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-306*

*Савіцький Н. О.*

**Прийняв:**

доцент кафедри ЕОМ

*Іванов Ю. С.*

Львів – 2023

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

**Завдання (варіант № 19)**

1. Створити параметризований клас, що реалізує предметну область задану варіантом (19. Сейф). 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. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.
4. Дати відповідь на контрольні запитання.

**Вихідний код програми**

**Файл SafeDriver.java**

package KI306.Savitskyi.lab6;

import java.util.\*;

import java.io.\*;

/\*\*

\* The SafeDriver class is the main class for running the Safe program.

\* It demonstrates the usage of the Safe class to store and manipulate

\* data items of type Data, which can be either Jewel or Cash objects.

\*/

public class SafeDriver {

public static void main(String[] args) {

// Create a Safe instance to store Data objects

Safe<? super Data> safe = new Safe<Data>();

// Add various Data objects to the safe

safe.AddData(new Jewel("Golden ring", 4, 1500));

safe.AddData(new Cash("US Dollar", 7200));

safe.AddData(new Cash("Euro", 400));

safe.AddData(new Jewel("Silver chain", 23, 150));

// Find and print the Data object with the greatest value

Data res = safe.findMax();

res.print();

// Print all items in the safe in descending order of value

System.***out***.print("The greatest value in safe is: \n");

safe.printAllDescendingByValue();

// Print only the Jewel items in the safe

safe.printJewelsOnly();

}

}

/\*\*

\* The Safe class represents a safe for storing data items of type T, where T

\* extends the Data interface. It provides methods for adding, deleting, and

\* performing operations on the stored data.

\*/

class Safe<T extends Data> {

private ArrayList<T> arr;

/\*\*

\* Constructor to initialize the Safe with an empty ArrayList.

\*/

public Safe() {

arr = new ArrayList<T>();

}

/\*\*

\* Find and return the Data object with the maximum value in the safe.

\*

\* **@return** The Data object with the maximum value, or null if the safe is empty.

\*/

public T findMax() {

if (!arr.isEmpty()) {

T max = arr.get(0);

for (int i = 1; i < arr.size(); i++) {

if (arr.get(i).compareTo(max) > 0)

max = arr.get(i);

}

return max;

}

return null;

}

/\*\*

\* Add a Data object to the safe.

\*

\* **@param** data The Data object to add to the safe.

\*/

public void AddData(T data) {

arr.add(data);

System.***out***.print("Element added: ");

data.print();

}

/\*\*

\* Delete a Data object from the safe at the specified index.

\*

\* **@param** i The index of the Data object to delete.

\*/

public void DeleteData(int i) {

arr.remove(i);

}

/\*\*

\* Print all items in the safe in descending order of their values.

\*/

public void printAllDescendingByValue() {

Collections.*sort*(arr, Collections.*reverseOrder*());

System.***out***.println("Contents of the safe in descending order of value:");

for (T data : arr) {

data.print();

}

}

/\*\*

\* Print only the Jewel items in the safe.

\*/

public void printJewelsOnly() {

System.***out***.println("Jewels in the safe:");

for (T data : arr) {

if (data instanceof Jewel) {

data.print();

}

}

}

}

/\*\*

\* The Data interface represents a common interface for objects that have a

\* value and can be compared based on their values.

\*/

interface Data extends Comparable<Data> {

/\*\*

\* Get the value of the data object.

\*

\* **@return** The value of the data object.

\*/

public int getValue();

/\*\*

\* Print information about the data object.

\*/

public void print();

}

/\*\*

\* The Jewel class represents a type of Data object that represents a jewel.

\*/

class Jewel implements Data {

private String description;

private int weight;

private int value;

public Jewel(String jDescription, int jWeight, int jValue) {

description = jDescription;

weight = jWeight;

value = jValue;

}

public String getJewelDescription() {

return description;

}

public void setJewelDescription(String jDescription) {

description = jDescription;

}

public int getWeight() {

return weight;

}

public void setWeight(int n) {

weight = n;

}

public int getValue() {

return value;

}

public int compareTo(Data p) {

Integer s = value;

return s.compareTo(p.getValue());

}

public void print() {

System.***out***.print("Jewel description: " + description + ", Weight: " + weight + ", Jewel value: " + value + ";\n");

}

}

/\*\*

\* The Cash class represents a type of Data object that represents currency.

\*/

class Cash implements Data {

private String currencyName;

private int value;

public Cash(String cName, int cValue) {

currencyName = cName;

value = cValue;

}

public String getCurrencyName() {

return currencyName;

}

public void setCurrencyName(String name) {

currencyName = name;

}

public void setValue(int n) {

value = n;

}

public int getValue() {

return value;

}

public int compareTo(Data p) {

Integer s = value;

return s.compareTo(p.getValue());

}

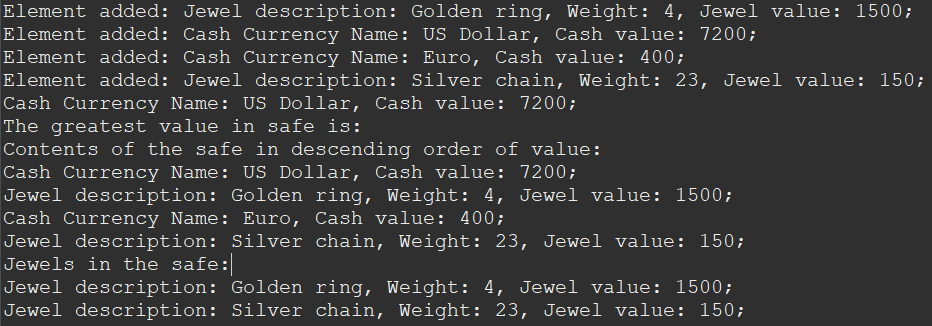
public void print() {

System.***out***.print("Cash Currency Name: " + currencyName + ", Cash value: " + value + ";\n");

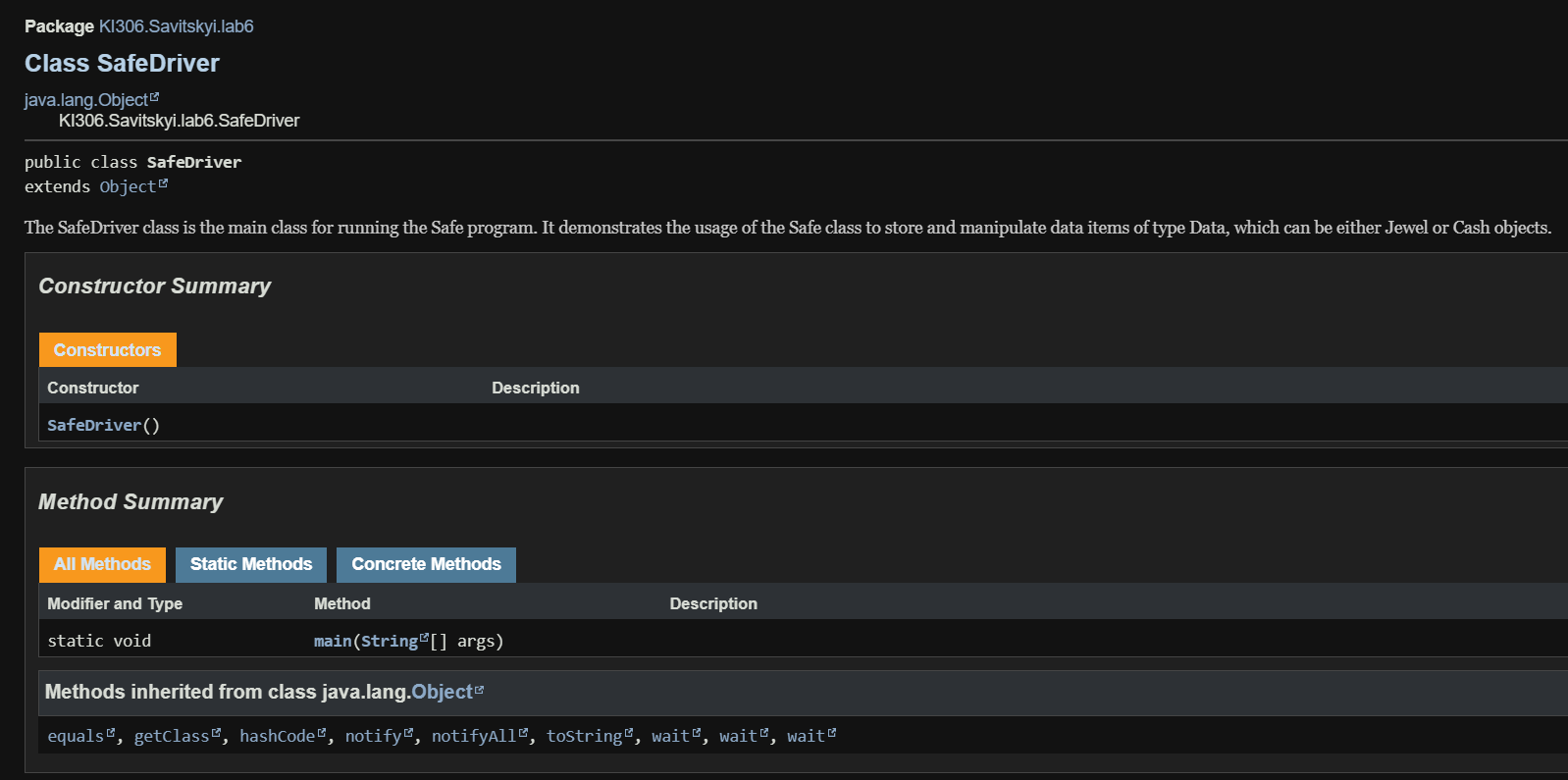
}

}

**Результат виконання програми**



**Фрагмент згенерованої документації**



**Відповіді на контрольні запитання**

1. Дайте визначення терміну «параметризоване програмування».

* це підхід до програмування, що дозволяє створювати класи і методи, які можна використовувати з різними типами даних, надаючи більшу гнучкість і безпеку типів у програмах.

1. Розкрийте синтаксис визначення простого параметризованого класу.

* public class НазваКласу<параметризованийТип> {

// Тіло класу

}

1. Розкрийте синтаксис створення об’єкту параметризованого класу.

* НазваКласу<перелікТипів> зміннаКласу = new НазваКласу<перелікТипів>(параметри);

1. Розкрийте синтаксис визначення параметризованого методу.

* public <параметризованийТип> типПовернення назваМетоду(параметри) {

// Тіло методу

}

1. Розкрийте синтаксис виклику параметризованого методу.

* (НазваКласу|НазваОб’єкту).<перелікТипів>назваМетоду(параметри);

1. Яку роль відіграє встановлення обмежень для змінних типів?

* дозволяє заборонити використання деяких типів або вимагати, щоб тип підставлений за замовчуванням був підкласом або реалізував певний інтерфейс.

1. Як встановити обмеження для змінних типів?

* за допомогою ключового слова extends для суперкласу або інтерфейсу, від яких має походити реальний тип.

1. Розкрийте правила спадкування параметризованих типів.

* Всі класи, створені з параметризованого класу, незалежні один від одного.
* Зазвичай немає залежності між класами, створеними з різними параметрами типів.

1. Яке призначення підстановочних типів?

* використовуються для забезпечення безпеки типів при використанні параметризованих класів та методів. Вони дозволяють визначити, які типи можна використовувати замість параметризованих типів.

1. Застосування підстановочних типів.

* <?> (unbounded wildcard) дозволяє читати об'єкти з колекції без змінення її.
* <? extends Тип> (bounded wildcard) дозволяє читати об'єкти з колекції, але забороняє додавання в неї нових об'єктів.
* <? super Тип> (lower bounded wildcard) дозволяє додавати об'єкти в колекцію, але забороняє їх читання.

**Висновок**

У ході виконання даної лабораторної роботи, я отримав важливі навички параметризованого програмування мовою Java. Ознайомився з різними аспектами мови, такими як використання параметрів у методах, створення та використання класів та інтерфейсів.