

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*

Ярема Максим

Прийняв:

доцент кафедри ЕОМ

Іванов Ю. С.

Мета роботи: оволодіти навиками параметризованого програмування мовою Java.

Завдання (варіант № 29)

1. Створити параметризований клас, що реалізує предметну область задану варіантом (29. Тумбочка). 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 Main.java

```
package ki306.yarema.lab7;

/**
 * Class EquationsApp Implements driver for Equations class
 *
 * @author Yarema Maksym
 * @version 1.0
 */
public class Main {
    /**
     * The main method is the entry point of the program.
     *
     * @param args The command-line arguments (not used in this program).
     */
    public static void main(String[] args) {
        BedsideTable <? super Item> bsTable = new BedsideTable<>();
        bsTable.putItem(new Book("Atomic Habits", 1.852));
        bsTable.putItem(new Flashlight("Mobile flashlight" , 0.13));
        bsTable.putItem(new Book("The Catcher in the Rye" , 1.243));
        bsTable.putItem(new Flashlight("Endless light +Ultra" , 1.726));

        Item item = bsTable.getItem(2);
        item.print();

        item = bsTable.getItem(3);
        item.use();

        Item max = bsTable.getMax();
        System.out.println("\nThe heaviest item in bedside table is: ");
        max.print();
    }
}
```

Файл Item.java

```
package ki306.yarema.lab7;

/**
 * The Item interface represents items that can be used and have weight.
 * Classes that implement this interface should provide implementations
 * for the methods declared here.
 *
 * @author Yarema Maksym
 * @version 1.0
 */
public interface Item extends Comparable<Item> {
    /**
     * Get the weight of the item.
     *
     * @return The weight of the item as a double value.
     */
    public double getWeight();

    /**
     * Perform an action with the item.
     */
    public void use();

    /**
     * Print information about the item.
     */
    public void print();
}
```

Файл Book.java

```
package ki306.yarema.lab7;

/**
 * The Book class represents a book item that implements the Item interface.
 * It has a name and a weight, and it can be used for reading.
 *
 * @author Yarema Maksym
 * @version 1.0
 */
public class Book implements Item {
    private String name;
    private double weight;

    /**
     * Constructs a new Book object with the specified name and weight.
     *
     * @param name The name of the book.
     * @param weight The weight of the book as a double value.
     */
    public Book(String name, double weight) {
        this.name = name;
        this.weight = weight;
    }

    /**
     * Compares this book's weight to the weight of another item that implements the Item
     * interface.
     *
     * @param item The item to compare to.
     * @return A negative integer if this book is lighter, a positive integer if
     * it's heavier, or 0 if they have the same weight.
     */
    public int compareTo(Item item) {
        Double w = weight;
        return w.compareTo(item.getWeight());
    }
}
```

```

/**
 * Get the weight of the book.
 *
 * @return The weight of the book as a double value.
 */
@Override
public double getWeight() {
    return weight;
}

/**
 * Use the book for reading.
 */
@Override
public void use() {
    System.out.println("Reading a " + name);
}

/**
 * Print information about the book.
 */
@Override
public void print() {
    System.out.println("Book: " + name + ", weights: " + weight);
}
}

```

Файл Flashlight.java

```

package ki306.yarema.lab7;

/**
 * The Flashlight class represents a flashlight item that implements the Item interface.
 * It has a name and a weight, and it can be used as a flashlight.
 *
 * @author Yarema Maksym
 * @version 1.0
 */
public class Flashlight implements Item {
    private String name;
    private double weight;

    /**
     * Constructs a new Flashlight object with the specified name and weight.
     *
     * @param name The name of the flashlight.
     * @param weight The weight of the flashlight as a double value.
     */
    public Flashlight(String name, double weight) {
        this.name = name;
        this.weight = weight;
    }

    /**
     * Compares this flashlight's weight to the weight of another item that implements the
     * Item interface.
     *
     * @param item The item to compare to.
     * @return A negative integer if this flashlight is lighter, a positive integer if
     * it's heavier, or 0 if they have the same weight.
     */
    public int compareTo(Item item) {
        Double w = weight;
        return w.compareTo(item.getWeight());
    }

    /**
     * Get the weight of the flashlight.
     *
     */
}

```

```

* @return The weight of the flashlight as a double value.
*/
@Override
public double getWeight() {
    return weight;
}

/**
* Use the flashlight.
*/
@Override
public void use() {
    System.out.println("Using " + name + " flashlight");
}

/**
* Print information about the flashlight.
*/
@Override
public void print() {
    System.out.println("Flashlight: " + name + ", weights: " + weight);
}
}

```

Файл BedsideTable.java

```

package ki306.yarema.lab7;

import java.util.ArrayList;

/**
* The BedsideTable class represents a container for items of a specific type that
* implement the Item interface.
* It allows adding, retrieving, finding the maximum-weight item, and using items.
*
* @param <T> The type of items that can be stored in the bedside table, which must
* implement the Item interface.
*
* @author Yarema Maksym
* @version 1.0
*/
public class BedsideTable<T extends Item> {

    private ArrayList<T> array;

    /**
    * Constructs a new BedsideTable object.
    * Initializes an empty array to store items.
    */
    public BedsideTable() {
        array = new ArrayList<T>();
    }

    /**
    * Adds an item to the bedside table and prints a message indicating the addition.
    *
    * @param item The item to add to the bedside table.
    */
    public void putItem(T item) {
        array.add(item);
        System.out.print("Item added: ");
        item.print();
    }

    /**
    * Retrieves an item from the bedside table by its index.
    *
    * @param i The index of the item to retrieve.
    * @return The item at the specified index, or null if the index is out of bounds.
    */
    public T getItem(int i) {

```

```

return array.get(i);
}

/**
 * Finds and returns the item with the maximum weight in the bedside table.
 *
 * @return The item with the maximum weight, or null if the bedside table is empty.
 */
public T getMax() {
    if (!array.isEmpty()) {
        T max = array.get(0);
        for (int i = 1; i < array.size(); i++) {
            if (array.get(i).compareTo(max) > 0) {
                max = array.get(i);
            }
        }
        return max;
    }
    return null;
}

/**
 * Uses an item in the bedside table by its index.
 *
 * @param i The index of the item to use.
 */
public void useItem(int i) {
    array.get(i).use();
}
}

```

Результат виконання програми

```


Main x
C:\Users\Xsakon\.jdk\corretto-18.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.3.
Item added: Book: Atomic Habits, weights: 1.852
Item added: Flashlight: Mobile flashlight, weights: 0.13
Item added: Book: The Catcher in the Rye, weights: 1.243
Item added: Flashlight: Endless light +Ultra, weights: 1.726
Book: The Catcher in the Rye, weights: 1.243
Using Endless light +Ultra flashlight

The heaviest item in bedside table is:
Book: Atomic Habits, weights: 1.852

Process finished with exit code 0
|

```

Фрагмент згенерованої документації



SEARCH:

Package ki306.yarema.lab7

package ki306.yarema.lab7

All Classes and Interfaces

Interfaces

Classes

Class	Description
BedsideTable <T extends Item>	The BedsideTable class represents a container for items of a specific type that implement the Item interface.
Book	The Book class represents a book item that implements the Item interface.
Flashlight	The Flashlight class represents a flashlight item that implements the Item interface.
Item	The Item interface represents items that can be used and have weight.
Main	Class EquationsApp Implements driver for Equations class

Відповіді на контрольні запитання

- 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.
- Розкрийте синтаксис визначення простого параметризованого класу.
 - ```
public class НазваКласу<параметризованийТип> {
 // Тіло класу
}
```
- Розкрийте синтаксис створення об'єкту параметризованого класу.
  - ```
НазваКласу<перелікТипів> зміннаКласу = new  
НазваКласу<перелікТипів>(параметри);
```
- Розкрийте синтаксис визначення параметризованого методу.

- `public <параметризованийТип> типПовернення
назваМетоду(параметри) {
 // Тіло методу
}`
5. Розкрийте синтаксис виклику параметризованого методу.
 - `(НазваКласу|НазваОб'єкту).<перелікТипів>назваМетоду(параметри);`
 6. 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.
 7. Як встановити обмеження для змінних типів?
 - за допомогою ключового слова `extends` для суперкласу або інтерфейсу, від яких має походити реальний тип.
 8. 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.
 9. 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.
 10. 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.