Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра «Електронних обчислювальних машин»



Звіт

з лабораторної роботи № 6

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «ПАРАМЕТРИЗОВАНЕ ПРОГРАМУВАННЯ»

**Виконав:**

студент групи КІ-301

*Солтисяк О.Р.*

**Прийняв:**

доцент кафедри ЕОМ

*Іванов Ю. С.*

Львів – 2024

**Мета роботи:** оволодіти навиками параметризованого програмування мовою Java.

**Завдання (варіант № 26)**

1. Створити параметризований клас, що реалізує предметну область задану варіантом **(Банка)**. Клас має містити мінімум 4 методи опрацювання даних включаючи розміщення та виймання елементів. Парні варіанти реалізують пошук мінімального елементу, непарні – максимального. Написати на мові Java та налагодити програму-драйвер для розробленого класу, яка мстить мінімум 2 різні класи екземпляри яких розмішуються у екземплярі розробленого класу-контейнеру. Програма має розміщуватися в пакеті Група.Прізвище.Lab6 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
2. Автоматично згенерувати документацію до розробленого пакету.
3. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.
4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.
5. Дати відповідь на контрольні запитання.

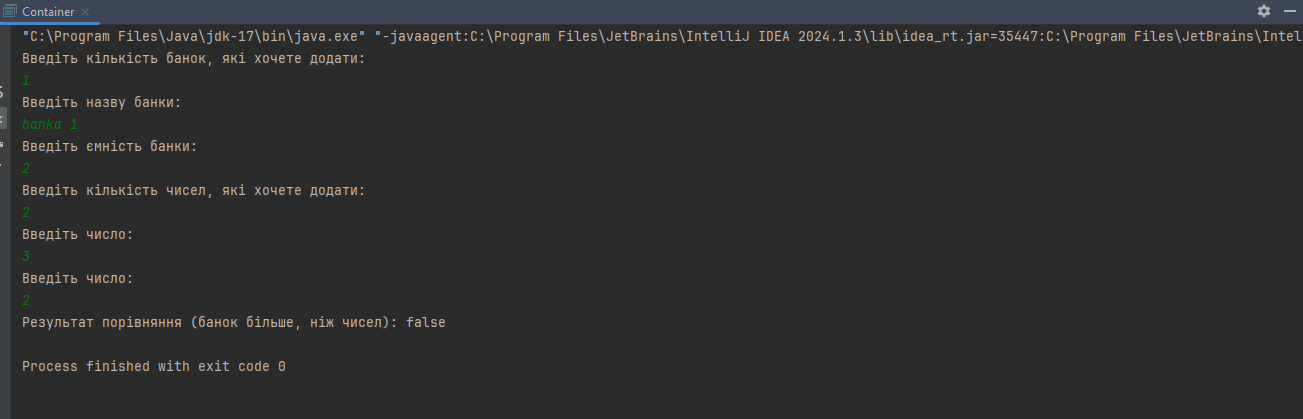
**Вихідний код програми:**

package org.example;/\* Bank.java \*/  
  
*/\*\*  
 \* Клас Bank представляє предметну область "банка"  
 \*/*public class Banka implements Comparable<Banka> {  
 private String name;  
 private double capacity;  
  
 */\*\*  
 \* Конструктор для створення нової банки  
 \** ***@param*** *name назва банки  
 \** ***@param*** *capacity ємність банки  
 \*/* public Banka(String name, double capacity) {  
 this.name = name;  
 this.capacity = capacity;  
 }  
  
 */\*\*  
 \* Повертає назву банки  
 \** ***@return*** *назва банки  
 \*/* public String getName() {  
 return name;  
 }  
  
 */\*\*  
 \* Повертає ємність банки  
 \** ***@return*** *ємність банки  
 \*/* public double getCapacity() {  
 return capacity;  
 }  
  
 @Override  
 public int compareTo(Banka other) {  
 return Double.*compare*(this.capacity, other.capacity);  
 }  
  
 @Override  
 public String toString() {  
 return "Bank{" +  
 "name='" + name + '\'' +  
 ", capacity=" + capacity +  
 '}';  
 }  
  
}

***Файл Container.java***

package org.example;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
  
*/\*\*  
 \* Клас Container реалізує параметризований контейнер для предметної області "банка".  
 \** ***@param*** <*T*> *тип об'єктів, які будуть зберігатися в контейнері  
 \*/*public class Container<T extends Comparable<T>> {  
 private List<T> elements;  
  
 */\*\*  
 \* Конструктор для створення нового контейнера  
 \*/* public Container() {  
 elements = new ArrayList<>();  
 }  
  
 */\*\*  
 \* Додає новий елемент у контейнер  
 \** ***@param*** *element елемент, який необхідно додати  
 \*/* public void addElement(T element) {  
 elements.add(element);  
 }  
  
 */\*\*  
 \* Повертає кількість елементів у контейнері  
 \** ***@return*** *кількість елементів  
 \*/* public int getSize() {  
 return elements.size();  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
  
 Container<Banka> bankContainer = new Container<>();  
  
 Container<Integer> intContainer = new Container<>();  
  
  
 System.*out*.println("Введіть кількість банок, які хочете додати: ");  
 int bankCount = scanner.nextInt();  
 scanner.nextLine();  
  
 for (int i = 0; i < bankCount; i++) {  
 System.*out*.println("Введіть назву банки: ");  
 String name = scanner.nextLine();  
 System.*out*.println("Введіть ємність банки: ");  
 double capacity = scanner.nextDouble();  
 scanner.nextLine(); // споживаємо залишок нового рядка  
 bankContainer.addElement(new Banka(name, capacity));  
 }  
  
  
 System.*out*.println("Введіть кількість чисел, які хочете додати: ");  
 int intCount = scanner.nextInt();  
  
 for (int i = 0; i < intCount; i++) {  
 System.*out*.println("Введіть число: ");  
 int number = scanner.nextInt();  
 intContainer.addElement(number);  
 }  
  
  
 boolean result = bankContainer.getSize() > intContainer.getSize();  
 System.*out*.println("Результат порівняння (банок більше, ніж чисел): " + result);  
  
 scanner.close();  
 }  
}

**Результат виконання програми:**



**Порівнював от так, оскільки це було моїм додатковим завданням при захисті.**

**Відповіді на контрольні запитання:**

1. Дайте визначення терміну «параметризоване програмування».

Параметризоване програмування (або generics) - це механізм мов програмування, який дозволяє створювати код, який може працювати з різними типами даних. Воно дозволяє створювати універсальні абстракції, які можуть бути використані для різних типів, забезпечуючи при цьому безпеку типів.

1. Розкрийте синтаксис визначення простого параметризованого класу.

public class MyGenericClass<T> {

// код класу

}

1. Розкрийте синтаксис створення об’єкту параметризованого класу.

MyGenericClass<Integer> myObject = new MyGenericClass<>();

1. Розкрийте синтаксис визначення параметризованого методу.

public <T> void myGenericMethod(T parameter) {

// код методу

}

1. Розкрийте синтаксис виклику параметризованого методу.

myGenericMethod("Hello");

1. Яку роль відіграє встановлення обмежень для змінних типів?

Встановлення обмежень для змінних типів (generics constraints) дозволяє обмежити можливі типи, які можна використовувати в параметризованих типах, забезпечуючи при цьому певний функціонал або властивості для цих типів.

1. Як встановити обмеження для змінних типів?

public class MyGenericClass<T extends SomeClass> {

// код класу

}

1. Розкрийте правила спадкування параметризованих типів.

Параметризовані типи спадковуються аналогічно непараметризованим типам. При цьому враховується також параметр типу.

1. Яке призначення підстановочних типів?

Підстановочні типи (wildcards) використовуються для створення більш гнучких методів та класів, які можуть працювати з різними типами. Два основних типи підстановочних типів: ? extends T (upper-bounded wildcard) і ? super T (lower-bounded wildcard).

1. Застосування підстановочних типів.

* ? extends T: Використовується для доступу до об'єктів з типами, які є підтипами T.
* ? super T: Використовується для передачі об'єктів з типами, які є супертипами T.

**Висновок:**

На даній лабораторній роботі я ознайомився з використанням параметризованого програмування. Створив клас який реалізує предметну область «валіза». Та розробив клас драйвер який показує роботу параметризованого класу-контейнера.