Міністерство освіти і науки України

Національний технічний університет

«Харківський політехнічний інститут»

Кафедра «Обчислювальна техніка та програмування»

ЗВІТ

Про виконання лабораторної роботи 10

“Обробка параметризованих контейнерів”

Виконавець:

студент гр. КН-920В

Олексієнко Микита

Харків 2022

Лабораторна робота №10.

**Обробка праметризованих контейнерів**

**Мета**

Розширення функціональності параметризованих класів.

**1.Вимоги**

**1.1 Розробник:**

Олексієнко Микита Віталійович

студент групи КН-920В

1. варіант

**1.2 Загальне завдання**

1. Розробити параметризовані методи (Generic Methods) для обробки колекцій об'єктів згідно прикладної задачі.
2. Продемонструвати розроблену функціональність (створення, управління та обробку власних контейнерів) в діалоговому та автоматичному режимах.
   * Автоматичний режим виконання програми задається параметром командного рядка **-auto**. Наприклад, java ClassName -auto.
   * В автоматичному режимі діалог з користувачем відсутній, необхідні данні генеруються, або зчитуються з файлу.
3. Забороняється використання алгоритмів 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.

Автосалон. Сортування за роком випуску авто, за витратою палива в змішаному циклі, за запитуваною ціною.

**2. Опис програми**

**2.1 Засоби ООП:**

У лабороторній роботі 10 , я використовую клас Program, який має у собі лише запуск меню, розробленого лабороторної роботи 8. Ця работа передбачає удосконалення власного контейнера, додатком Generic Method сортування.

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:**

Container.java ( додаткові методи ) :

public <Y extends Comparable<Y>, F extends Comparable<F>, P extends Comparable<P>> void sortByReleaseYear () {  
  
 Object[] array = container.toArray();  
 Object temp = null;  
  
 for ( int i = 0; i < array.length; i++ ) {  
 for( int j = i + 1; j < array.length; j++) {  
 if ( ((Auto) array[i]).getReleaseYear().compareTo(((Auto) array[j]).getReleaseYear() ) > 0 )  
 {  
 temp = array[i];  
 array[i] = array[j];  
 array[j] = temp;  
 }  
 }  
 }  
  
 container.fromArray(array);  
}  
  
public void sortByFuel () {  
  
 Object[] array = container.toArray();  
 Object temp = null;  
  
 for ( int i = 0; i < array.length; i++ ) {  
 for( int j = i + 1; j < array.length; j++) {  
  
 final int IFuel = (Integer) ((Auto) array[i]).getUrbanFuel() + (Integer) ((Auto) array[i]).getSubUrbanFuel();  
 final int JFuel = (Integer) ((Auto) array[j]).getUrbanFuel() + (Integer) ((Auto) array[j]).getSubUrbanFuel();  
  
 if ( IFuel > JFuel )  
 {  
 temp = array[i];  
 array[i] = array[j];  
 array[j] = temp;  
 }  
 }  
 }  
  
 container.fromArray(array);  
}  
public <Y extends Comparable<Y>, F extends Comparable<F>, P extends Comparable<P>> void sortByPrice () {  
  
 // P.S.  
 // относительно этого метода сортировки не было точного понятия,  
 // как можно отсортировать массив по заданной цене.  
 // Поэтому сортировка происходит от дешевой машины к дорогой.  
  
 Object[] array = container.toArray();  
 Object temp = null;  
  
 for ( int i = 0; i < array.length; i++ ) {  
 for( int j = i + 1; j < array.length; j++) {  
  
 if ( ((Auto) array[i]).getPrice().compareTo(((Auto) array[j]).getPrice() ) > 0 )  
 {  
 temp = array[i];  
 array[i] = array[j];  
 array[j] = temp;  
 }  
 }  
 }  
  
 container.fromArray(array);  
}

Menu.java ( перероблений метод create ) :

public static void create (String[] args) {  
  
 String ConsArg = args.length==0 ? "" : args[0];  
  
 switch (ConsArg) {  
 case "-auto" -> CarContainerInit();  
 case "" -> {  
 System.out.println("\n" + ANSI\_YELLOW + "WARNING:" + ANSI\_RESET);  
 System.out.println("The program was launched without flags.\n");  
 }  
 default -> {  
 System.out.println("\n" + ANSI\_YELLOW + "WARNING:" + ANSI\_RESET);  
 System.out.println("Invalid flags.\n");  
 }  
 }  
  
 while (true) {  
 System.out.println(ANSI\_GREEN + "CHOOSE ANY TASK:" + ANSI\_RESET);  
 System.out.println(ANSI\_YELLOW + "[1] Add Car");  
 System.out.println("[2] Remove Car");  
 System.out.println("[3] Car List");  
 System.out.println("[4] Standard Save");  
 System.out.println("[5] Standard Load");  
 System.out.println("[6] XML Save");  
 System.out.println("[7] XML Load");  
 System.out.println("[8] Close" + ANSI\_RESET + "\n");  
  
 Scanner scan = new Scanner(System.in);  
 int chooseNumber = scan.nextInt();  
  
 Auto car = new Auto();  
  
 switch (chooseNumber) {  
 case 1:  
 if (carAdding()) {  
 System.out.println(ANSI\_GREEN + "ADDED!\n" + ANSI\_RESET );  
 } else {  
 System.err.println("Please try add correctly");  
 }  
 break;  
 case 2: container.removeByPos( inputCarNumber() ); break;  
 case 3: {  
 container.carList();  
 break;  
 }  
 case 4: SAVING( true ); break;  
 case 5: LOADING( true ); break;  
 case 6: SAVING( false ); break;  
 case 7: LOADING( false ); break;  
 case 8:  
 return;  
 default: {  
 System.err.println("ERROR");  
 System.out.println("- Please select the correct command!");  
 }  
 }  
 }  
}

**3. Висновок:**

Розширено функціональності параметризованих класів.