**Міністерство освіти і науки України**

**Харківський національний університет ім.В.Н. Каразіна**

**Факультет комп’ютерних наук**

**Кафедра безпеки інформаційних систем і технологій**

**Лабораторна робота №4**

***З навчальної дисципліни***

**«Спеціалізовані мови програмування та проектування електронних елементів і систем»**

Виконали:

Студенти групи КБ-41

**Шахов Б.С.**

**Дерев’янко Я.А.**

Перевірив:

Професор

**Олійников Р.В.**

Харків-2020

Лабораторна робота № 4

Хід роботи:

Завдання 1. Реализовать собственный класс с несколькими полями и методами, переопределить унаследованный метод toString().

Код програми:

public class Task1 {  
 private String prf;  
 public String pf;  
 private static String *prsf*;  
 public static String *psf*;  
  
 public static void main(String[] args) {  
 System.*out*.println(new Task1());  
 }  
  
 @Override  
 public String toString() {  
 return "Override method toString()";  
 }  
  
 public void setPf(String pf) {  
 this.pf = pf;  
 }  
  
 public String getPf() {  
 return pf;  
 }  
  
 public void setPrf(String prf) {  
 this.prf = prf;  
 }  
  
 public String getPrf() {  
 return prf;  
 }  
  
 public static void setPrsf(String prsf) {  
 Task1.*prsf* = prsf;  
 }  
  
 public static String getPrsf() {  
 return *prsf*;  
 }  
  
 public static void setPsf(String psf) {  
 Task1.*psf* = psf;  
 }  
  
 public static String getPsf() {  
 return *psf*;  
 }  
}

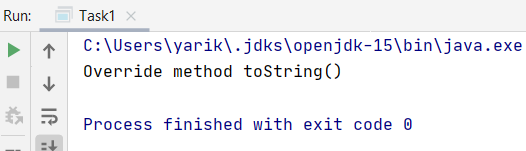


Рис. 1. – результат виконання першого завдання.

Завдання 2. Реализовать метод, принимающий пять разнотипных параметров (в т.ч. собственных классов), тремя разными способами: с явным заданием пяти разных параметров, через массив object[] и varargs. Напечатать на экране тип параметра и его значение (для класса - основные поля).

Код програми:

public class Task2 {  
  
  
 public static void main(String[] args) {  
 Task2 t2 = new Task2();  
 t2.method0("Test", 5, 10.0, true, 'A');  
 System.*out*.println();  
 t2.method(new Object[]{"Test2", 15, 20.0, false, 'B'});  
 System.*out*.println();  
 t2.method1("Test3", 25, 30.0, true, 'C');  
  
 }  
  
 public void method0(String str, Integer i, Double d, Boolean b, Character c) {  
 System.*out*.println(str.getClass().getName() + " " + str);  
 System.*out*.println(i.getClass().getName() + " " + i);  
 System.*out*.println(d.getClass().getName() + " " + d);  
 System.*out*.println(b.getClass().getName() + " " + b);  
 System.*out*.println(c.getClass().getName() + " " + c);  
 }  
  
 public void method(Object[] arr) {  
 for (Object o : arr) {  
 System.*out*.println(o.getClass().getName() + " " + o);  
 }  
 }  
  
 public void method1(Object... arr) {  
 for (Object o : arr) {  
 System.*out*.println(o.getClass().getName() + " " + o);  
 }  
 }  
  
}

Результат виконання програми:

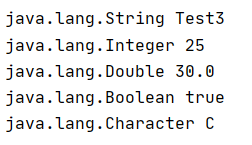
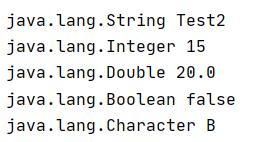
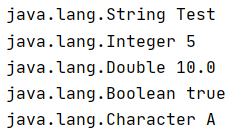


Рис. 2. – результат виконання другого завдання.

Завдання 3. Реализовать вложенный класс, статический вложенный класс, продемонстрировать доступность полей и методов внешнего класса.

Код програми:

public class Task3 {  
 private String prf = "private field";  
 public String pf = "public field";  
 private static String *prsf* = "private static field";  
 public static String *psf* = "public static field";  
  
 public static void main(String[] args) {  
 Task3 t3 = new Task3();  
 t3.print();  
 }  
  
 public void print() {  
 Task3.StaticClass staticClass = new Task3.StaticClass();  
 staticClass.print();  
 Task3.NotStaticClass notStaticClass = new Task3.NotStaticClass();  
 notStaticClass.print();  
  
 }  
  
 public static class StaticClass {  
 public void print() {  
 System.*out*.println(*prsf* + " is available from static class");  
 System.*out*.println(*psf* + " is available from static class");  
 }  
 }  
  
 public class NotStaticClass {  
 public void print() {  
 System.*out*.println(pf + " is available from not static class");  
 System.*out*.println(prf + " is available from not static class");  
 System.*out*.println(*prsf* + " is available from not static class");  
 System.*out*.println(*psf* + " is available from not static class");  
 }  
 }  
  
}

Результат виконання програми:

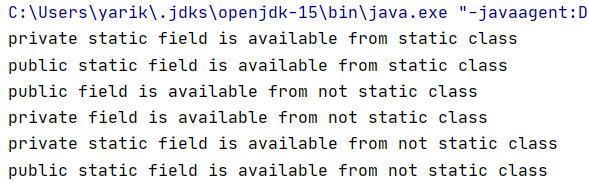


Рис. 3. – результат виконання третього завдання.

Завдання 4. Реализовать локальный класс, проверить доступность полей и методов внешнего класса, локальных переменных метода (изменяемых и нет), который создает локальный класс.

Код програми:

public class Task4 {  
 public String pf = "public field";  
 private String prs = "private field";  
  
  
 public static void main(String[] args) {  
 Task4 t4 = new Task4();  
 t4.task4();  
 *//System.out.println(var);  
 //System.out.println(finalVar);* }  
   
 public void task4() {  
 class LocalClass {  
 int var = 3;  
 final int finalVar = 4;  
  
 public void print() {  
 System.*out*.println(pf);  
 System.*out*.println(var);  
 System.*out*.println(finalVar);  
 *// System.out.println(prf);* }  
  
 */\* public static void main(String[] args) {  
 LocalClass lc = new LocalClass();  
 lc.print();  
 } \*/* }  
 }  
}

Результат виконання програми:

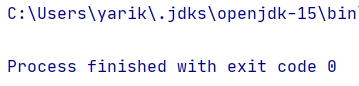


Рис. 4. – результат виконання четвертого завдання.

Завдання 5. Разработать собственный интерфейс шифрования, на его основе интерфейсы симметричного и асимметричного шифрования, на основе симметричного - блочного и поточного шифрования. Создать класс, реализующий интерфейс поточного шифрования (XOR с константой) со статическим методом, на вход принимающим байтовый массив и возвращающий его же на выходе. Задать начальное значение массива, зашифровать и расшифровать, напечатать каждый из результатов.

Код програми:

public class Task5 {  
  
 public static void main(String[] args) {  
 System.*out*.println("Start Encription");  
 byte[] shifr = StreamEncryptionImpl.*encrypt*("Test Encription".getBytes());  
 System.*out*.println(Arrays.*toString*(shifr));  
 System.*out*.println("Start Decription");  
 System.*out*.println(new String(StreamEncryptionImpl.*decrypt*(shifr)));  
 }  
}

public class StreamEncryptionImpl implements StreamEncryption {  
 private static final byte *KEY* = 0b1010101;  
  
 public static byte[] encrypt(byte[] bytes) {  
 for (int i = 0; i < bytes.length; i++) {  
 bytes[i] ^= *KEY*;  
 }  
 return bytes;  
 }  
  
 public static byte[] decrypt(byte[] bytes) {  
 return *encrypt*(bytes);  
 }  
}

Результат виконання програми:

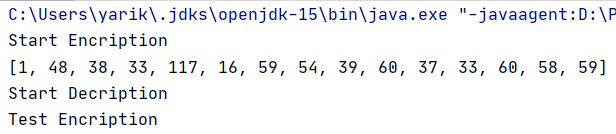


Рис. 5. – результат виконання п’ятого завдання.

Завдання 6. Создать три анонимных класса на основе интерфейса, продемонстрировать полиморфизм. Создать анонимный класс, вызвать его методы без определения локальной переменной.

Код програми:

public class Task6 {  
  
 public void print() {  
 }  
  
 public static void main(String[] args) {  
 Task6 t6;  
 t6 = new Task6() {  
 @Override  
 public void print() {  
 System.*out*.println("Variant I");  
 }  
 };  
 t6.print();  
 t6 = new Task6() {  
 @Override  
 public void print() {  
 System.*out*.println("Variant II");  
 }  
 };  
 t6.print();  
 t6 = new Task6() {  
 @Override  
 public void print() {  
 System.*out*.println("Variant III");  
 }  
 };  
 t6.print();  
 new Task6() {  
 @Override  
 public void print() {  
 System.*out*.println("Variant IV without variable declaration");  
 }  
 }.print();  
 }  
}

Результат виконання програми:

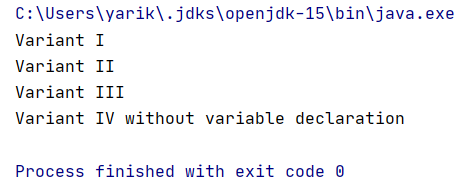


Рис. 6. – результат виконання шостого завдання.

Висновки: виконуючи дану лабораторну роботу, ми повторили принципи роботи з класами. Повторили реалізацію методів з різними типами параметрів на прийом. Реалізували вкладені та локальні класи. Розробили власний інтерфейс шифрування та реалізували різні класи, які наслідували даний інтерфейс. Ознайомилися з анонімними класами та їх застосуванням.