Лабораторна робота №9

Виконав:

ст. групи КН-107

Древницький Ю.А.

Прийняв :

Асистент

кафедри СШІ

Швороб І.Б.

Обов’язковий розділ - Зауваження

Тема:

Розробка власних контейнерів. Ітератори. Серіалізація/десеріалізація об’єктів. Бібліотека класів користувача

Мета:

- Набуття навичок розробки власних контейнерів.

- Використання ітераторів.

- Тривале зберігання та відновлення стану об’єктів.

- Ознайомлення з принципами серіалізації/десеріалізації об’єктів.

- Використання бібліотек класів користувача.

Вимоги:

1. Розробити клас-контейнер, що ітерується (docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html) для збереження початкових даних Вашого варіанту завдання з роботи №8 (Прикладні задачі. Список з 1-15 варіантів) у вигляді масиву рядків з можливістю додавання, видалення і зміни елементів.2. В контейнері реалізувати та продемонструвати наступні методи:

○ String toString() повертає вміст контейнера у вигляді рядка;

○ void add(String string) додає вказаний елемент до кінця контейнеру;

○ void clear() видаляє всі елементи з контейнеру;

○ boolean remove(String string) видаляє перший випадок вказаного елемента з контейнера;

○ Object[] toArray() повертає масив, що містить всі елементи у контейнері;

○ int size() повертає кількість елементів у контейнері;

○ boolean contains(String string) повертає true , якщо контейнер містить вказаний елемент;

○ boolean containsAll(Container container) повертає true , якщо контейнер містить всі елементи з зазначеного у параметрах;

○ public Iterator<String> iterator() повертає ітератор відповідно до Interface Iterable .

http://docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html

3. В класі ітератора відповідно до Interface Iterator (http://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html) реалізувати методи:

○ public boolean hasNext() ;

○ public String next() ;

○ public void remove() .

4. Продемонструвати роботу ітератора за допомогою циклів while и for each .

5. Забороняється використання контейнерів (колекцій) і алгоритмів з Java Collections Framework - https://docs.oracle.com/javase/8/docs/technotes/guides/collections/

6. Реалізувати і продемонструвати тривале зберігання/відновлення розробленого контейнера за допомогою серіалізації/десеріалізації .

https://docs.oracle.com/javase/8/docs/technotes/guides/serialization/index.html

7. Обмінятися відкомпільованим (без початкового коду) службовим класом (Utility Class) рішення одного варіанту задачі (Прикладні задачі. Список з 1-15 варіантів) з сусіднім номером. 1 міняється з 2, 2 з 3, 3 з 4, 4 з 5 і т.д. Останній, 15 міняється з 1 варіантом і далі аналогічно.

8. Продемонструвати послідовну та вибіркову обробку елементів

розробленого контейнера за допомогою власного і отриманого за

обміном службового класу.9. Реалізувати та продемонструвати порівняння, сортування та пошук

елементів у контейнері.

10. Розробити консольну програму та забезпечити діалоговий режим роботи

з користувачем для демонстрації та тестування рішення.

Розробник:

Студент академічної групи КН-107

Древницький Юрій Анатолійович

Варіант 18

Розв’язок:

```
package lab_9_11;

import java.io.Serializable;
import java.util.Iterator;

import static java.lang.String.valueOf;

public class MyList<E> implements LinkedReal<E>,Iterable<E>,DescIter<E>,Serializable {

   private int size = 0;
   private Node<E> firstNode;
   private Node<E> lastNode;

   public MyList()
   {
       lastNode = new Node<>(null,firstNode,null);
       firstNode = new Node<>(null,null,lastNode);
   }

   @Override
   public void addFirst(E e) {
      Node<E> next = firstNode;
      next.setCurrentElement(e);
      firstNode = new Node<>(null,null,next);
```

```
    next.setPrevElement(firstNode);
    size++;

}

@Override
public void addLast(E e) {
    Node<E> prev = lastNode;
    prev.setCurrentElement(e);
    lastNode = new Node<>(null,prev,null);
    prev.setNextElement(lastNode);
    size++;

}

@Override
public void delEl(int counter) {
    Node<E> target = firstNode.getNextElement();
    Node<E> preTarget = firstNode.getNextElement();
    Node<E> nextTarget = firstNode.getNextElement();

    //finding
    for (int i = 0; i < counter; i++)
    {
        target = getElement(target);
    }
    preTarget = target.getPrevElement();
    nextTarget = target.getNextElement();

    //deleting
    preTarget.setNextElement(nextTarget);
    nextTarget.setPrevElement(preTarget);

    size--;
}

@Override
public int getSize() {
    return size;
}


@Override
public E getElementByIndex(int counter) {
    Node<E> target = firstNode.getNextElement();

    for (int i = 0; i < counter; i++)
    {
        target = getElement(target);
```

```
    }

    return target.getCurrentElement();
}
private Node<E> getElement(Node<E> current)
{

    return current.getNextElement();
}


@Override
public String toStringg(MyList listt) {

    String sum = "";

    for (int i=0; i<size; i++)
    {
        sum = sum.concat(valueOf(listt.getElementByIndex(i)));
    }

    return sum;
}

@Override
public void clear(MyList listt) {

    firstNode.nextElement = lastNode;
    lastNode.prevElement = firstNode;

    size = 0;
}

@Override
public Object[] toArray() {

    Node<E> target = firstNode.getNextElement();

    Object[] listElements = new Object[size];
    for (int i = 0; i < size; i++)
    {
        listElements[i] = target.getCurrentElement();
        target = getElement(target);
    }

    return listElements;
}

@Override
```

```
public boolean contains(String string) {

   Node<E> target = firstNode.getNextElement();

   for (int i = 0; i < size; i++)
   {
     if ( valueOf(target.getCurrentElement()).equals(string) )
     {
        return true;
     }

      target = getElement(target);
   }

   return false;
}

@Override
public boolean containsAll(MyList listt) {

   Node<E> target = firstNode.getNextElement();

   for (int i = 0; i < size; i++)
   {
      if ( !target.getCurrentElement().equals(listt.getElementByIndex(i)) )
      {
         return false;
      }

      target = getElement(target);
   }


   return true;
}


@Override
public Iterator<E> iterator() {
   return new Iterator<E>() {
      int counter = 0;
      @Override
      public boolean hasNext() {
         return (counter<size);
      }

      @Override
      public E next() {
         return getElementByIndex(counter++);
```

```
        }

    };
}

@Override
public Iterator<E> descendingIterator() {
    return new Iterator<E>() {
        int counter = size -1;

        @Override
        public boolean hasNext() {
            return (counter >= 0);
        }

        @Override
        public E next() {
            return getElementByIndex(counter--);
        }
    };
}


/**
 * NODE
 */
private class Node<E> implements Serializable {

    private E currentElement;
    private Node<E> nextElement;
    private Node<E> prevElement;

    private Node(E currentElement, Node<E> prevElement, Node<E> nextElement) {
        this.currentElement = currentElement;
        this.prevElement = prevElement;
        this.nextElement = nextElement;

    }


    /**
     * getters
     */
    public E getCurrentElement() {
        return currentElement;
    }

    public Node<E> getPrevElement() {
        return prevElement;
```

```
        }

        public Node<E> getNextElement() {
            return nextElement;
        }

        /**
         * setters
         */
        public void setCurrentElement(E currentElement) {
            this.currentElement = currentElement;
        }

        public void setPrevElement(Node<E> prevElement) {
            this.prevElement = prevElement;
        }

        public void setNextElement(Node<E> nextElement) {
            this.nextElement = nextElement;
        }
    }

}
```

```
package lab_9_11;

public interface LinkedReal<E> {

    void  addFirst(E e);
    void  addLast(E e);
    void delEl(int counter);
    int getSize();
    E getElementByIndex(int counter);

    // lab 9 specials

    String toStringg(MyList myList);
    void clear(MyList myList);
    Object[] toArray();
    boolean contains(String string);
    boolean containsAll(MyList myList);
}
```

```
package lab_9_11;

import java.util.Iterator;

public interface DescIter<E> {

   public Iterator<E> descendingIterator();



}


package com.serialization;

import company.*;
import javax.swing.*;
import java.io.*;
import java.util.ArrayList;


public class Serial {

   public static void main  (String[] args) {

   }


   public static Object deserialize(String filename) {

      Object returnObject = null;

      try {
         FileInputStream fileInputStream = new FileInputStream(filename + ".txt"); // ridding file
         ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream); // reduce bytes to
object type, need casting

         returnObject = objectInputStream.readObject();

         fileInputStream.close();
         objectInputStream.close();

      }catch (FileNotFoundException f)
      {
         System.out.println("file not found");
         System.exit(1);
      } catch (IOException e) {
         System.out.println("IOException");
```

```
            System.exit(2);
        } catch (ClassNotFoundException e) {
            System.out.println("class not fount");
            System.exit(3);
        }

        return returnObject;
    }

    public static void serialize(String filename, Listt object) {

        try {
            FileOutputStream fileOutputStream = new FileOutputStream(filename+".txt");   // ridding file
            ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream);   //
making appropriate byte code

            objectOutputStream.writeObject(object);

            fileOutputStream.close();
            objectOutputStream.close();

        }catch (FileNotFoundException f)
        {
            System.out.println("file not found");
            System.exit(1);
        } catch (IOException e) {
            System.out.println("IOException "+e.fillInStackTrace());
            System.exit(2);
        }

    }

}

package com.serialization;

import java.io.Serializable;

public class Profile implements Serializable {

        // parameters
    private String name;
    private String surname;
    private String age;



        //getters and setters
    public String getSurname() {
        return surname;
```

```
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }


    public String getAge() {
        return age;
    }

    public void setAge(String age) {
        this.age = age;
    }
}
```

Висновок:

Отже, я навчився створювати власний контейнер, що ітерується, зберігати об’єкти в ньому, ознайомився з принципами серіалізації.