МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н.Каразіна

Факультет комп’ютерних наук

**Лабораторна робота**

з дисципліни «крос-платформне програмування»

на тему «**Java Reflection»**

|  |
| --- |
| Виконав:  студент групи КУ-21  Черкасов А. І. |
| Перевірив:  Родінко М.Ю. |

ХАРКІВ – 2021

Задание №1

Напишите метод, который по полному имени типа, заданному в виде строки, либо по объекту типа Class, возвращает строку с его полным описанием: имя пакета, в котором класс определен, модификаторы и имя анализируемого класса, его базовый класс, список реализованных интерфейсов, а также список всех полей, конструкторов и методов, объявленных в классе, и их характеристики. Следует предусмотреть, что в программу для анализа могут быть переданы как примитивные типы, так и ссылочные типа: массивы, классы и интерфейсы. Для проверки работы напишите консольную программу и программу с графическим интерфейсом пользователя.

Задание №2

Напишите метод, который по полученному объекту выводит его состояние — список всех полей, объявленных в классе, вместе с их значениями, а также список объявленных в классе открытых методов. Пользователь может просмотреть этот список, выбрать для вызова только методы без параметров, вызвать их на этом объекте и просмотреть результат вызова.

Задание №3

Напишите метод, который получает объект, имя метода в виде строки и список требуемых для вызова метода параметров. Если данный метод может быть вызван на заданном объекте, то вывести результат, иначе выбросить исключение FunctionNotFoundException.

Задание №4

Напишите программу, которая позволяет создавать одномерные массивы и матрицы как примитивных, так и ссылочных типов, указанных во время работы программы. Программа должна уметь изменять размеры массива и матрицы с сохранением значений и преобразовывать массивы и матрицы в строку.

Задание №5

Напишите программу, которая демонстрирует особенности применения «универсальных» динамических объектов прокси для профилирования метода (выводит на экран время вычисления метода) и для трассировки метода (выводит на экран имя, параметры метода и вычисленное значение).

**Задание 1:**

Код:

import java.lang.reflect.Constructor;

import java.lang.reflect.Field;

import java.lang.reflect.Method;

import java.lang.reflect.Modifier;

import java.util.Arrays;

import java.util.Scanner;

public class Task1 {

public static void main(String[] args) {

System.out.println("Enter the name of the class");

Scanner in = new Scanner(System.in);

String name = in.next();

MyClass myClass = new MyClass();

String s = getInfo(name);

String s1 = getInfo(myClass);

int c [] = new int[]{1,2,3};

String s2 = getInfo(c);

System.out.println(s);

System.out.println(s1);

System.out.println(s2);

Task1\_GUI app = new Task1\_GUI();

app.setVisible(true);

}

public static String getInfo(String name){

String s = null;

Class<?> myClass = null;

try {

myClass = Class.forName(name);

} catch (ClassNotFoundException e) {

System.out.println("Class Not Found");

}

if(myClass.getPackage()!=null)

s = myClass.getPackage().toString();

int mod = myClass.getModifiers();

s+="\n" + Modifier.toString(mod)+" " + myClass + "\n";

s+= "extends " + myClass.getSuperclass();

Class[] i = myClass.getInterfaces();

s+= "\nInterfaces:\n" + Arrays.asList(i);

s+="\n\nFields:\n";

Field[] fields1 = myClass.getDeclaredFields();

for(Field field : fields1)

s+=field+ "\n";

s+="\nConstructors:\n";

Constructor c[] = myClass.getConstructors();

for (Constructor constructor : c) {

s+=constructor+"\n";

}

s+="\nMethods:\n";

Method m[] = myClass.getDeclaredMethods();

for (Method method : m) {

s+=method+"\n";

}

return s;

}

public static String getInfo(Object myClass1){

String s;

Class myClass = myClass1.getClass();

if(myClass.getPackage()!=null)

s = myClass.getPackage().toString();

else s = "";

int mod = myClass.getModifiers();

s+="\n" + Modifier.toString(mod)+" " + myClass + "\n";

s+= "extends " + myClass.getSuperclass();

Class[] i = myClass.getInterfaces();

s+= "\nInterfaces:\n" + Arrays.asList(i);

s+="\n\nFields:\n";

Field[] fields1 = myClass.getDeclaredFields();

for(Field field : fields1)

s+=field+ "\n";

s+="\nConstructors:\n";

Constructor c[] = myClass.getConstructors();

for (Constructor constructor : c) {

s+=constructor+"\n";

}

s+="\nMethods:\n";

Method m[] = myClass.getDeclaredMethods();

for (Method method : m) {

s+=method+"\n";

}

return s;

}

//java.lang.String

}

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class Task1\_GUI extends JFrame {

private JButton button = new JButton("Go");

private JTextField input = new JTextField("", 5);

private JLabel label = new JLabel("Full name of the class:");

JTextArea textArea = new JTextArea ();

public Task1\_GUI() {

super("Simple Example");

this.setSize(400, 300);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Container container = this.getContentPane();

//

label.setPreferredSize(new Dimension(300, 20));

JPanel panel = new JPanel();

container.add(panel, BorderLayout.NORTH);

panel.add(label);

panel.add(input);

panel.setLayout(new GridLayout(1,2,1,5));

container.add(textArea, BorderLayout.CENTER);

final JScrollPane scrollPane = new JScrollPane(textArea);

container.add(scrollPane, BorderLayout.CENTER);

scrollPane

.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_ALWAYS);

button.addActionListener(new ButtonEventListener());

container.add(button, BorderLayout.PAGE\_END);

//java.lang.String

}

class ButtonEventListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

String message = "";

message += "Class: " + input.getText() + "\n";

message += Task1.getInfo(input.getText());

textArea.setText(message);

}

}

public static void main(String[] args) {

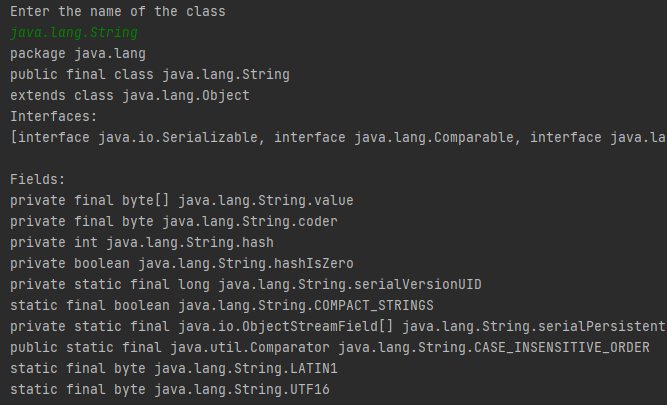
Task1\_GUI app = new Task1\_GUI();

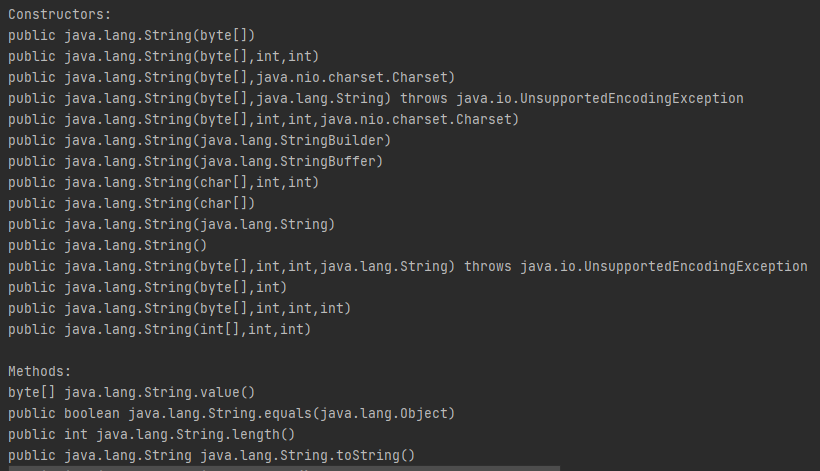
app.setVisible(true);

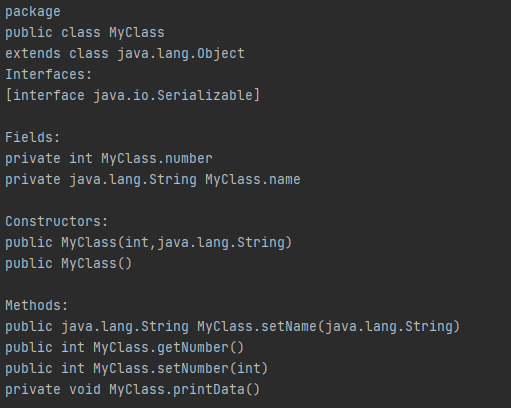
}

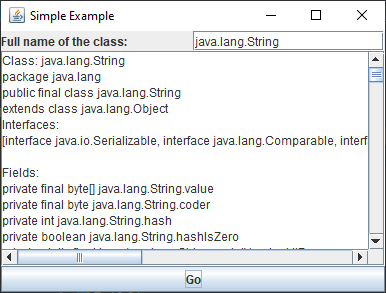
}

Пример работы программы:









**Задание 2 и 3:**

Код:

import java.lang.reflect.Field;

import java.lang.reflect.InvocationTargetException;

import java.lang.reflect.Method;

import java.util.Scanner;

public class Task2\_3 {

public static void task2(Object o) throws IllegalAccessException, InvocationTargetException {

Class<?> myClass = o.getClass();

System.out.println(o.getClass());

Field[] fields = myClass.getDeclaredFields();

for(Field field : fields){

field.setAccessible(true);

System.out.println(field + " " + field.get(o).toString());

}

System.out.println("\nMethods\n");

Method m[] = myClass.getMethods();

for (Method method : m) {

System.out.println(method);

}

System.out.println("Enter the method without parameter:\n");

Scanner in = new Scanner(System.in);

String meth = in.next();

for (Method method : m) {

if(method.getName().equals(meth))

System.out.println(method.invoke(o));

}

}

public static void task3(Object myClass1, String name, Object [] param ) throws FunctionNotFoundException, IllegalAccessException, InvocationTargetException {

Class<?> myClass = myClass1.getClass();

System.out.println("\nTask3\n");

Method m[] = myClass.getDeclaredMethods();

boolean flag = false;

for (Method method : m) {

if (method.getName().equals(name)) {

method.setAccessible(true);

System.out.println(method.invoke(myClass1, param));

flag = true;

}

}

if (!flag) {

throw new FunctionNotFoundException("FunctionNotFoundException!");

}

}

public static void main(String[] args) throws IllegalAccessException, InvocationTargetException, FunctionNotFoundException {

MyClass myClass = new MyClass();

task2(myClass);

Object[] param = new Object[] {568};

task3(myClass, "setNumber", param);

}

}

class FunctionNotFoundException extends Exception{

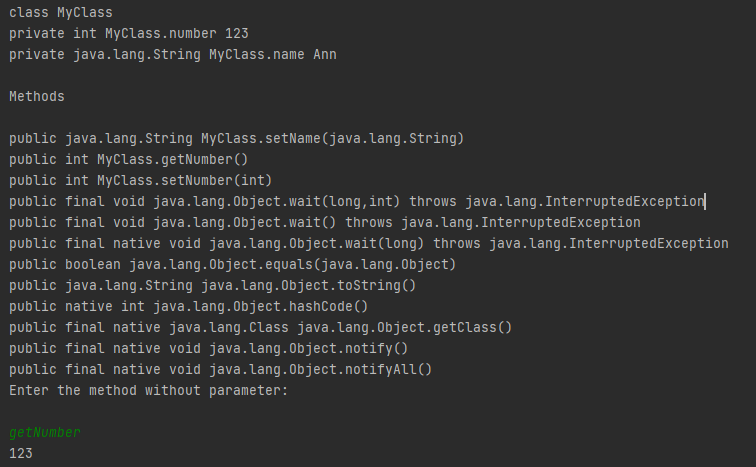
public FunctionNotFoundException(String message){

super(message);

}

}

Пример работы программы:





**Задание 4:**

Код:

import java.lang.reflect.Array;

import java.lang.reflect.Field;

import java.util.\*;

import java.util.stream.IntStream;

public class Task4 {

public static void main(String[] args) throws IllegalAccessException {

System.out.println("Mas:");

List<Integer> list = Arrays.asList(1,2,3);

Integer[] integers = task4(list, Integer.class);

System.out.println(myToString(integers));

List<String> list1 = Arrays.asList("one", "two", "three");

String[] strings = task4(list1, String.class);

System.out.println(myToString(strings));

List<MyClass> list2 = Arrays.asList(new MyClass(456, "Jane"), new MyClass());

MyClass[] myClasses = task4(list2, MyClass.class);

System.out.println(myToString(myClasses));

System.out.println("\nMatrix:");

List<String[]> matr1 = Arrays.asList(new String[]{"one", "two"}, new String[]{"three"});

String[][] matrStr = task4\_matr(matr1, String.class);

for(int r = 0; r <matrStr.length; r++) {

System.out.print(myToString(matrStr[r]));

System.out.println();

}

System.out.println();

List<MyClass[]> matr2 = Arrays.asList(new MyClass[]{new MyClass(456, "Jane"),

new MyClass(457, "Ostin")},

new MyClass[]{new MyClass(111, "Peter")});

MyClass[][] matrMyClass = task4\_matr(matr2, MyClass.class);

for(int r = 0; r <matrMyClass.length; r++) {

System.out.print(myToString(matrMyClass[r]));

System.out.println();

}

System.out.println("\nChangeSize");

Integer[] newIntegers = changeSize(integers, Integer.class, 10);

System.out.println(myToString(newIntegers));

MyClass[] newMyClasses = changeSize(myClasses, MyClass.class, 10);

System.out.println(myToString(newMyClasses));

System.out.println();

MyClass[][] newMyClasses1 = changeSize(matrMyClass, MyClass.class, 4, 3);

for(int r = 0; r <newMyClasses1.length; r++) {

System.out.print(myToString(newMyClasses1[r]));

System.out.println();

}

System.out.println();

}

public static <T> T[] task4(Collection<T> c, Class<T> type) {

if (c == null) {

return null;

}

Object o = Array.newInstance(type, c.size());

Iterator<T> iterator = c.iterator();

IntStream.range(0, c.size())

.forEach(i -> Array.set(o, i, iterator.next()));

T[] ts = (T[]) o;

return ts;

}

public static <T> T[][] task4\_matr(Collection<T[]> c, Class<T> type) {

if (c == null) {

return null;

}

Iterator<T[]> it1 = c.iterator();

int [] sz = new int[c.size()];

int k = 0;

while(it1.hasNext()){

sz[k] = it1.next().length;

k++;

}

T[][] o = (T[][]) Array.newInstance(type, c.size(), sz[0]);

ArrayIterator <T> arrayIterator = new ArrayIterator<T>(c);

Iterator<T> iterator = arrayIterator.iterator();

for(int i = 0; i<c.size(); i++){

for(int j = 0; j<sz[i]; j++){

Array.set(o[i], j, iterator.next());

if(!iterator.hasNext())

break;

}

}

return (T[][])o;

}

public static <T> T[] changeSize(T[] arr, Class<T> type, int size) {

Object o = Array.newInstance(type, size);

System.arraycopy(arr, 0, o, 0, arr.length);

T[] ts = (T[]) o;

return ts;

}

public static <T> T[][] changeSize(T[][] arr, Class<T> type, int rows, int cols) {

T[][] o = (T[][])Array.newInstance(type, rows, cols);

System.arraycopy(arr, 0, o, 0, arr.length);

return o;

}

public static <T> String myToString(T[] myClasses) throws IllegalAccessException {

String s = "";

if(isWrapperType(myClasses.getClass().getComponentType())){

s = Arrays.toString(myClasses);

}

else

for(int i =0; i< myClasses.length; i++) {

s+="[ ";

if(myClasses[i]!=null) {

Field[] fields = myClasses[i].getClass().getDeclaredFields();

Object value[] = new Object[fields.length];

int j = 0;

for (Field field : fields) {

field.setAccessible(true);

value[j] = field.get(myClasses[i]);

s += value[j] + " ";

j++;

}

s += "] ";

}

else s += "... ] ";

}

return s;

}

private static final Set<Class<?>> WRAPPER\_TYPES = getWrapperTypes();

public static boolean isWrapperType(Class<?> clazz)

{

return WRAPPER\_TYPES.contains(clazz);

}

private static Set<Class<?>> getWrapperTypes() {

Set<Class<?>> ret = new HashSet<Class<?>>();

ret.add(Boolean.class);

ret.add(Character.class);

ret.add(Byte.class);

ret.add(Short.class);

ret.add(Integer.class);

ret.add(Long.class);

ret.add(Float.class);

ret.add(Double.class);

ret.add(Void.class);

ret.add(String.class);

return ret;

}

}

class ArrayIterator <T> implements Iterable <T> {

private final List<T> elements;

public ArrayIterator(Collection<T[]> array) {

elements = new LinkedList<>();

for (T[] array1 : array) {

elements.addAll(Arrays.asList(array1));

}

}

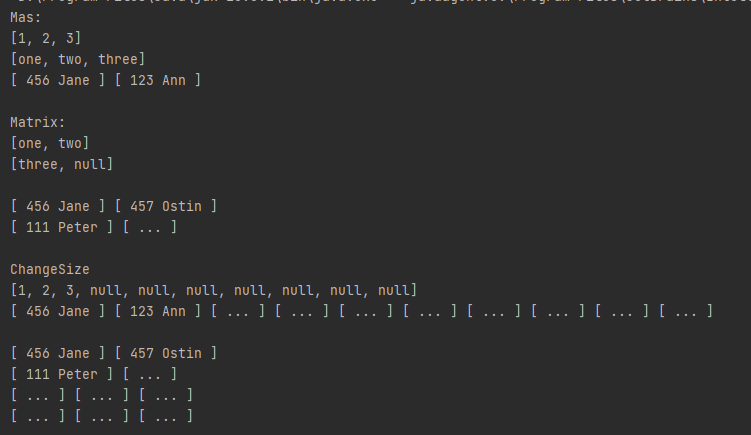
@Override

public Iterator<T> iterator() {

return elements.iterator();

}

Пример работы программы:



**Задание 5:**

Код:

import java.lang.reflect.Proxy;

public class Task5 {

public static void main(String[] args) {

NumFunction fun1 = new NumFunction("exp(-2.5\*x)\*sin(x)");

NumFunction fun = new NumFunction("x^2");

Evaluatable e = (Evaluatable) Proxy.newProxyInstance(fun.getClass().getClassLoader(),

fun.getClass().getInterfaces(), new FunHandler(fun));

e.evalf(2.0);

Evaluatable e1 = (Evaluatable) Proxy.newProxyInstance(fun1.getClass().getClassLoader(),

fun1.getClass().getInterfaces(), new FunHandler(fun1));

e1.evalf(0.9);

}

}

import edu.hws.jcm.data.Expression;

import edu.hws.jcm.data.Parser;

import edu.hws.jcm.data.Variable;

public class NumFunction implements Evaluatable{

String func;

public NumFunction (String func){

this.func = func;

}

public double evalf(double x){

Parser parser = new Parser(Parser.STANDARD\_FUNCTIONS);

Variable var = new Variable("x");

Variable par = new Variable("a");

parser.add(var);

parser.add(par);

String funStr = func;

Expression fun = parser.parse(funStr);

var.setVal(x);

return fun.getVal();

}

}

import java.lang.reflect.InvocationHandler;

import java.lang.reflect.Method;

public class FunHandler implements InvocationHandler {

private Object obj;

public FunHandler(Object obj){

this.obj = obj;

}

@Override

public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {

System.out.println("method: " + method.getName());

for(Object a: args)

System.out.println("arg: " + a);

double start = System.nanoTime();

double res = (double) method.invoke(obj, args);

double finish = System.nanoTime();

System.out.println("Elapsed time: " + (finish-start)+" ns");

System.out.println("res: "+res);

return res;

}

}

Пример работы программы:

