Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра “Електронних обчислювальних машин”



**з дисципліни**

**Кросплатформні засоби програмування**

**Звіт з Лабораторної роботи № 5**

**ФАЙЛИ У JAVA**

**Виконав:**

КІ-307

Герега Р. О.

**Перевірив:**

Іванов Ю.С

**2023**

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

**Теоретичний вступ**

ТЕОРЕТИЧНІ ВІДОМОСТІ

Параметризоване програмування є аналогом шаблонів у С++. Воно полягає у

написанні коду, що можна багаторазово застосовувати з об’єктами різних класів.

Користувачів параметризованого програмування можна поділити на 3 рівні кваліфікації:

1. ті, що користуються готовими класами;

2. ті, що користуються готовими класами і вміють виправляти помилки, що

виникають при їх використанні;

3. ті, що пишуть власні параметризовані класи.

Для успішного застосування параметризованого програмування слід навчитися

розуміти помилки, що генерує середовище при компіляції програми, що можуть

стосуватися, наприклад, неоднозначності визначення спільного суперкласу для всіх

переданих об’єктів. З іншої сторони необхідно передбачити захист від присвоєння

об’єктів параметризованого класу, що містять об’єкти підкласу об’єктам

параметризованого класу, що містять об’єкти суперкласу і дозволити зворотні

присвоєння. Для вирішення цієї проблеми у мові Java введено так звані підстановочні

типи. Це далеко не всі «підводні камені», що виникають при застосуванні

параметризованого програмування.

Визначення простого параметризованого класу

Параметризований клас – це клас з однією або більше змінними типу.

Тут T і U – це змінні параметризованих типів, що використовуються по всьому тілу

класу для специфікації типу повернення методів, типів полів і локальних змінних. При

створенні об’єкту параметризованого класу замість них підставляються реальні типи, що

визначаються в трикутних дужках у місці створення об’єкту параметризованого класу.

Синтаксис створення об’єкту параметризованого класу:

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

Приклад створення об’єкту параметризованого класу:

GenericClass<String, Integer> obj = new GenericClass<String, Integer> ();

**Варіант 6**

6. Шафа

**Завдання:**

1. Створити параметризований клас, що реалізує предметну область задану варіантом. 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.
2. Програма має розміщуватися в пакеті Група.Прізвище.Lab6 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету. 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 їївиконання та фрагменту згенерованої документації та завантажити його у ВНС.
5. Дати відповідь на контрольні запитання.

**Код**

package KI.Hereha.Lab6;

/\*\*

\* Клас для предметів, які розміщуються у шафі.

\*/

public class Book implements Comparable<Book> {

private String title;

private int pageCount;

public Book(String title, int pageCount) {

this.title = title;

this.pageCount = pageCount;

}

public String getTitle() {

return title;

}

public int getPageCount() {

return pageCount;

}

@Override

public int compareTo(Book other) {

return Integer.compare(this.pageCount, other.pageCount);

}

@Override

public String toString() {

return "Book{" +

"title='" + title + '\'' +

", pageCount=" + pageCount +

'}';

}

}

package KI.Hereha.Lab6;

/\*\*

\* Клас для предметів, які розміщуються у шафі.

\*/

public class Dish implements Comparable<Dish> {

private String name;

private double weight;

public Dish(String name, double weight) {

this.name = name;

this.weight = weight;

}

public String getName() {

return name;

}

public double getWeight() {

return weight;

}

@Override

public int compareTo(Dish other) {

return Double.compare(this.weight, other.weight);

}

@Override

public String toString() {

return "Dish{" +

"name='" + name + '\'' +

", weight=" + weight +

'}';

}

}

package KI.Hereha.Lab6;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

/\*\*

\* Параметризований клас, що реалізує предметну область "Шафа".

\* @param <T> Тип елементів, які можна розміщувати у шафі

\*/

public class Shelf<T extends Comparable<T>> {

private List<T> items;

/\*\*

\* Конструктор класу

\*/

public Shelf() {

this.items = new ArrayList<>();

}

/\*\*

\* Розміщення елемента у шафі.

\* @param item Елемент для розміщення

\*/

public void placeItem(T item) {

items.add(item);

}

/\*\*

\* Виймання мінімального елемента з шафи.

\* @return Мінімальний елемент

\*/

public T takeMinItem() {

if (items.isEmpty()) {

throw new IllegalStateException("Шафа порожня");

}

T minItem = Collections.min(items);

items.remove(minItem);

return minItem;

}

/\*\*

\* Виймання максимального елемента з шафи.

\* @return Максимальний елемент

\*/

public T takeMaxItem() {

if (items.isEmpty()) {

throw new IllegalStateException("Шафа порожня");

}

T maxItem = Collections.max(items);

items.remove(maxItem);

return maxItem;

}

/\*\*

\* Отримання усіх елементів у шафі.

\* @return Список елементів

\*/

public List<T> getAllItems() {

return new ArrayList<>(items);

}

}

package KI.Hereha.Lab6;

import java.util.List;

public class ShelfDriver {

public static void main(String[] args) {

// Створення шафи для книг

Shelf<Book> bookShelf = new Shelf<>();

// Розміщення книг у шафі

bookShelf.placeItem(new Book("Java Programming", 300));

bookShelf.placeItem(new Book("Data Structures", 250));

// Вивід усіх книг у шафі

System.out.println("Книги у шафі:");

List<Book> allBooks = bookShelf.getAllItems();

for (Book book : allBooks) {

System.out.println(book);

}

// Виймання мінімальної кількості сторінок

Book minPageCountBook = bookShelf.takeMinItem();

System.out.println("Мінімальна кількість сторінок: " + minPageCountBook.getPageCount());

// Створення шафи для посуду

Shelf<Dish> dishShelf = new Shelf<>();

// Розміщення посуду у шафі

dishShelf.placeItem(new Dish("Plate", 0.5));

dishShelf.placeItem(new Dish("Cup", 0.3));

// Вивід усього посуду у шафі

System.out.println("Посуд у шафі:");

List<Dish> allDishes = dishShelf.getAllItems();

for (Dish dish : allDishes) {

System.out.println(dish);

}

// Виймання максимальної ваги

Dish maxWeightDish = dishShelf.takeMaxItem();

System.out.println("Максимальна вага посуду: " + maxWeightDish.getWeight());

}

}

**Результат:**

Книги у шафі:

Book{title='Java Programming', pageCount=300}

Book{title='Data Structures', pageCount=250}

Мінімальна кількість сторінок: 250

Посуд у шафі:

Dish{name='Plate', weight=0.5}

Dish{name='Cup', weight=0.3}

Максимальна вага посуду: 0.5

**Короткий висновок**

Під час виконання даної лабораторної роботи я оволодів навиками параметризованого програмування мовою Java.