Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Звіт з лабораторної роботи №7

з дисципліни «Кросплатформні засоби програмування»

на тему: «ПАРАМЕТРИЗОВАНЕ ПРОГРАМУВАННЯ»

Виконав: ст.групи КІ-36

Борисов І. С.

Прийняв, та перевірив :

Іванов Ю. С.

Львів 2022

Мета: оволодіти навиками параметризованого програмування мовою Java.

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 83 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.

Варіант №13

13. Словник (тип даних)

Хід роботи:

Код програми:

Main.java

```
public class Main {
    public static void main(String[] args) {
        Vocabulary <? super Words> MyVocabulary = new Vocabulary <Words>(); //
створення нового обекту калсу

        MyVocabulary.AddData(new ProWord("Car", "кар" , "автомобіль"));
        MyVocabulary.AddData(new Word("Computer"    , "Комп'ютер"));
        MyVocabulary.AddData(new Word("Laptop"   , "ноутбук"));
        MyVocabulary.AddData(new ProWord("Mouse", "маус" , "миша"));

        Words res = MyVocabulary.findMax();
        System.out.print("The longest word: \n");
        res.print();
```

```
    }
}
```

Vocabulary.java

```
class Vocabulary <T extends Words>{    //параметризований клас
    private ArrayList<T> arr;

    public Vocabulary(){  //конструктор
        arr = new ArrayList<T>();
    }

    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;
    }

    public void AddData(T data){
        arr.add(data);
        System.out.print("Element added: ");
        data.print();
    }
}
```

Words.java

```
// Це інтерфейся який описує 2 методи

interface Words extends Comparable<Words>{
    public int get_WordLenght();
    public void print();
}
```

ProWord.java

```
class ProWord implements Words
{
    private String Word;
    private int WordSize;
    private String Translations;
    private String Transcript;

    public ProWord(String Word, String Transcript, String Translations){ //
конструктор
        this.Word = Word;
        this.Transcript = Transcript;
        this.Translations = Translations;
        WordSize = Word.length();
    }

    // Метод повертає слово

    public String get_Word(){
```

```
        return Word;
    }

    // Метод встановлює значення поля Word

    public void set_Word(String Word){
        this.Word = Word;
    }

     // Метод повертає довжину слова

    public int get_WordLenght(){
        WordSize = Word.length();
        return Word.length();
    }

     // Метод приймає значення Transcript

    public void set_Transcrip (String Transcript) {
        this.Transcript = Transcript;
    }

     // метод повертає значення параметра Transcript

    public String get_Transcript() {
        return Transcript;
    }

    // Метод приймає значення Translations

    public void  setTranslations(String Translations) {
        this.Translations = Translations;
    }

     // метод повертає значення параметра Translations

    public String getTranslations(){
        return Translations;
    }

    public int compareTo(Words p){
        Integer s = Word.length();
        return s.compareTo(p.get_WordLenght());
    }

     // Вивід інформації про слово

    public void print(){
        System.out.print("Word: " + Word + ", Size: " + WordSize + ",
Translations: " + Translations + ", Transcript: " + Transcript + ";\n");
    }
}
```

Word.java

```
//Клас моделює словник без транскрипції

class Word implements Words {
    private String Word;
    private int WordSize;
    private String Translation;

    public Word(String Word, String Translation){
        this.Word = Word;
        this.Translation = Translation;
```

```
        WordSize = Word.length();
    }

     // метод повертає значеня поля Word

    public String getWord(){
        return Word;
    }

     //  метод встановлює значеня поля Word

    public void setWord(String w){
        Word = w;
    }

     // метод встановлює значеня поля Translation

    public void SetTranslations(String Translation){
        this.Translation= Translation;
    }

     // Метод повертає значення поля Translation

    public String getTranslations(){
        return Translation;
    }

    public int compareTo(Words p){
        Integer s = WordSize;
        return s.compareTo(p.get_WordLenght());
    }

     // метод повертає значення поля WordSize

    public int get_WordLenght() {

        WordSize = Word.length();
        return WordSize;
    }

     // Мето виводить інформацію про стлово

    public void print(){
        System.out.print("Word: " + Word + ", Word Size: " + WordSize + ",
Translations: " + Translation + ";\n");
    }
}
```

Результат виконання:

```
Element added: Word: Car, Size: 3, Translations: автомобіль, Transcript: кар;
Element added: Word: Computer, Word Size: 8, Translations: Комп'ютер;
Element added: Word: Laptop, Word Size: 6, Translations: ноутбук;
Element added: Word: Mouse, Size: 5, Translations: миша, Transcript: маус;
The longest word:
Word: Computer, Word Size: 8, Translations: Комп'ютер;

Process finished with exit code 0
```

Контрольні питання

1. Дайте визначення терміну «параметризоване програмування».

Параметризоване програмування - це такий підхід до опису даних і алгоритмів, який дозволяє їх використовувати з різними типами даних без зміни їх опису.

2. Розкрийте синтаксис визначення простого параметризованого класу.

[public] class НазваКласу <параметризованийТип{,параметризованийТип}>
{…}

3. Розкрийте синтаксис створення об'єкту параметризованого класу.

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

4. Розкрийте синтаксис визначення параметризованого методу.

Модифікатори<параметризованийТип{,параметризованийТип}>типПовернення назваМетоду(параметри);

5. Розкрийте синтаксис виклику параметризованого методу.

Модифікатори<параметризованийТип{,параметризованийТип}>типПовернення назваМетоду(параметри);

6. Яку роль відіграє встановлення обмежень для змінних типів?

Може бути ситуація, коли метод у процесі роботи викликає з-під об'єкта параметризованого типу метод, що визначається у деякому інтерфейсі. У такому випадку немає ніякої гарантії, що цей метод буде реалізований у кожному класі, що передається через змінну типу. Щоб вирішити цю проблему у мові Java можна задати обмеження на множину можливих типів, що можуть бути підставлені замість параметризованого типу.

7. Як встановити обмеження для змінних типів?

Модифікатори<параметризований тип extends обмежуючийТип {& обмежуючий тип} {, параметризований тип extends обмежуючийТип {& обмежуючий тип} }>

типПовернення назваМетоду(параметри);

8. Розкрийте правила спадкування параметризованих типів.

1. Всі класи, що утворені з одного і того ж параметризованого класу з використанням різних значень змінних типів є незалежними навіть якщо між цими типами є залежність спадкування.

2. Завжди можна перетворити параметризований клас у «сирий» клас, при роботі з яким захист від некоректного коду є значно слабшим, що дозволяє здійснювати небезпечні присвоєння об’єктів параметризованого класу об’єктам «сирого» класу

3. Параметризовані класи можуть розширювати або реалізовувати інші параметризовані класи.

9. Яке призначення підстановочних типів?

Підстановочні типи дозволяють враховувати залежності між типами, що виступають параметрами для параметризованих типів. Це в свою чергу дозволяє застосовувати обмеження для параметрів, що підставляються замість параметризованих типів. Завдяки цьому підвищується надійність параметризованого коду, полегшується робота з ним та розділяється використання безпечних методів доступу і небезпечних модифікуючих методів.

10.Застосування підстановочних типів.

Підстановочні типи застосовуються у вигляді параметру типу, що передається у трикутних дужках при утворені реального типу з параметризованого типу, наприклад, у методі main.

Висновок: на даній лабораторній роботі я оволодів навиками параметризованого програмування мовою Java.