Отчёт по лабораторной работе 3

Студент: Вороненков Николай Вадимович

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. Информация о реализации

Для выполнения задания использовалась среда разработки IntelliJ IDEA Community Edition 2022.2.3.

4. Выполнение задания

1. ООП.
    a. Был создан интерфейс под названием Car:

```
public interface Car {
    public void showDate(int x);
    public void showFuel(String s);
    public void showCountry(String n);
    public void showType(String n);
}
```

b. Был создан абстрактный класс под названием Human:

```
public abstract class Human {
    public abstract void showStudGoes();
    public abstract void showStudCar();
    public void goFinish(){
        System.out.println("Студен приехал в вуз");
    }
}
```

c. Класс Car, имплементирующий интерфейс Lada:

```
public class Lada implements Car {
    public void showDate(int x) {
        System.out.println(x);
    }
    public void showFuel(String s) {
        System.out.println(s);
    }
    public void showCountry(String n) {
        System.out.println(n);
    }
    public void showType(String u){
        System.out.println(u);
    }
    public void showInfo(int x, String s,String n, String u){
        showDate(x);
        showFuel(s);
        showCountry(n);
        showType(u);
    }
}
```

d. Класс Student - наследник класса Human:

```
public class Student extends Human {
    public void showStudGoes(){
        System.out.println("Студент в пути");
    }
    public void showStudCar() {
```

```
        System.out.println("Студент сел в машину");
```

```
    }
}
```

e. Класс Main показывающий работу классов, описанных выше, а так же запускающий в работу класс с рефлексией

```
public class Main {
    public Main() {
    }
```

```
    public static void main(String[] args) {
        Lada lada = new Lada();
        lada.showInfo(1999, "АИ-92 ", "Russian", "Легковое авто");
        Student stud = new Student();
        System.out.println("Машина подходит студенту");
        stud.showStudCar();
        stud.showStudGoes();
        stud.goFinish();
        System.out.println("----------------------------
Reflector----------------------------");
        Reflector.dumpEverything(lada.getClass().getName());
        Reflector.dumpEverything(lada.getClass().getSuperclass().getName());
    }
}
```

f. Результат работы:

```
1999
АИ-92
Russian
Легковое авто
Машина подходит студенту
Студент сел в машину
Студент в пути
Студен приехал в вуз
----------------------------Reflector----------------------------
----------------------------Lada----------------------------
----------------------------Methods----------------------------
public void Lada.showDate(int)
public void Lada.showType(java.lang.String)
public void
Lada.showInfo(int,java.lang.String,java.lang.String,java.lang.String)
public void Lada.showFuel(java.lang.String)
public void Lada.showCountry(java.lang.String)
```

```
public final void java.lang.Object.wait() throws java.lang.InterruptedException
public final void java.lang.Object.wait(long,int) 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()
----------------------------Fields----------------------------
----------------------------java.lang.Object----------------------------
----------------------------Methods----------------------------
public final void java.lang.Object.wait() throws java.lang.InterruptedException
public final void java.lang.Object.wait(long,int) 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()
```

2. Reflection

   a. Был создан класс Reflector, код которого решает поставленные задачи

```
import java.lang.reflect.*;
public class Reflector {
    public static void dumpEverything(String className) {
        try {
            Class<?> c = Class.forName(className);
            System.out.println("----------------------------"+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 e) {
            System.err.println(e);
        }
    }
}
```

a. Результат работы рефлексии показан в результате работы класса Main

3. Collections

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

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
public class TaskMap {
    public static String getAlphabet(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) {
        HashMap<Integer, String> myMap = new HashMap<>();
        for (int i = 1; i < 27; i++) {
            myMap.put(i, getAlphabet(i));
        }
        HashSet<Integer> myHashSet = new HashSet<Integer>(myMap.keySet());
        System.out.println(myHashSet);
```

```
ArrayList<String> myArrayList = new ArrayList<>(myMap.values());
System.out.println(myArrayList);
```

```
System.out.println("Размер массива - " + myMap.size());
myMap.clear();
System.out.println("Проверка на пустоту - "+myMap.isEmpty());
```

```
for (Integer i = 27; i < 31; i++) {
    myHashSet.add(i);
}
System.out.println(myHashSet);
for (Integer i = 1; i < 26; i++) {
    myHashSet.remove(i);
}
System.out.println(myHashSet);
if (myHashSet.add(30)){
    System.out.println("Число добавлено");
}else{
    System.out.println("Такое число уже есть");
```

```
}
```

```
        // ArrayList
        int lSize = myArrayList.size();
        System.out.println("ArrayList - "+lSize);
        for (Integer i=1; i<14; i++){
            myArrayList.remove(getAlphabet(i));
            myArrayList.add(getAlphabet(i));
        }
        System.out.println(myArrayList);
    }
}
```

b. Результат работы программы:

```
-
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25, 26]
[A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z]
Размер массива - 26
Проверка на пустоту - true
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25, 26, 27, 28, 29, 30]
[26, 27, 28, 29, 30]
Такое число уже есть
ArrayList - 26
[N, O, P, Q, R, S, T, U, V, W, X, Y, Z, A, B, C, D, E, F, G, H, I, J, K, L, M]
-
```

4. Generics

a. Был создан *generic* class с названием GenClass

```
public class GenClass<T>{
    private T id;
    GenClass(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 *GenClass'a*.

```
public class GenMeth {
```

```
    public static <T> void printT(T[] items){
        for (T item: items){
            System.out.print(item + " ");
        }
    }
```

```
public static void main (String args[]){
    GenClass <String> myGenClass1 = new GenClass<String>("Привет мир!");
    String a = myGenClass1.getId();
    System.out.println(a);
```

```
GenClass <Integer> myGenClass2 = new GenClass<Integer>(1234567890);
Integer b = myGenClass2.getId();
System.out.println(b);
```

```
        GenMeth gm = new GenMeth();
        String[] chars = {"А","Б","В","Г","Д"};
        Integer[] numbs = {1,2,3,4,5,6,7,8,9,0};
        gm.printT(chars);
        gm.printT(numbs);
    }
}
    .. Результат работы:
```

```
Привет мир!
1234567890
А Б В Г Д 1 2 3 4 5 6 7 8 9 0
```

6. Вывод

В результате выполнения лабораторной были изучены интерфейсы и абстрактные классы, успешно использованы на практике. Были изучены методы рефлексии, и применены на практике. Также было произведено знакомство с map,set,list, а так их реализация в коде. Был создан generic класс и generic метод и протестированы их вызовы.