Міністерство освіти і науки України

Національний технічний університет

«Харківський політехнічний інститут»

Кафедра «Обчислювальна техніка та програмування»

ЗВІТ

Про виконання лабораторної роботи №9

“Параметризація в Java”

Виконавець:

студент гр. КН-920В

Олексієнко Микита

Харків 2022

Лабораторна робота №9.

**Параметризація в Java**

**Мета**

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

**1.Вимоги**

**1.1 Розробник:**

Олексієнко Микита Віталійович

студент групи КН-920В

1. варіант

**1.2 Загальне завдання**

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

**1.3 Прикладні задачі**

Автомобіль: марка, рік випуску, технічні характеристики у вигляді "параметр - значення" (серед яких витрата палива в міському і заміському циклі), технічний стан, ціна.

**2. Опис програми**

**2.1 Засоби ООП:**

У лабороторній роботі 9 , я використовую клас Program, який має у собі лише запуск меню, з лабороторної роботи 8. Ця работа передбачає розроблення зв’язного списку для його викорастання у власному контейнері.

Попередньо оголошено клас Auto, який має поля з індивідуального завдання та методи обробки ціх полей та клас Menu для роботи з користувачем.

**2.2 Ієрархія та структура класів:**

* Auto.java
* Container.java
* Iterator.java ( interface )
* LinkedList.java
* Menu.java
* Program.java

**2.3 Важливі фрагменти програми:**

LinkedList.java :

package ua.khpi.oop.alekseenko09;  
  
public class LinkedList<T> {  
 private Node head;  
 private int size;  
  
  
 private static class Node <T> {  
 private T data;  
 Node next;  
  
 // Constructor  
 Node ( T data ) {  
 this.data = data;  
 this.next = null;  
 }  
 }  
  
 public void push( T data ) {  
 // Create a new node with given data  
 Node new\_node = new Node(data);  
 new\_node.next = null;  
  
 // If the Linked List is empty,  
 // then make the new node as head  
 if (this.head == null) {  
 this.head = new\_node;  
 }  
 else {  
 // Else traverse till the last node  
 // and insert the new\_node there  
 Node last = this.head;  
 while (last.next != null) {  
 last = last.next;  
 }  
  
 // Insert the new\_node at last node  
 last.next = new\_node;  
 }  
  
 this.size++;  
 }  
  
 public void remove( T data ) {  
 if ( this.head == null ) {  
 System.err.println("Error: Container is empty! Nothing to delete!");  
 return;  
 } else {  
 Node node = this.head;  
  
 if ( node.data == data ) {  
 this.head = node.next;  
 this.size--;  
 return;  
 }  
  
 while ( node.next != null ) {  
  
 if ( node.next.data == data) {  
  
 if ( node.next.next == null ) {  
 node.next = null;  
 } else {  
 node.next = node.next.next;  
 }  
  
 this.size--;  
 break;  
 }  
  
 node = node.next;  
 }  
 }  
 }  
  
 public void removeByPos( int number ) {  
 if ( this.head == null ) {  
 System.err.println("Error: Container is empty! Nothing to delete!");  
 return;  
 } else {  
 Node node = this.head;  
 int count = 0;  
 number--;  
  
 while ( node.next != null ) {  
 if ( number == 0 ) {  
 this.head = node.next;  
 } else if ( count == number ) {  
 if ( node.next.next == null ) {  
 node.next = null;  
 break;  
 } else {  
 node.next = node.next.next;  
 }  
 }  
  
 count++;  
 node = node.next;  
 }  
 }  
  
 this.size--;  
 }  
  
 public void clear() {  
 // Delete chaining with making head as null  
 this.head = null;  
 this.size = 0;  
 }  
  
 public Object[] toArray() {  
 Object[] result = new Object[this.size];  
  
 Node currNode = this.head;  
  
 for ( int i = 0; currNode != null; i++ ) {  
  
 result[i] = currNode.data;  
 // Go to next node  
 currNode = currNode.next;  
 }  
  
 return result;  
 }  
  
 public void fromArray( Object[] o ) {  
 this.clear();  
  
 for (Object value : o) {  
 this.push((T) value);  
 }  
 }  
  
 @Override  
 public String toString() {  
 Node currNode = this.head;  
  
 StringBuilder result = new StringBuilder();  
  
 while ( currNode != null ) {  
  
 result.append(currNode.data);  
 // Go to next node  
 currNode = currNode.next;  
 }  
  
 return result.toString();  
 }  
  
 public boolean contains( T data ) {  
 Node currNode = this.head;  
  
 while ( currNode != null ) {  
  
 if ( currNode.data == data ) {  
 return true;  
 }  
  
 // Go to next node  
 currNode = currNode.next;  
 }  
  
 return false;  
 }  
  
 public boolean isEmpty() {  
 return size == 0;  
 }  
  
 // Method to print the LinkedList.  
 public void printList() {  
 Node currNode = this.head;  
  
 // Check chain for empty  
 if ( currNode == null ) {  
 System.out.println("List is empty!");  
 } else {  
 System.out.println("Cars List:");  
  
 // Traverse through the LinkedList  
 while (currNode != null) {  
 // Print the data at current node  
 System.out.println(currNode.data + " ");  
  
 // Go to next node  
 currNode = currNode.next;  
 }  
  
 System.out.println();  
 }  
 }  
  
 public int getSize () {  
 return this.size;  
 }  
// public void setSize ( int number ) { this.size = number; }  
  
}

Container.java :

package ua.khpi.oop.alekseenko09;  
  
import java.beans.XMLDecoder;  
import java.beans.XMLEncoder;  
import java.io.\*;  
import java.util.Iterator;  
  
  
public class Container implements Iterable<Auto> {  
 private LinkedList<Auto> container = new LinkedList();  
// private Iterator<Auto> iterator = container.iterator();  
  
 // добавление машины в гараж  
 public void add( Auto data ) {  
 container.push(data);  
 }  
  
 // удаление машины из гаража если такая переданная машина имеется в списке  
 public void remove( Auto data ) {  
 container.remove(data);  
 }  
  
 public void removeByPos( int data ) {  
 container.removeByPos(data);  
 }  
  
 // очищение списка  
 public void clear() {  
 container.clear();  
 }  
  
 // список в массив  
 public Object[] toArray() {  
 return container.toArray();  
 }  
  
 // список в строку  
 public String toString() {  
 return container.toString();  
 }  
  
 // проверка на наличие объекта в списке  
 public boolean Contains( Auto i ) {  
 return container.contains(i);  
 }  
  
 public void carList() {  
 container.printList();  
 }  
  
 public void xml\_save ( String path ) {  
 System.out.println("SAVING...");  
 try (XMLEncoder xmlEncoder = new XMLEncoder(new FileOutputStream(path))) {  
 xmlEncoder.writeObject( container.toArray() );  
 xmlEncoder.flush();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 System.out.println("SAVING SUCCESSFULLY ENDED\n");  
 }  
  
 public void xml\_load ( String path ) {  
 System.out.println("LOADING...");  
 try (XMLDecoder xmlDecoder = new XMLDecoder(new FileInputStream(path))) {  
 container.fromArray((Object[]) xmlDecoder.readObject());  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 System.out.println("LOADING SUCCESSFULLY ENDED\n");  
 }  
  
 public void standard\_save(String path ) {  
 System.out.println("SAVING...");  
 try {  
 FileOutputStream fileOutputStream = new FileOutputStream( path );  
 ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream);  
 objectOutputStream.writeObject(container.toArray());  
 objectOutputStream.close();  
 } catch (Exception e) {  
 System.err.println(e);  
 }  
 System.out.println("SAVING SUCCESSFULLY ENDED\n");  
 }  
  
 public void standard\_load ( String path ) {  
 try {  
 System.out.println("LOADING...");  
 FileInputStream fileInputStream = new FileInputStream( path );  
 ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);  
 container.fromArray((Object[]) objectInputStream.readObject());  
 } catch (Exception e) {  
 System.err.println(e);  
 }  
 System.out.println("LOADING SUCCESSFULLY ENDED\n");  
 }  
  
 public Iterator<Auto> iterator() {  
 return new ListIterator();  
 };  
  
 public class ListIterator implements Iterator<Auto> {  
 private int index;  
 private Object[] object = container.toArray();  
  
 @Override  
 public boolean hasNext() {  
 return index < container.getSize();  
 }  
  
 public Auto next() {  
 if ( hasNext() ) {  
 return (Auto) object[index++];  
 }  
 return null;  
 }  
  
 public void remove() {  
 removeByPos(index);  
 }  
 }  
}

**3. Висновок:**

Вивчено принципі параметризації в Java. Розроблено параметризованих класів та методів.