Міністерство освіти і науки України

Національний Університет «Львівська Політехніка»

Кафедра АСУ

Звіт до лабораторної роботи №4

з курсу: **«Інноваційні інформаційні технології»**

на тему:

**«Похідні класи. Об’єктно орієнтований зв’язний список в Java.»**

**Виконав:**

студент групи КНМ-12

Новосад Б. І.

**Перевірив:**

старший викладач

Ковальчук А.М.

Львів 2015

**Мета** : На прикладі об'єктно-орієн­то­ва­ного зв'язаного спи­ску закріплення знання теорії наслідування в Java.

Вибрати два завдання з номерами: ***n*1** = порядковому номеру студента в групі, ***n*2**= **⏐*N* + 2 - 2*n*1⏐**, де ***N*** **-** число студентів в групі.

***n*1 =** **18**

***n*2 = ⏐*29* + 2 – 2\**18*⏐** = 5

**Завдання 5 та 18.**

* Описати клас, що реалізує стек. Написати програму, що використовує цей клас для моделювання Т-образного сортувального вузла на залізниці. Програма повинна розділяти на два напрямки склад, що складається з вагонів двох типів (на кожен напрямок формується склад з вагонів одного типу). Передбачити можливість формування складу з файлу і з клавіатури.
* Заданий список iз *n* дiйсних чисел. Написати програму, яка повертає значення true в тому випадку, якщо числа у списку розмiщенi в порядку зростання.

Хід роботи

**Клас Stack**

*/\*\*  
 \* Created by Bogdan on 16.10.2016.  
 \*/***class** Stack<E> {  
 **private final int size**;  
  
 **private int top**;  
  
 **private** E[] **elements**;  
  
 **public** Stack() {  
 **this**(10);  
 }  
  
 **public** Stack(**int** s) {  
 **size** = s > 0 ? s : 10;  
 **top** = -1;  
  
 **elements** = (E[]) **new** Object[**size**]; *// create array* }  
  
 **public void** push(E pushValue) {  
 **if** (**top** == **size** - 1) *// if stack is full* **throw new** FullStackException(String.*format*(**"Stack is full, cannot push %s"**, pushValue));  
  
 **elements**[++**top**] = pushValue; *// place pushValue on Stack* }  
  
 **public** E pop() {  
 **if** (**top** == -1) *// if stack is empty* **throw new** EmptyStackException(**"Stack is empty, cannot pop"**);  
  
 **return elements**[**top**--]; *// remove and return top element of Stack* }  
}  
  
**class** EmptyStackException **extends** RuntimeException {  
 **public** EmptyStackException() {  
 **this**(**"Stack is empty"**);  
 }  
  
 **public** EmptyStackException(String exception) {  
 **super**(exception);  
 }  
}  
  
**class** FullStackException **extends** RuntimeException {  
 **public** FullStackException() {  
 **this**(**"Stack is full"**);  
 }  
  
 **public** FullStackException(String exception) {  
 **super**(exception);  
 }  
}

**Клас Carload**

**package** com.company;  
  
**import** java.util.Random;  
  
*/\*\*  
 \* Created by Bogdan on 16.10.2016.  
 \*/***public class** Carload {  
 @Override  
 **public** String toString() {  
 **return "Carload{"** +  
 **"typeCarLoad="** + **typeCarLoad** +  
 **'}'**;  
 }  
  
 TypeCarload **typeCarLoad**;  
 **public** Carload(TypeCarload typeCarLoad) {  
 **this**.**typeCarLoad** = typeCarLoad;  
 }  
  
 **public static** Carload getNewRandomCarload(){  
 Random random = **new** Random();  
 **int** a = random.nextInt(100);  
 **if**(a>60){  
 **return new** Carload(TypeCarload.***RED***);  
 }  
 **else  
 return new** Carload(TypeCarload.***GREEN***);  
 }  
  
}

**package** com.company;  
  
*/\*\*  
 \* Created by Bogdan on 16.10.2016.  
 \*/***public enum** TypeCarload {  
 RED(0),  
 GREEN(1);  
  
 **int type**;  
 TypeCarload(**int** i) {  
 **type** = i;  
 }  
}

**Клас Zaliznuca**

**package** com.company;  
  
**import** java.util.Stack;  
  
*/\*\*  
 \* Created by Bogdan on 16.10.2016.  
 \*/***public class** Zaliznuca {  
 **private** Stack<Carload> **redCarload** = **new** Stack<>();  
 **private** Stack<Carload> **greenCarload** = **new** Stack<>();  
  
 @Override  
 **public** String toString() {  
 **return "Zaliznuca{"** +  
 **"redCarload="** + **redCarload** +  
 **", greenCarload="** + **greenCarload** +  
 **'}'**;  
 }  
  
 **public** Zaliznuca() {  
  
 }  
  
 **public void** pushCarload(Carload carload){  
 **switch** (carload.**typeCarLoad**.**type**){  
 **case** 0:  
 **redCarload**.push(carload);  
 **break**;  
 **case** 1:  
 **greenCarload**.push(carload);  
 **break**;  
 }  
 }  
 **public void** getStatus(){  
 System.***out***.print(**this**);  
 }  
}

**Використання класів.**

**package** com.company;  
  
**import** java.io.IOException;  
**import** java.util.Stack;  
  
**public class** Main {  
  
 **public static void** main(String[] args) {  
 *// write your code here* Zaliznuca zaliznuca = **new** Zaliznuca();  
  
 **for**(**int** i=0;i<10;i++){  
 zaliznuca.pushCarload(Carload.*getNewRandomCarload*());  
 zaliznuca.getStatus();  
 System.***out***.println();  
 }  
 }  
}

**Результат роботи програми**

Zaliznuca{redCarload=[], greenCarload=[Carload{typeCarLoad=GREEN}]}

Zaliznuca{redCarload=[Carload{typeCarLoad=RED}], greenCarload=[Carload{typeCarLoad=GREEN}]}

Zaliznuca{redCarload=[Carload{typeCarLoad=RED}, Carload{typeCarLoad=RED}], greenCarload=[Carload{typeCarLoad=GREEN}]}

Zaliznuca{redCarload=[Carload{typeCarLoad=RED}, Carload{typeCarLoad=RED}], greenCarload=[Carload{typeCarLoad=GREEN}, Carload{typeCarLoad=GREEN}]}

**Завдання №2**

**Клас List**

**package** com.company.Lab2;  
  
*/\*\*  
 \* Created by Bogdan on 17.10.2016.  
 \*/***public class** List {  
 **private** ListElement **head**;  
 **private** ListElement **tail**;  
 **public void** addFront(**int** data){  
 ListElement a = **new** ListElement();  
 a.**data** = data;  
 **if**(**head** == **null**)  
 {  
 **head** = a;  
 **tail** = a;  
 }  
 **else** {  
 a.**next** = **head**;  
 **head** = a;  
 }  
 }  
 **public void** addBack(**int** data) {  
 ListElement a = **new** ListElement();  
 a.**data** = data;  
 **if** (**tail** == **null**)  
 {  
 **head** = a;  
 **tail** = a;  
 } **else** {  
 **tail**.**next** = a;  
 **tail** = a;  
 }  
 }  
 **public void** printList()  
 {  
 ListElement t = **head**;  
 **while** (t != **null**)  
 {  
 System.***out***.print(t.**data** + **" "**);  
 t = t.**next**;  
 }  
 }  
 **public void** delEl(**int** data)  
 {  
 **if**(**head** == **null**)  
 **return**;  
  
 **if** (**head** == **tail**) {  
 **head** = **null**;  
 **tail** = **null**;  
 **return**;  
 }  
  
 **if** (**head**.**data** == data) {  
 **head** = **head**.**next**;  
 **return**;  
 }  
  
 ListElement t = **head**;  
 **while** (t.**next** != **null**) {  
 **if** (t.**next**.**data** == data) {  
 **if**(**tail** == t.**next**)  
 {  
 **tail** = t;  
 }  
 t.**next** = t.**next**.**next**;  
 **return**;  
 }  
 t = t.**next**;  
 }  
 }  
  
 **public boolean** check() {  
 ListElement t = **head**;  
 **while** (t.**next** != **null**)  
 {  
 **if**(t.**data**>t.**next**.**data**){  
 **return false**;  
 }  
 t = t.**next**;  
 }  
 **return true**;  
 }  
}  
**class** ListElement {  
 ListElement **next**;  
 **double data**;  
}

**Використання класу**

List ml = **new** List();  
ml.addBack(1);  
ml.addBack(2);  
ml.addBack(3);  
ml.addBack(6);  
ml.addBack(8);  
ml.addBack(9);  
ml.addBack(11);  
ml.addBack(13);  
ml.addBack(16);  
  
ml.printList();  
System.***out***.println();  
System.***out***.println(ml.check());  
  
  
ml.addBack(0);  
ml.printList();  
System.***out***.println();  
System.***out***.println(ml.check());

**Результат роботи програми.**

1.0 2.0 3.0 6.0 8.0 9.0 11.0 13.0 16.0

true

1.0 2.0 3.0 6.0 8.0 9.0 11.0 13.0 16.0 0.0

false

**Висновок**: списки різних типів є дуже корисними в руках досвідченого програміста. Це дає змогу використовувати вже готові конструкції для роботи з даними, і це покращує розуміння коду. На лабораторній роботі перед мною стояло два завдання, які я успішно виконав, і використав набуті знання на практиці.