Отчёт по лабораторной работе 3

Студент: Курочкин Е. А.

Группа: ПИМ-22

1. Постановка задачи

1. ООП.
    a. Создать интерфейс.
    b. Создать абстрактный класс.
    c. Создать класс, имплементирующий интерфейс.
    d. Создать класс-наследник абстрактного класса.
2. Reflection
    a. Выгрузить все поля и методы класса с помощью рефлексии.
    b. Вызвать несколько методов класса.
    c. Вывести на экран всех предков класса.
3. Collections
    a. Ознакомится со всеми коллекциями java (list, set, map) и их реализацией.
    b. Продемонстрировать в программе работу с каждым видом реализации коллекции (list, set, map).
4. Generics
    a. Сделать дженерик на класс.
    b. Сделать дженерик на метод.

2. Структура проекта

Проект разделён на следующие директории

1. **src** - папка, в которой находятся *.java* файлы
2. **out.production.lab3** - папка, в которой находятся *.class* файлы

3. Выполнение задания

1. ООП.
    a. Был создан интерфейс под названием *Human*:

```
public interface Human {
public void showAge(int x);
public void showEducation(String s);
public void showGroup(String n);
public void showUniversity(String u);
```

```
}
```

b. Был создан абстрактный класс под названием *Transport*:

```
public abstract class Transport {
    public abstract void makeSound();
    public abstract void showName();
    public void responsible(){
        System.out.println("shovel");
    }
}
```

c. Класс *Student*, имплементирующий интерфейс *Human*:

```
public class Student implements Human{
    public void showAge(int x) {
        System.out.println(x);
    }
    public void showEducation(String s) {
        System.out.println(s);
    }
    public void showGroup(String n) {
        System.out.println(n);
    }
    public void showUniversity(String u){
        System.out.println(u);
    }
    public void showInfo(int x, String s,String n, String u){
        showAge(x);
        showEducation(s);
        showGroup(n);
        showUniversity(u);
    }
}
```

d. Класс *Tractor* - наследник класса *Transport*:

```
public class Tractor extends Transport{
    public void makeSound(){
        System.out.println("Trtrtrtrtrtr");
    }
    public void showName() {
        System.out.println("Tractor");
```

```
    }
```

```
}
```

e. Класс *Main* показывающий работу классов, описанных выше

```
public class Main {
    public static void main(String[] args){
        Student st = new Student();
        st.showInfo(22, "Magistracy","ПИМ-22", "RSATU");
        Tractor tr = new Tractor();
        System.out.print("Red ");
        tr.showName();
        tr.makeSound();
        tr.responsible();
```

```
        Reflect.dumpEverything(tr.getClass().getName());
        Reflect.dumpEverything(tr.getClass().getSuperclass().getName());
    }
}
```

f. Результат работы:

```
22
Magistracy
ПИМ-22
RSATU
Red Tractor
Trtrtrtrtrtr
shovel
```

2. Reflection

a. Был создан класс *Reflect*, код которого решает поставленные задачи

```
import java.lang.reflect.*;
public class Reflect {
    public static void dumpEverything(String className) {
        try {
            Class<?> c = Class.forName(className);
            System.out.println("Name:  "+className);
            Method[] m = c.getMethods();
            System.out.println("Methods:");
            for (Method method : m)
                System.out.println(method.toString());
            Field[] f = c.getDeclaredFields();
            System.out.println("Fields:");
            for (Field field : f)
                System.out.println(field.toString());
```

```
        }
        catch (Throwable err) {
            System.err.println(err);
        }
    }
}
```

b. Результат работы рефлексии

```
Name:  Tractor
Methods:
public void Tractor.makeSound()
public void Tractor.showName()
public void Transport.responsible()
public final void java.lang.Object.wait(long,int) throws java.lang
.InterruptedException
public final void java.lang.Object.wait() throws java.lang.InterruptedException
public final native void java.lang.Object.wait(long) throws java.lang
.InterruptedException
public boolean java.lang.Object.equals(java.lang.Object)
public java.lang.String java.lang.Object.toString()
public native int java.lang.Object.hashCode()
public final native java.lang.Class java.lang.Object.getClass()
public final native void java.lang.Object.notify()
public final native void java.lang.Object.notifyAll()
```

3. Collections

a. Был создан класс *Map_set_list*, в котором использованы *map*, *set* и *list*

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
public class Map_set_list {
    public static String getCharForNumber(int i) {//функция чтобы найти букву по
номеру
        return i > 0 && i < 27 ? String.valueOf((char)(i + 64)) : null;
    }
    public static void main(String[] args) {
        //Map
        HashMap<Integer, String> numAndLetter = new HashMap<>();
        for (int i = 1; i <= 5; i++) {
            numAndLetter.put(i, getCharForNumber(i));//заполнение
        }
        //Set
        HashSet<Integer> numbs = new HashSet<Integer>(numAndLetter.keySet
());//создание сета и одновременно выборка данных из мапы
        System.out.println(numbs);
        //List
```

```
        ArrayList<String> letters = new ArrayList<>(numAndLetter.values
());//создание листа и одновременно выборка данных из мапы
        System.out.println(letters);
        System.out.println("Размер мапы = " + numAndLetter.size());
        numAndLetter.clear();
        System.out.println("Пусто = "+numAndLetter.isEmpty());
        for (Integer i = 6; i < 8; i++) {
            numbs.add(i);//добавление
        }
        System.out.println(numbs);
        for (Integer i = 1; i <= 5; i++) {
            numbs.remove(i);//удаление
        }
        System.out.println(numbs);
        if (numbs.add(8)){//попытка добавить число, которое уже есть
            System.out.println("Добавление.");
        }else{
            System.out.println("Такое число уже есть.");
        }
        int lSize = letters.size();//размер листа
        System.out.println("Размер списка = "+lSize);
        for (Integer i=1; i<3; i++){
            letters.remove(getCharForNumber(i));//удаление элементов
            letters.add(getCharForNumber(i));//добавление элементов
        }
        System.out.println(letters);
    }
}
```

b. Результат работы программы:

```
-
[1, 2, 3, 4, 5]
[A, B, C, D, E]
Размер мапы = 5
Пусто = true
[1, 2, 3, 4, 5, 6, 7]
[6, 7]
Добавление.
Размер списка = 5
[C, D, E, A, B]
-
```

4. Generics

a. Был создан *generic* class с названием *Generic*

```
public class Generic <T>{
    private T id;
```

```
    Generic(T id){
        this.id = id;
    }
    public T getId(){
        return id;
    }
}
```

b. В следующем классе, *GenMeth*, был сделан *generic* метод, а также, в нём вызываются методы из *Generic*.

```
public class GenMeth {
    public static <T> void printT(T[] items){
        for (T item: items){
            System.out.print(item + " ");
        }
    }
    public static void main (String args[]){
        Generic <String> GenClass1 = new Generic<String>("lr_3");
        String a = GenClass1.getId();
        System.out.println(a);
        Generic <Integer> GenClass2 = new Generic<Integer>(1);
        Integer b = GenClass2.getId();
        System.out.println(b);
        GenMeth gm = new GenMeth();
        String[] chars = {"K","E","K","LoL"};
        Integer[] numbs = {228,22};
        gm.printT(chars);
        gm.printT(numbs);
    }
}
```

c. Результат работы:

```
lr_3
1
K E K LoL 228 22
```

6. Вывод

В результате выполнения лабораторной работы были изучены интерфейсы и абстрактные классы, метод *Reflection*, знакомство с коллекциями *java* и был создан *generic* класс и *generic* метод и протестированы их вызовы.