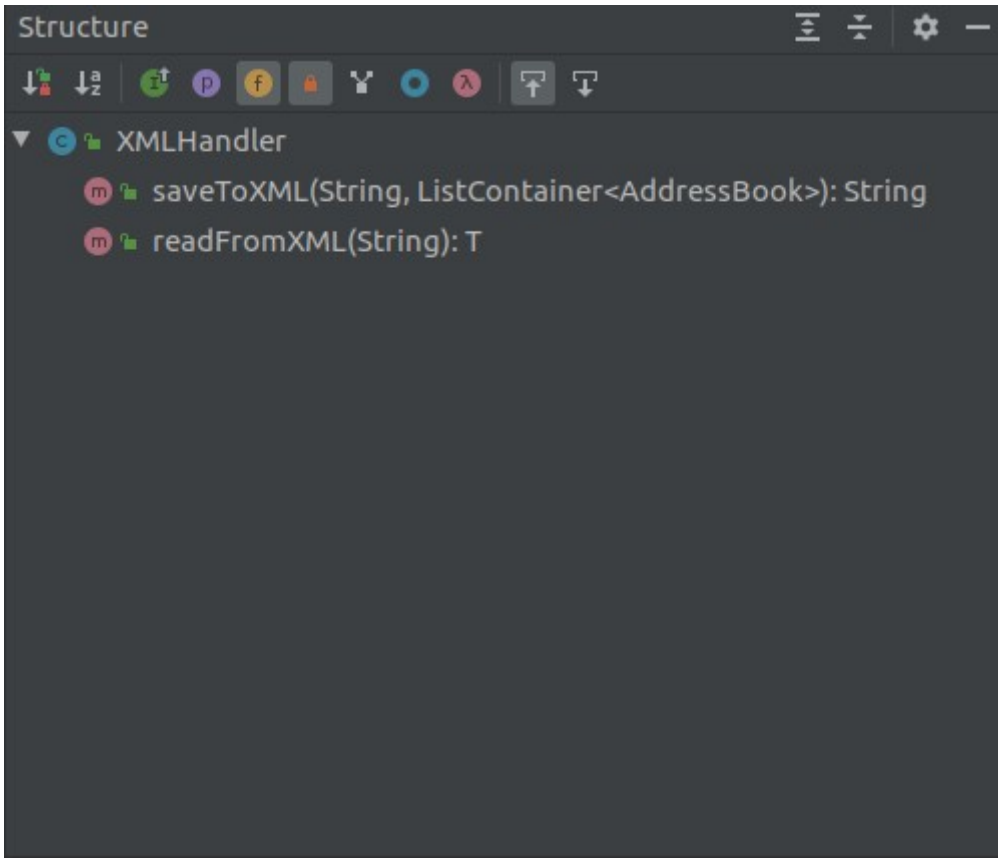
Клас **ListContainer**

Реалізує всі потрібні методи контейнеру та основан на зв’язних списках



Клас **XMLHandler**

Реалізує методи XML сереалізації та десереалізациї.



ТЕКСТ ПРОГРАМИ

ListContainer.Class

```
package ua.khpi.oop.bezpalyi09;
import java.io.Serializable;
import java.util.Iterator;

public class ListContainer<T> implements Serializable, Iterable<ListContainer.Node<T>>
{
   /**
    *
    */
   private static final long serialVersionUID = 1L;

   @Override
   public Iterator<Node<T>> iterator() {
      return new Iterator<>() {

         private Node<T> tempNode = head;

         @Override
         public boolean hasNext() {
            return tempNode.next != null;
         }

         @Override
         public Node<T> next() {
            Node<T> next = tempNode;
            tempNode = tempNode.next;
            return next;
          }

      };
   }

   private Node<T> head;

   public ListContainer() {
      head = null;
   }

   public void setHead(Node<T> head) {
      this.head = head;
   }

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

      T value;
      Node<T> next;

      public T getValue() {
         return value;
      }
```

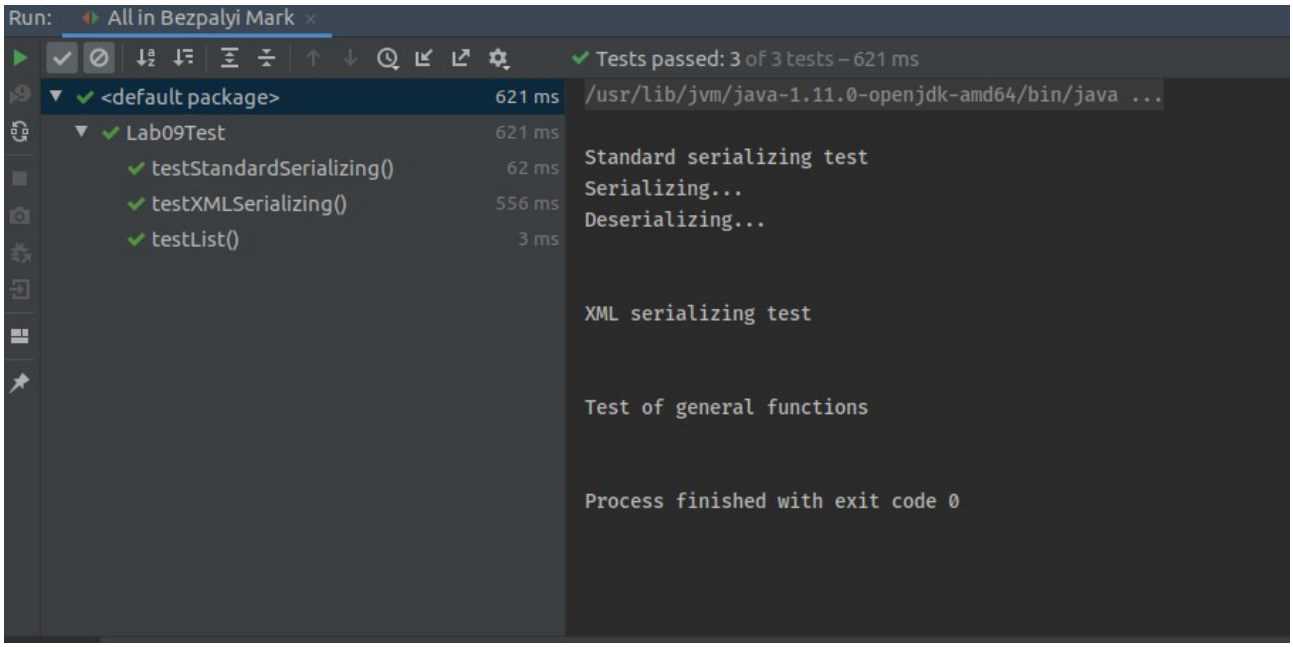
```
    public void setValue(T value) {
        this.value = value;
    }

    public Node<T> getNext() {
        return next;
    }

    public void setNext(Node<T> next) {
        this.next = next;
    }

    public Node() {

    }

    public Node(T book) {
        value = book;
        next = null;
    }
}

public void insert(T value) {
    Node<T> newNode = new Node<>(value);
    newNode.next = null;

    if (head == null) {
        head = newNode;
    } else {
        Node<T> endNode = head;
        while (endNode.next != null) {
            endNode = endNode.next;
        }
        endNode.next = newNode;
    }
}

public String toString() {
    Node<T> tempNode = head;
    if (head == null) {
        return "";
    } else {
        final int capacyti = 50000;
        StringBuilder stringBuilder = new StringBuilder(capacyti);
        while (tempNode != null) {
            stringBuilder.append(tempNode.value.toString()).append('\n');
            tempNode = tempNode.next;
        }
        return stringBuilder.toString();
    }
}

public T get(int index) {
    Node<T> tempNode = head;
    if (head == null) {
        return null;
    } else {
        int length = 0;
```

```
            while (tempNode.next != null && length != index) {
                tempNode = tempNode.next;
                length++;
            }

            if (length == index) {
                return tempNode.value;
            } else {
                return null;
            }
        }
    }

    public boolean remove(int index) {
        if (index < 0) {
            return false;
        } else if (index == 0) {
            head = head.next;
            return true;
        }
        Node<T> prevNode = head;
        Node<T> tempNode = head;
        int counter = 0;
        while (tempNode != null && counter != index) {
            counter++;
            prevNode = tempNode;
            tempNode = tempNode.next;
        }

        if (tempNode == null) {
            return false;
        } else if (tempNode.next == null) {
            prevNode.next = null;
            return true;
        }

        prevNode.next = tempNode.next;
        return true;
    }

    public Node<T> getHead() {
        return head;
    }

    public int size() {
        Node<T> tempNode = head;
        int size = 0;
        while(tempNode != null) {
            size++;
            tempNode = tempNode.next;
        }
        return size;
    }
    public void clear() {
        head = null;
    }

}
```

XMLHandler.Class

```
package ua.khpi.oop.bezpalyi09;

import ua.khpi.oop.bezpalyi07.AddressBook;

import java.beans.XMLDecoder;
import java.beans.XMLEncoder;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class XMLHandler<T extends ListContainer<AddressBook>> {

    public String saveToXML(String path, ListContainer<AddressBook> obj) {
        try {
            Date date = new Date();
            SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd");
            String dateInString = simpleDateFormat.format(date);
            FileOutputStream fos = new FileOutputStream(new File(path + dateInString +
".xml"));
            XMLEncoder xmlEncoder = new XMLEncoder(fos);
            xmlEncoder.writeObject(obj);
            xmlEncoder.close();
            fos.close();
            return dateInString;
        } catch (IOException e) {
            e.printStackTrace();
            return "";
        }
    }

    public T readFromXML(String path) {
        try {
            FileInputStream fis = new FileInputStream(new File(path));
            XMLDecoder xmlDecoder = new XMLDecoder(fis);
            T value = (T) xmlDecoder.readObject();
            xmlDecoder.close();
            fis.close();
            return value;
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }
    }

}
```

Результати роботи програми:



Висновок: під час виконання даної лабораторної роботи було набуто навичок тривалого зберігання та відновлення стану об’єктів, я ознайомився з принципами серіалізації/десеріалізації об’єктів та використанням бібліотек класів користувача. Набув навичок роботи з дженеріками та зв’язними списками в джава.