Державний університет «Одеська Політехніка»
Інститут комп’ютерних систем
Кафедра інформаційних систем

Лабораторна робота №7
з дисципліни «Об’єктно-орієнтоване програмування»
Тема: «Лямбда-выражения. Параметризация поведения»

Виконав:
студентка групи АІ-204
Сіренко М.О

Прийняв:
доц. Годовиченко М.А.

Одеса 2021

Цели лабораторной работы:

- изучить процесс создания и сценарии использования анонимных объектов;
- научиться создавать анонимные классы;
- разобраться с созданием одиночных и блочных лямбда-выражения.

Ход работы:

Задание 1

Напишем функцию для фильтрации массиваю в заисимости от заданного предиката. Напишем предикат, который возвращает `true`, если число простое.

Для проверки генерируем массив из 1000 случайных целых чисел и профельтруем его. Выведем массив до и после фильтра.

Метод filter (принимает в себя предикат):

```
public static int[] filter(int[] input, Predicate p) {
        int[] result = new int[input.length];

        int counter = 0;
        for (int i : input) {
            if (p.test(i)) {
                result[counter] = i;
                counter++;
            }
        }

        return Arrays.copyOfRange(result, 0, counter);
    }
```

Код Main:

```
public static void main(String[] args) {

        final int ARRAY_SIZE = 1000;

        int[] array = new int[ARRAY_SIZE];


        //fill array with random numbers (from 1 to 100)

        for (int i = 0; i < ARRAY_SIZE; ++i) {

            array[i] = (int) (Math.random( ) * 10000) + 1;
```

```
        }

        System.out.println(Arrays.toString(array));

        // we pass the block lambda expression of the predicate to the method
        int[] res = filter(array, o -> {
            int value = (int) o;
            for (int i = 2; i <= Math.sqrt(value); i++) {
                if (value % i == 0)
                    return false;
            }
            return true;
        });

        System.out.println(Arrays.toString(res));

    }
```

Пример результата отработки программы

```
/snap/intellij-idea-community/330/jbr/bin/java -javaagent:/snap/intellij-idea-community/330/lib/idea_rt.jar=33299:/snap/intellij-idea-community/330/bin -Dfile.encoding=UTF-8 -classpath /home/masha/Univer/OOP/lab7/
[5254, 1591, 6891, 1505, 2705, 2519, 7943, 1364, 901, 1471, 3037, 4207, 8850, 1426, 4903, 7041, 3353, 5428, 4640, 531, 8854, 3140, 9915, 9827, 9730, 6166, 6782, 8063, 319, 8028, 4913, 1013, 2329, 5925, 7391, 975, 
[1471, 3037, 4903, 1013, 2803, 1123, 6091, 3631, 3529, 967, 5233, 9533, 3089, 1811, 8849, 2267, 2269, 4999, 1439, 7559, 6359, 787, 6427, 9859, 7723, 937, 6287, 4591, 5479, 7789, 7583, 8783, 643, 6691, 617, 9161, 5

Process finished with exit code 0
```

Задание 2

Создадим класс Student который хранит в себе имя студента, группу и оценки.Напишем метод фильтрации массива студентов. Для проверки метода фильтрации напишем предикат, который отсеивает студентов, которые имеют 1 и более задолженности (оценка меньше 60).

Класс Student

```
import java.util.Arrays;
```

```
public class Student {
    private String name;
    private String group;

    private int[] marks;

    public Student(String name, String group, int[] marks) {
        this.name = name;
        this.group = group;
        this.marks = marks;
    }


    public String getName() {
        return name;
    }

    public String getGroup() {
        return group;
    }

    public int[] getMarks() {
        return marks;
    }

}
```

Метод filter (принимает в себя предикат):

```
public static Student[] filter(Student[] input, Predicate p) {
        Student[] result = new Student[input.length];

        int counter = 0;
        for (Student i : input) {
            if (p.test(i)) {
                result[counter] = i;
                counter++;
            }
        }

        return Arrays.copyOfRange(result, 0, counter);
    }
```

Предикат для фильтрации

```
//фильтр студентов не сдавших одни и больше предметов
        Predicate<Student> isDebtor = o -> {

            for (int i : o.getMarks()) {
                if (i < 60) return false;
            }
            return true;
        };
```

Код Main:

```
public static void main(String[] args) {

        Student[] group = {
                  new Student("Harry Potter", "Gryffindor", new int[]{75, 90, 90,
100, 60, 75, 40, 75, 75}),
                    new Student("Hermione Granger", "Gryffindor", new int[]{100,
100, 100, 90, 100, 100, 100, 100, 100}),
                  new Student("Draco Malfoy", "Slytherin", new int[]{75, 60, 100,
90, 60, 75, 90, 40, 75}),
                  new Student("Ron Weasley", "Gryffindor", new int[]{100, 75, 90,
90, 60, 40, 40, 75, 75}),
                   new Student("Cedric Diggory", "Hufflepuff", new int[]{90, 100,
90, 100, 90, 75, 90, 75, 75})
        };

        //фильтр студентов не сдавших одни и больше предметов
        Predicate<Student> isDebtor = o -> {

            for (int i : o.getMarks()) {
                if (i < 60) return false;
            }
            return true;
        };

        //для вывода
           Consumer<Student> c1 = o -> System.out.println(o.getName() + " " +
o.getGroup() + " " + Arrays.toString(o.getMarks()));

        Student[] res = filter(group, isDebtor);
        forEach(res, c1);



    }
```

Результат отработки программы на азданном массиве

```
/snap/intellij-idea-community/330/jbr/bin/java -javaagent:/snap/intellij-ide
Hermione Granger Gryffindor [100, 100, 100, 90, 100, 100, 100, 100, 100]
Cedric Diggory Hufflepuff [90, 100, 90, 100, 90, 75, 90, 75, 75]
```

Задание 3

Напишем метод фильтрации по двум условиям . Элемент проходит через фильтр, если удовлетворяет оба условия. Используем код из предыдущего задания

Метод filter (принимает в себя предикаты):

```
public static Student[] filter(Student[] input, Predicate p1, Predicate p2) {
        Student[] result = new Student[input.length];

        int counter = 0;
        for (Student i : input) {
            if (p1.test(i) && p2.test(i)) {
                result[counter] = i;
                counter++;
            }
        }

        return Arrays.copyOfRange(result, 0, counter);
    }
```

Предикаты для фильтрации(по оценкам)

```
//фильтр студентов не сдавших одни и больше предметов
        Predicate<Student> isDebtor = o -> {

            for (int i : o.getMarks()) {
                if (i < 60) return false;
            }
            return true;
        };
```

Предикаты для фильтрации(по группе)

```
 //фильтр студентов не из гриффиндора
        Predicate<Student> isGryff = o -> {

            if (!o.getGroup().equals("Gryffindor")) return false;

            return true;
        };
```

Код Main:

```
public static void main(String[] args) {

        Student[] group = {
                  new Student("Harry Potter", "Gryffindor", new int[]{75, 90, 90,
100, 60, 75, 40, 75, 75}),
                     new Student("Hermione Granger", "Gryffindor", new int[]{100,
100, 100, 90, 100, 100, 100, 100, 100}),
                  new Student("Draco Malfoy", "Slytherin", new int[]{75, 60, 100,
90, 60, 75, 90, 40, 75}),
                   new Student("Ron Weasley", "Gryffindor", new int[]{100, 75, 90,
90, 60, 40, 40, 75, 75}),
                    new Student("Cedric Diggory", "Hufflepuff", new int[]{90, 100,
```

```
90, 100, 90, 75, 90, 75, 75})
        };

        //фильтр студентов не сдавших одни и больше предметов
        Predicate<Student> isDebtor = o -> {

            for (int i : o.getMarks()) {
                if (i < 60) return false;
            }
            return true;
        };
        //фильтр студентов не из гриффиндора
        Predicate<Student> isGryff = o -> {

            if (!o.getGroup().equals("Gryffindor")) return false;

            return true;
        };

        //для вывода
           Consumer<Student> c1 = o -> System.out.println(o.getName() + " " +
o.getGroup() + " " + Arrays.toString(o.getMarks()));


        Student[] res = filter(group, isDebtor,isGryff);
        forEach(res, c1);


    }
```

Результат работы программы на заданном массиве

```
/snap/intellij-idea-community/330/jbr/bin/java -javaagent:/snap/intellij-ide
Hermione Granger Gryffindor [100, 100, 100, 90, 100, 100, 100, 100, 100]


Process finished with exit code 0
```

Задание 4

Напишем `Consumer`, который принимает на вход объект типа `Student` и выводит в консоль строку вида `ФАМИЛИЯ + ИМЯ`. Создаим массив из нескольких студентов и проверим работу функции `forEach()`

Класс Student

```
public class Student {
    String name;
```

```
    String surname;

    public Student(String name, String surname) {
        this.name = name;
        this.surname = surname;
    }

    public String getName() {
        return name;
    }

    public String getSurname() {
        return surname;
    }


}
```

Код класса Main

```
public class Main {

    public static void main(String[] args) {
        Student[] group = {
                new Student("Harry", "Potter"),
                new Student("Hermione", "Granger"),
                new Student("Draco", "Malfoy"),
                new Student("Ron", "Weasley"),
                new Student("Cedric", "Diggory")
        };

          Consumer<Student> c1 = o -> System.out.println(o.getSurname() + " " +
o.getName());

        forEach(group,c1);
    }

    public static void forEach(Student[] input, Consumer action) {
        for (Student i : input) {
            action.accept(i);
        }
    }
}
```

Результат работы программы на заданном массиве

```
Potter Harry
Granger Hermione
Malfoy Draco
Weasley Ron
Diggory Cedric
```

Задание 5

Напишием метод, который принимает `Predicate` и `Consumer`. Действие в `Consumer` выполняется только, если условие в `Predicate` выполняется. Создадим массив из целых чисел и проверим работу этого метода.

Метод для фильтрации:

```
public static void printIf(int[] input, Predicate p, Consumer c) {
        for (int i : input) {
            if (p.test(i)) {
                c.accept(i);
            }
        }
    }
```

Код Main (условие предиката — проверка на четность)

```
 public static void main(String[] args) {
        final int ARRAY_SIZE = 10;
        int[] array = new int[ARRAY_SIZE];

        //fill array with random numbers (from 1 to 100)
        for (int i = 0; i < ARRAY_SIZE; ++i) {
            array[i] = (int) (Math.random() * 100) + 1;
        }

        System.out.println("Before: " + Arrays.toString(array));

        //filter array from odd
        System.out.print("Filter: ");
        printIf(array, o -> {
            int value = (int) o;
            return value % 2 == 0;
        }, (i) -> System.out.print(i + " "));
    }
```

Результат работы программы на заданном массиве

```
/snap/intellij-idea-community/330/jbr/bin/java -javaagent:
Before: [28, 87, 94, 99, 25, 32, 25, 78, 57, 57]
Filter: 28 94 32 78
```

Задание 6

Напишем `Function`, который принимает на вход целое число `N` и возвращает целое число `2^N`. Создадим массив из 10 целых чисел и проверим работу метода.

Функция которая принимает `Function`

```
public static int[] processArray(int[] input, Function function) {
        int[] result = new int[input.length];

        for (int i = 0; i < input.length; i++)
            result[i] = (int) function.apply(input[i]);

        return result;
    }
```

Код Main (`Function` возвращает число во 2 степени)

```
public static void main(String[] args) {
        final int ARRAY_SIZE = 10;
        int[] array = new int[ARRAY_SIZE];

        //fill array with random numbers (from 1 to 100)
        for (int i = 0; i < ARRAY_SIZE; ++i) {
            array[i] = (int) (Math.random() * 100) + 1;
        }

        System.out.println(Arrays.toString(array));


        Function function = o -> {
            int value = (int) o;
            return value * value;
        };

        int[] result = processArray(array, function);
        System.out.println(Arrays.toString(result));


    }
```

Результат работы программы на заданном массиве

```
/snap/intellij-idea-community/330/jbr/bin/java -javaagent:/snap/in
[7, 33, 3, 79, 50, 2, 41, 3, 31, 84]
[49, 1089, 9, 6241, 2500, 4, 1681, 9, 961, 7056]
```

Задание 7

Напишем метод `stringify()`, который принимает на вход массив целых чисел от 0 до 9 и `Function` который принимает на вход целое число от 0 до 9 и возвращает его значение в виде строки ("ноль", "один", "два", "три" и так далее).

Код `Function`

```
Function<Integer, String> function = o -> {
            switch (o) {
                case 0:
                    return "Zero";
                case 1:
                    return "One";
                case 2:
                    return "Two";
                case 3:
                    return "Three";

                case 4:
                    return "Four";

                case 5:
                    return "Five";

                case 6:
                    return "Six";

                case 7:
                    return "Seven";

                case 8:
                    return "Eight";

                case 9:
                    return "Nine";

                default:
                    return "";
            }
        };
```

Код метода `stringify()`

```
public static String stringify(int[] input, Function function) {
        String res = "";
        for (int i : input) {
            res = res.concat((String) function.apply(input[i]) + " ");
        }
        return res;
    }
```

Код Main

```
 public static void main(String[] args) {

        int[] array = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
        Function<Integer, String> function = o -> {
            switch (o) {
```

```
                case 0:
                    return "Zero";
                case 1:
                    return "One";
                case 2:
                    return "Two";
                case 3:
                    return "Three";

                case 4:
                    return "Four";

                case 5:
                    return "Five";

                case 6:
                    return "Six";

                case 7:
                    return "Seven";

                case 8:
                    return "Eight";

                case 9:
                    return "Nine";

                default:
                    return "";
            }
        };
        System.out.println(stringify(array, function));

    }
```

Результат отработки программы с заданным набором данных:

```
/snap/intellij-idea-community/330/jbr/bin/java -java
Zero One Two Three Four Five Six Seven Eight Nine


Process finished with exit code 0
```

Дополнительное задание

Перепишем выполненное приложение **SortingList** из 6 лабораторной работы, чтобы оно использовало механизм лямбда-выражений.

Переписанный код с использованием лямбда-выражений для сортировки списка студентов по определенным признакам.

```
 //массив для хранения состояния нажатия кнопки (какой вид сортировки)
        final int[] sortType = {1, 1, 1};

        // Обработка нажатия на кнопку сортировки по имени
        sortByNameButton.setOnAction(event -> {
            students.sort((o1, o2) -> {
                if (o1 != null && o2 != null) {
                    return sortType[0] * o1.getName().compareTo(o2.getName());
                }
                return 0;
            });
            sortType[0] *= (-1);
        });

        // Обработка нажатия на кнопку сортировки по фамилии
        sortByLastNameButton.setOnAction(event -> {
            students.sort((o1, o2) -> {
                if (o1 != null && o2 != null) {
                                                        return   sortType[1]   *
o1.getLastName().compareTo(o2.getLastName());
                }
                return 0;

            });
            sortType[1] *= (-1);
        });

        // Обработка нажатия на кнопку сортировки по среднему балу
        sortByMarkButton.setOnAction(event -> {
            students.sort((o1, o2) -> {
                if (o1 != null && o2 != null) {
                                return  sortType[2]  *  (int)  (o1.getAvgMark()  -
o2.getAvgMark());
                }
                return 0;
            });
            sortType[2] *= (-1);

        });
```

Выводы к лабораторной работе:

В рамках выполнения данной лабораторной работы было изучить процесс создания и сценарии использования анонимных объектов, создания анонимных классов и одиночных и блочных лямбда-выражений.

Были выполнены задания задания на проработку материала. Никаких проблем с их выполнением не возникло.