**Звіт**

**Лабораторна работа 9. Параметризація в Java**

**Мета роботи**: Оволодіння навичками управління введенням / виведенням даних з використанням класів платформи Java SE.

**ВИМОГИ**

* Вивчення принципів параметризації в Java.
* Розробка параметризованих класів та методів.
  1. **Розробник**: Капелька Ярослав Іванович, КІТ-119а, варіант №9.

1. **ОПИС ПРОГРАМИ**
   1. **Засоби ООП**: клас, метод класу, поле класу.
   2. **Ієрархія та структура класів:** один публічний клас Main, публічний клас Route, у якого є поля: назва маршруту, загальна кількість місць, дні тижня; номер рейсу, назва станції, час прибуття , час відправлення, кількість вільних місць, статус станції, гетери, сетери, конструктор класу та метод виведення даних класу. Також є клас Data, який виконує роль покажчика на елемент і клас MyCollection, який містить покажчик на головний елемент та методи обробки масиву елементів.
   3. **Важливі фрагменти програми:**

**Data.java**

**public** **class** Data <T>

{

**public** T obj;

**public** Data<T> next;

**public** Data<T> prev;

Data()

{

}

Data(T obj, Data<T> prev, Data<T> next)

{

**this**.obj = obj;

**this**.next = next;

**this**.prev = prev;

}

}

**Main09.java**

**public** **class** Main09

{

**public** **static** **void** main(String[] args)

{

MyCollection<Route> ticketOffice = **new** MyCollection<Route>();

Route routeData = **new** Route("Лозовая-Харьков","Лозовая","15:20","15:24",55,"Начальная",150,**new** GregorianCalendar(2021, 11, 7),1);

ticketOffice.add(routeData);

ticketOffice.add( **new** Route("Харьков-Изюм","Маяк","14:52","14:53",61,"Промежуточная",200,**new** GregorianCalendar(2021, 11, 8),2));

ticketOffice.add( **new** Route("Красноград-Харьков","Харьков","19:34","Времени отправления нету, так как это конечная станция.",78,"Конечная",170,**new** GregorianCalendar(2021, 1, 20),3));

ticketOffice.add( **new** Route("Львов-Мариуполь","Льзов","01:07","01:12",55,"Начальная",230,**new** GregorianCalendar(2021, 2, 15),4));

ticketOffice.add( **new** Route("Ужгород-Киев","Мукачево","17:59","18:05",69,"Промежуточная",200,**new** GregorianCalendar(2021, 2, 25),5));

System.***out***.println("Происходит поиск элемента в колекции...");

System.***out***.println(ticketOffice.find(routeData));

**if** (ticketOffice.find(routeData) == **true**)

{

System.***out***.println("Элемент в коллекции найден.");

}

**else** **if** (ticketOffice.find(routeData) == **false**)

{

System.***out***.println("Элемент в коллекции не найден.");

}

System.***out***.println("---------------");

System.***out***.println("Вывод коллекции:");

System.***out***.println(ticketOffice);

System.***out***.println("---------------");

System.***out***.println("Происходит удаление элемента из коллекции...");

ticketOffice.delete(4);

System.***out***.println("Элемент успешно удален");

System.***out***.println("---------------");

System.***out***.println("Вывод коллекции с удаленным элементом:");

System.***out***.println(ticketOffice);

System.***out***.println("---------------");

System.***out***.println("Происходит сериализация и десериализация коллекции...");

**try**

{

ticketOffice.saveXml("SaveXML.xml");

ticketOffice.clear();

System.***out***.println(ticketOffice);

ticketOffice.downloadXml("SaveXML.xml");

System.***out***.println(ticketOffice);

} **catch** (FileNotFoundException e)

{

e.printStackTrace();

}

System.***out***.println("Вывод коллекции в виде массива:");

Route []mass = ticketOffice.toArray(**new** Route[ticketOffice.getSize()]);

**for**(Route p : mass)

{

System.***out***.println(p);

}

}

}

**MyCollection.java**

**public** **class** MyCollection <T> **implements** Iterable<T>, Serializable

{

**static** **final** **long** ***serialVersionUID*** = 1L;

**private** **int** size;

**private** Data<T> start;

**private** Data<T> last;

**public** **void** saveSer(String fileName) **throws** IOException

{

FileOutputStream outputStream = **new** FileOutputStream(fileName);

ObjectOutputStream objectOutputStream = **new** ObjectOutputStream(outputStream);

**for**(T value : **this**)

objectOutputStream.writeObject(value);

objectOutputStream.close();

}

@SuppressWarnings("unchecked")

**public** **void** downloadSer(String fileName) **throws** IOException, ClassNotFoundException

{

FileInputStream inStream = **new** FileInputStream(fileName);

ObjectInputStream objectInStream = **new** ObjectInputStream(inStream);

**try**

{

**while**(**true**)

{

add((T) objectInStream.readObject());

}

}**catch**(EOFException e)

{

objectInStream.close();

}

}

**public** **boolean** find(T obj)

{

**for**(T value : **this**)

{

**if**(value.equals(obj))

**return** **true**;

}

**return** **false**;

}

**public** String toString()

{

String str = **new** String();

**for**(T value : **this**)

{

str += value + "\n";

}

**return** str;

}

**public** **void** clear()

{

start.next = last;

last.prev = start;

size = 0;

}

**public** **void** saveXml(String fileName) **throws** FileNotFoundException

{

XMLEncoder encoder = **new** XMLEncoder(**new** BufferedOutputStream(**new** FileOutputStream(fileName)));

**for**(T value : **this** )

encoder.writeObject(value);

encoder.close();

System.***out***.println("Сериализация прошла успешно\n");

}

@SuppressWarnings("unchecked")

**public** **void** downloadXml(String fileName) **throws** FileNotFoundException

{

XMLDecoder d = **new** XMLDecoder(**new** BufferedInputStream(**new** FileInputStream(fileName)));

**try**

{

**while**(**true**)

{

add((T) d.readObject());

}

}**catch**(ArrayIndexOutOfBoundsException e)

{

d.close();

System.***out***.println("Десериализация прошла успешно\n");

}**catch**(Exception e)

{

**throw** e;

}

}

**public** **int** getSize()

{

**return** size;

}

MyCollection()

{

size = 0;

start = **new** Data<T>(**null**,**null**,last);

last = **new** Data<T>(**null**,start,**null**);

}

**void** add(T obj)

{

size++;

Data<T> temp = last.prev;

temp.next = **new** Data<T>();

last.prev = temp.next;

temp.next.obj = obj;

temp.next.next = last;

temp.next.prev = temp;

}

**void** delete(**int** itr)

{

**if**(itr>=size)

**return**;

Data<T> temp = start.next;

Data<T> temp2 = **null**;

**for**(**int** i = 0; i < itr;i++)

{

temp = temp.next;

}

temp2 = temp.prev;

temp2.next = temp.next;

temp.next.prev = temp2;

size--;

}

T get(**int** itr)

{

**if**(itr>=size && itr<0)

**return** **null**;

Data<T> temp = start.next;

**for**(**int** i = 0; i < itr;i++)

{

temp = temp.next;

}

**return** temp.obj;

}

**public** T[] toArray(T[] arr)

{

**for**(**int** i = 0; i < size; i ++)

{

arr[i] = get(i);

}

**return** arr;

}

@Override

**public** Iterator<T> iterator()

{

**return** **new** Iterator<T>()

{

**int** itr = 0;

@Override

**public** **boolean** hasNext()

{

**return** itr < size;

}

@Override

**public** T next()

{

**return** get(itr++);

}

@Override

**public** **void** remove()

{

delete(itr-1);

}

};

}

}

**Route.java**

**public** **class** Route **implements** Serializable

{

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** String name\_route;

**private** String station\_name;

**private** String departure\_time;

**private** String arrival\_time;

**private** **int** number\_of\_free\_seats;

**private** String status\_station;

**private** **int** total\_number\_of\_seats;

**private** GregorianCalendar days\_of\_the\_week;

**private** **int** flight\_number;

**public** **void** setNameRoute(String name\_route)

{

**this**.name\_route = name\_route;

}

**public** String getNameRoute()

{

**return** name\_route;

}

**public** String getStation\_name()

{

**return** station\_name;

}

**public** **void** setStation\_name(String station\_name)

{

**this**.station\_name = station\_name;

}

**public** String getDeparture\_time()

{

**return** departure\_time;

}

**public** **void** setDeparture\_time(String departure\_time)

{

**this**.departure\_time = departure\_time;

}

**public** String getArrival\_time()

{

**return** arrival\_time;

}

**public** **void** setArrival\_time(String arrival\_time)

{

**this**.arrival\_time = arrival\_time;

}

**public** **int** getNumber\_of\_free\_seats()

{

**return** number\_of\_free\_seats;

}

**public** **void** setNumber\_of\_free\_seats(**int** number\_of\_free\_seats)

{

**this**.number\_of\_free\_seats = number\_of\_free\_seats;

}

**public** String getStatus\_station()

{

**return** status\_station;

}

**public** **void** setStatus\_station(String status\_station)

{

**this**.status\_station = status\_station;

}

**public** **void** setTotal\_number\_of\_seats(**int** total\_number\_of\_seats)

{

**this**.total\_number\_of\_seats = total\_number\_of\_seats;

}

**public** **int** getTotal\_number\_of\_seats()

{

**return** total\_number\_of\_seats;

}

**public** **void** setDays\_of\_the\_week(GregorianCalendar days\_of\_the\_week)

{

**this**.days\_of\_the\_week = days\_of\_the\_week;

}

**public** GregorianCalendar getDays\_of\_the\_week()

{

**return** days\_of\_the\_week;

}

**public** **int** getFlight\_number()

{

**return** flight\_number;

}

**public** **void** setFlight\_number(**int** flight\_number)

{

**this**.flight\_number = flight\_number;

}

**public** Route()

{

**super**();

}

**public** **void** print()

{

System.***out***.println("\nИмя маршрута: " + **this**.getNameRoute()+"\nИмя станции: "+ **this**.getStation\_name() + "\nВремя пребытия на станцию: " + **this**.getArrival\_time()+ "\nВремя отправления со станции: " + **this**.getDeparture\_time()+"\nКоличество пустых мест: "+ **this**.getNumber\_of\_free\_seats()+"\nСтатус станции: "+ **this**.getStatus\_station()+"\nОбщее количество мест: "+ **this**.getTotal\_number\_of\_seats()+"\nДень недели: "+ **this**.getDays\_of\_the\_week().getTime()+"\nНомер рейсу: "+ **this**.getFlight\_number());

}

@Override

**public** String toString()

{

**return** **new** String("\nИмя маршрута: " + **this**.getNameRoute()+"\nИмя станции: "+ **this**.getStation\_name() + "\nВремя пребытия на станцию: " + **this**.getArrival\_time()+ "\nВремя отправления со станции: " + **this**.getDeparture\_time()+"\nКоличество пустых мест: "+ **this**.getNumber\_of\_free\_seats()+"\nСтатус станции: "+ **this**.getStatus\_station()+"\nОбщее количество мест: "+ **this**.getTotal\_number\_of\_seats()+"\nДень недели: "+ **this**.getDays\_of\_the\_week().getTime()+"\nНомер рейсу: "+ **this**.getFlight\_number());

}

Route(String name, **int** total\_number,GregorianCalendar days,**int** flight)

{

**this**.setNameRoute(name);

setTotal\_number\_of\_seats(total\_number);

**this**.setDays\_of\_the\_week(days);

setFlight\_number(flight);

}

Route(String name, String name1, String time,String time1, **int** number, String status, **int** total\_number, GregorianCalendar days,**int** flight)

{

**this**.setNameRoute(name);

**this**.setStation\_name(name1);

**this**.setArrival\_time(time);

**this**.setDeparture\_time(time1);

**this**.setNumber\_of\_free\_seats(number);

**this**.setStatus\_station(status);

setTotal\_number\_of\_seats(total\_number);

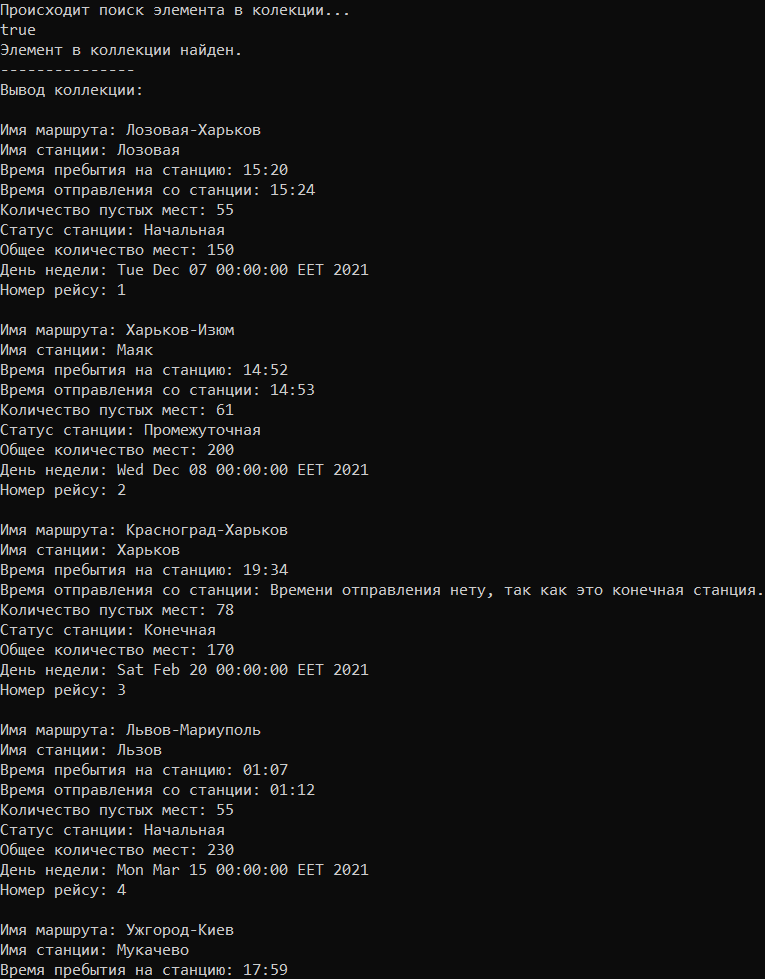
**this**.setDays\_of\_the\_week(days);

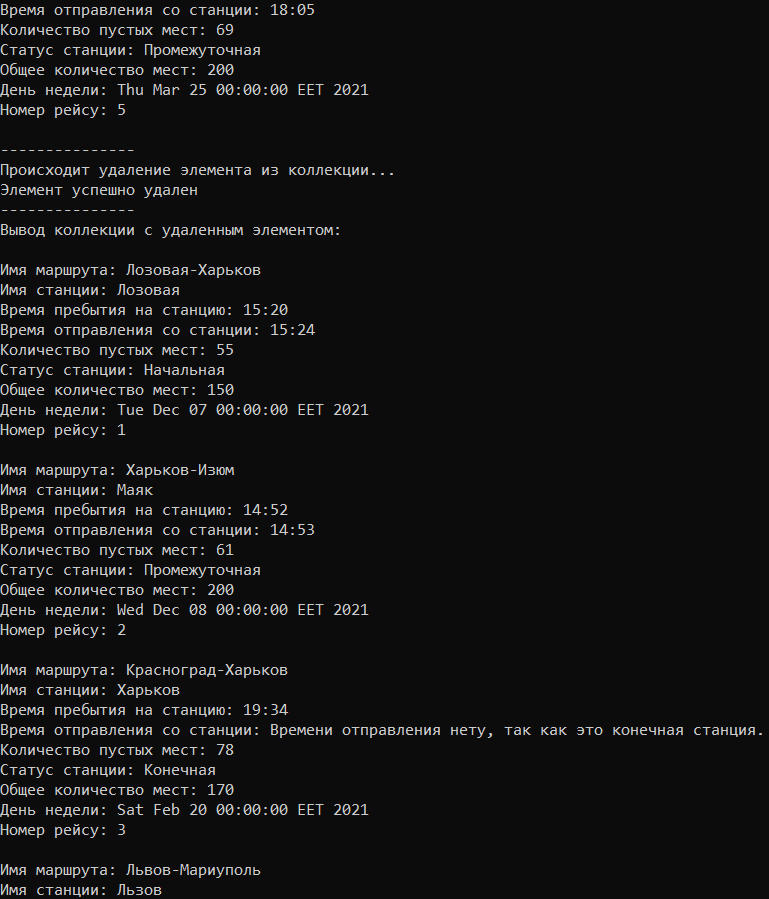
setFlight\_number(flight);

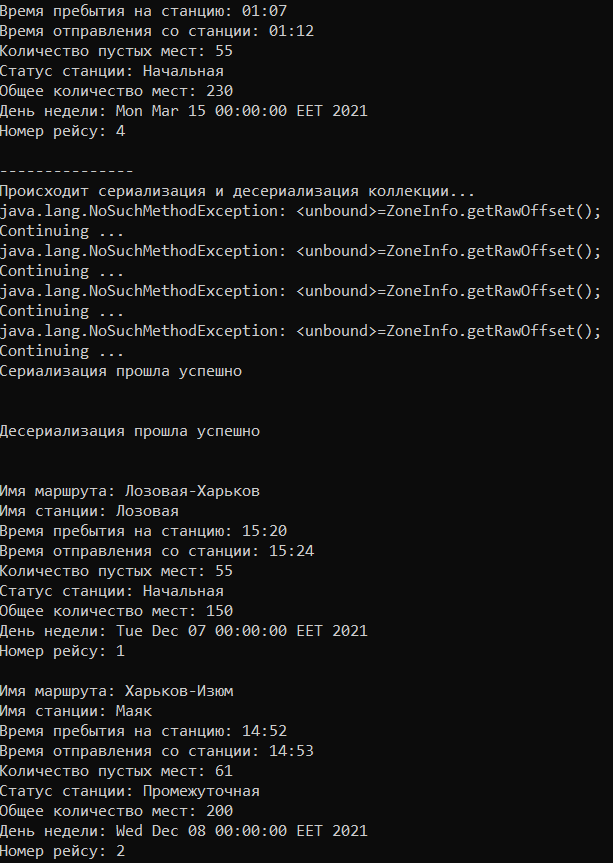
}

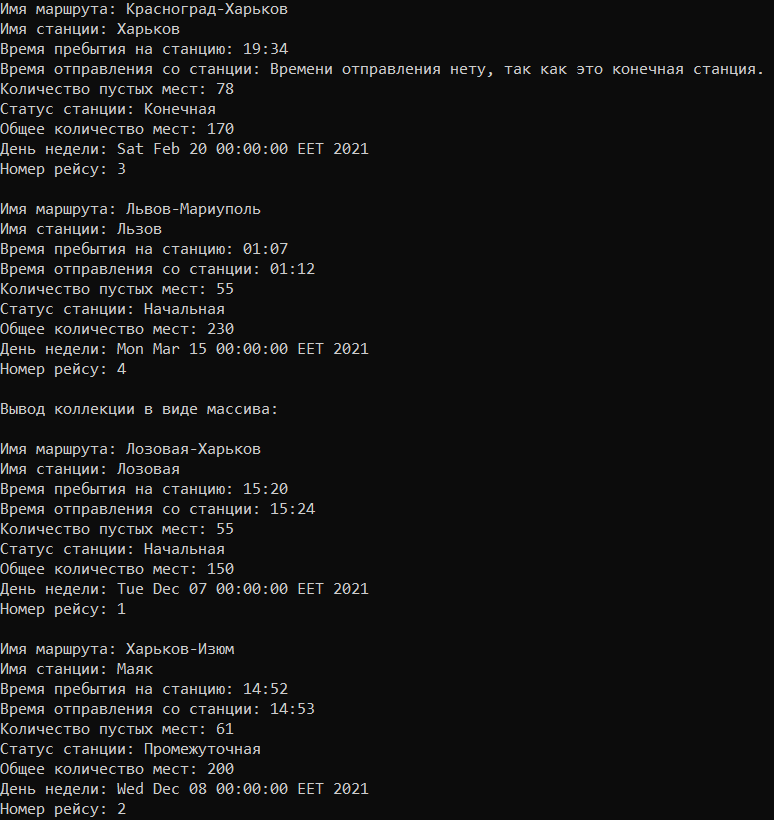
}

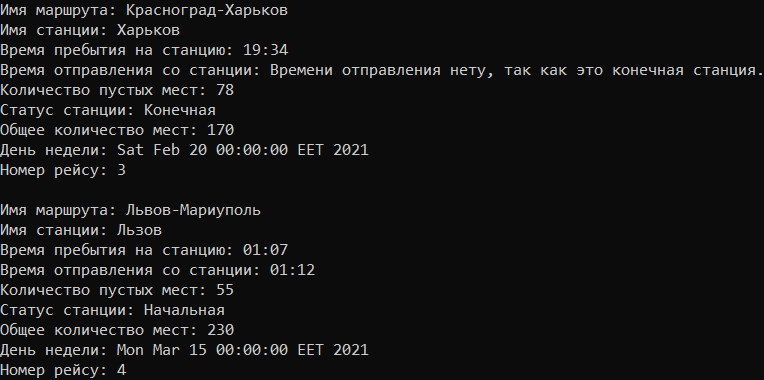
**Результат роботи програми**











**Висновки**

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з параметризацією.

Програма протестована, виконується без помилок.