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»

Виконав:

Студент групи КБ-41

Кравченко Є.М.

Перевірив:

Олійников Р.В.

Харків – 2020 р.

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

@Override  
**public** String toString() {  
 **return "Override App toString()"**;  
}

System.***out***.println(**new** App());

1. 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).

App app = **new** App();  
app.method(**new** Object[]{**"Hello"**, 1, 2.0, **false**, **'t'**});  
app.method1(**"Dadada"**, 4, 7., **true**, **'l'**);

**public void** method(Object[] arr) {  
 **for** (Object o : arr) {  
 System.***out***.println(o.getClass().getName());  
 }  
}  
  
**public void** method1(Object... arr) {  
 **for** (Object o : arr) {  
 System.***out***.println(o.getClass().getName());  
 }  
}

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

app.print();

**public void** print() {  
 App.StaticClass staticClass = **new** App.StaticClass();  
 staticClass.print();  
 NotStaticClass notStaticClass = **new** NotStaticClass();  
 notStaticClass.print();  
  
}

**public static class** StaticClass {  
 **public void** print() {  
 System.***out***.println(*privateStaticAppField* + **" is available from static class"**);  
 System.***out***.println(*publicStaticAppField* + **" is available from static class"**);  
 }  
}  
  
**public class** NotStaticClass {  
 **public void** print() {  
 System.***out***.println(**publicAppField** + **" is available from not static class"**);  
 System.***out***.println(**privateAppField** + **" is available from not static class"**);  
 System.***out***.println(*privateStaticAppField* + **" is available from not static class"**);  
 System.***out***.println(*publicStaticAppField* + **" is available from not static class"**);  
 }  
}

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

**public void** task4() {  
 **int** var = 3;  
 **final int** finalVar = 4;  
 **class** LocalClass {  
 **public void** print() {  
 System.***out***.println(var);  
 System.***out***.println(finalVar);  
 App.**this**.print();  
 }  
 }  
}

1. 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.

System.***out***.println(**"Hello"**);  
**byte**[] shifr = StreamEncryptionImpl.*encrypt*(**"Hello"**.getBytes());  
System.***out***.println(Arrays.*toString*(shifr));  
System.***out***.println(**new** String(StreamEncryptionImpl.*decrypt*(shifr)));

**public interface** Encryption {  
}

**public interface** StreamEncryption **extends** SymmetricEncryption {  
}

**public interface** BlockEncryption **extends** SymmetricEncryption {  
}

**public interface** AsymmetricEncryption **extends** Encryption {  
}

**public interface** SymmetricEncryption **extends** Encryption {  
}

**public class** StreamEncryptionImpl **implements** StreamEncryption {  
 **private static final byte *KEY*** = 0b1111111;  
  
 **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);  
 }  
}

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

Runnable r1;  
r1 = **new** Runnable() {  
 @Override  
 **public void** run() {  
 System.***out***.println(**"First realization"**);  
 }  
};  
r1.run();  
r1 = **new** Runnable() {  
 @Override  
 **public void** run() {  
 System.***out***.println(**"Second realization"**);  
 }  
};  
r1.run();  
r1 = **new** Runnable() {  
 @Override  
 **public void** run() {  
 System.***out***.println(**"Third realization"**);  
 }  
};  
r1.run();  
**new** Runnable() {  
 @Override  
 **public void** run() {  
 System.***out***.println(**"Easy"**);  
 }  
}.run();