ЛАБОРАТОРНАЯ РАБОТА № 7

«Введение в наследование»

Выполнил Козлов И.А.   
Группа РИВ-120908у

**Цель работы**  
Получить представление о механизме наследования в языке

программирования Java.

**Git репозиторий**

https://github.com/Catlerc/javadev

**Задание 1**

Напишите программу, в которой есть суперкласс с приватным текстовым полем, конструктором с текстовым параметром и где переопределен метод toString (). На основе суперкласса путем наследования создается подкласс. У него появляется еще одно приватное текстовое ноле. Также подкласс должен иметь версии конструктора с одним и двумя текстовыми аргументами, а еще в нем должен быть переопределен метод toString (). В обоих классах метод toString () переопределяется так, что он возвращает строку с названием класса и значение текстового поля или текстовых полей.**Результат**

class StringHolder {  
 private String str;  
  
 public StringHolder(String str) {  
 this.str = str;  
 }  
  
  
 @Override  
 public String toString() {  
 return "StringHolder{" +  
 "str='" + str + '\'' +  
 '}';  
 }  
  
 public void setStr(String str) {  
 this.str = str;  
 }  
  
 public String getStr() {  
 return str;  
 }  
}  
  
class SubStringHolder extends StringHolder {  
 private String str2;  
  
 public SubStringHolder(String str) {  
 super(str);  
 }  
  
 public SubStringHolder(String str, String str2) {  
 super(str);  
 this.str2 = str2;  
 }  
  
 @Override  
 public String toString() {  
 return "SubStringHolder{" +  
 "str='" + getStr() + "', " +  
 "str2='" + str2 + "'" +  
 '}';  
 }  
  
 public String getStr2() {  
 return str2;  
 }  
  
 public void setStr2(String str2) {  
 this.str2 = str2;  
 }  
}  
  
public class Task1 {  
 public static void main(String[] args) {  
 var holder = new StringHolder("holder");  
 System.*out*.println(holder);  
 var subHolder = new SubStringHolder("holder");  
 System.*out*.println(subHolder);  
 var subHolder2 = new SubStringHolder("holder", "subholder");  
 System.*out*.println(subHolder2);  
 }  
}

**Задание 2**

Напишите программу, в которой есть суперкласс с приватным текстовым полем. В базовом классе должен быть метод для присваивания значения полю: без параметров и с одним текстовым параметром. Объект суперкласса создается передачей одного текстового аргумента конструктору. Доступное только для чтения свойство результатом возвращает длину текстовой строки. На основе суперкласса создается подкласс. В подклассе появляется дополнительное открытое целочисленное поле. В классе должны быть такие версии метода для присваивания значений полям (используется переопределение и перегрузка метода из суперкласса): без параметров, с текстовым параметром, с целочисленным параметром, с текстовым и целочисленным параметром. У конструктора подкласса два параметра (целочисленный и текстовый).**Результат**

class StringHolder2 {  
 private String str;  
 public void setFields() {  
 this.str = "nothing!";  
 }  
  
 public int getLength() {  
 return str.length();  
 }  
  
 public void setFields(String str) {  
 this.str = str;  
 }  
  
 public StringHolder2(String str) {  
 this.str = str;  
 }  
}  
  
class SubStringHolder2 extends StringHolder2 {  
 public int integer;  
  
 public SubStringHolder2(String str, int integer) {  
 super(str);  
 this.integer = integer;  
 }  
  
 public void setFields() {  
 super.setFields();  
 this.integer=0;  
 }  
 public void setFields(String newStr) {  
 super.setFields(newStr);  
 this.integer=0;  
 }  
 public void setFields(int newInteger) {  
 this.integer=newInteger;  
 super.setFields();  
 }  
 public void setFields(String newStr, int newInteger) {  
 super.setFields(newStr);  
 this.integer=newInteger;  
 }  
}  
  
public class Task2 {  
 public static void main(String[] args) {  
 var subHolder = new SubStringHolder2("test", 42);  
 System.*out*.println("lenght: "+subHolder.getLength());  
 System.*out*.println("set str='str'");  
 subHolder.setFields("str");  
 System.*out*.println("lenght: "+subHolder.getLength());  
 subHolder.setFields("integer: "+subHolder.integer);  
 System.*out*.println("set integer=-1337");  
 subHolder.setFields(-1337);  
 System.*out*.println("integer: "+subHolder.integer);  
 System.*out*.println("set integer=123");  
 System.*out*.println("set str='sdf'");  
 subHolder.setFields("sdf", 123);  
 System.*out*.println("lenght: "+subHolder.getLength());  
 subHolder.setFields("integer: "+subHolder.integer);  
  
 }  
}

**Задание 3**

Напишите программу, в которой на основе суперкласса создается подкласс, а на основе этого подкласса создается еще один подкласс (цепочка наследования из трех классов). В первом суперклассе есть открытое целочисленное поле, метод с одним параметром для присваивания значения полю и конструктор с одним параметром. Во втором классе появляется открытое символьное поле, метод с двумя параметрами для присваивания значения полям (перегрузка метода из суперкласса) и конструктор с двумя параметрами. В третьем классе появляется открытое текстовое ноле, метод с тремя аргументами для присваивания значений полям (перегрузка метода из суперкласса) и конструктор с тремя параметрами. Для каждого класса определите метод toString () так, чтобы он возвращал строку с названием класса и значениями всех полей объекта

**Результат**

class A {  
 public int integer;  
  
 public void setFields(int integer) {  
 this.integer = integer;  
 }  
  
 public A(int integer) {  
 this.integer = integer;  
 }  
  
 @Override  
 public String toString() {  
 return "A{" +  
 "integer=" + integer +  
 '}';  
 }  
}  
  
class B extends A {  
 public char chr;  
  
 public void setFields(int newInteger, char newChr) {  
 super.setFields(newInteger);  
 this.chr = newChr;  
 }  
  
 public B(int integer, char chr) {  
 super(integer);  
 this.chr = chr;  
 }  
  
 public String toString() {  
 return "B{" +  
 "integer=" + integer + ", " +  
 "chr=" + chr +  
 '}';  
 }  
}  
  
class C extends B {  
 public C(int integer, char chr, String str) {  
 super(integer, chr);  
 this.str = str;  
 }  
  
 public String str;  
  
 public void setFields(int newInteger, char newChr, String newString) {  
 super.setFields(newInteger, newChr);  
 this.str = newString;  
 }  
  
 public String toString() {  
 return "C{" +  
 "integer=" + integer + ", " +  
 "chr=" + chr + ", " +  
 "str=" + str +  
 '}';  
 }  
  
}  
  
public class Task3 {  
 public static void main(String[] args) {  
 var a = new A(1);  
 System.*out*.println(a);  
 var b = new B(1, 'a');  
 System.*out*.println(b);  
 var c = new C(1, 'a', "str");  
 System.*out*.println(c);  
 c.setFields(23, 'r', "notstring");  
 System.*out*.println(c);  
 }  
}

**Задание 4**

Напишите программу, в которой использована цепочка наследования из трех классов. В первом классе есть открытое символьное поле. Во втором классе появляется открытое текстовое поле. В третьем классе появляется открытое целочисленное поле. В каждом из классов должен быть конструктор, позволяющий создавать объект на основе значений полей, переданных аргументами конструктору, а также конструктор создания

копии.**Результат**

class A2 {  
 public char chr;  
  
 public A2(char chr) {  
 this.chr = chr;  
 }  
  
 public A2(A2 a) {  
 this.chr = a.chr;  
 }  
  
 @Override  
 public String toString() {  
 return "A2{" +  
 "chr=" + chr +  
 '}';  
 }  
}  
  
class B2 extends A2 {  
 public String str;  
  
 public B2(char chr, String str) {  
 super(chr);  
 this.str = str;  
 }  
  
 public B2(B2 b) {  
 super(b.chr);  
 this.str = b.str;  
 }  
  
 @Override  
 public String toString() {  
 return "B2{" +  
 "chr=" + chr + ", " +  
 "str=" + str +  
 '}';  
 }  
}  
  
class C2 extends B2 {  
 public int integer;  
  
 public C2(char chr, String str, int integer) {  
 super(chr, str);  
 this.integer = integer;  
 }  
  
 public C2(C2 c) {  
 super(c.chr, c.str);  
 this.integer = c.integer;  
 }  
  
 @Override  
 public String toString() {  
 return "C2{" +  
 "chr=" + chr + ", " +  
 "str=" + str + ", " +  
 "integer=" + integer +  
 '}';  
 }  
}  
  
public class Task4 {  
 public static void main(String[] args) {  
 var a2 = new A2('a');  
 var b2 = new B2('a', "str");  
 var c2 = new C2('a', "str", 42);  
 var copyC = new C2(c2);  
 System.*out*.println(a2);  
 System.*out*.println(b2);  
 System.*out*.println(c2);  
 System.*out*.println(copyC);  
 }  
}

**Задание 5**

Напишите программу, в которой есть суперкласс с защищенным текстовым полем, конструктор с текстовым параметром и метод, при вызове которого в консольном окне отображается название класса и значение поля. На основе суперкласса создаются два подкласса (оба на основе одного и того же суперкласса). В одном из классов появляется защищенное целочисленное поле, там есть конструктор с двумя параметрами и переопределен метод для отображения значений полей объекта и названия класса. Во втором подклассе появляется защищенное символьное поле, конструктор с двумя параметрами и переопределен метод, отображающий в консоли название класса и значения полей. В главном методе создайте объекты каждого из классов. Проверьте работу метода, отображающего значения полей объектов, для каждого из объектов. Вызовите этот же метод через объектную переменную суперкласса, которая ссылается на объект производного класса.

**Результат**

class StringBox {  
 protected String str;  
  
 public StringBox(String str) {  
 this.str = str;  
 }  
  
 public void print() {  
 System.*out*.println("StringBox{str=" + str + '}');  
 }  
}  
  
class StringBoxA extends StringBox {  
 protected int integer;  
  
 public StringBoxA(String str, int integer) {  
 super(str);  
 this.integer = integer;  
 }  
  
 public void print() {  
 System.*out*.println("StringBoxA{str=" + str + ", integer=" + integer + '}');  
 }  
}  
  
class StringBoxB extends StringBox {  
 protected char chr;  
  
 public StringBoxB(String str, char chr) {  
 super(str);  
 this.chr = chr;  
 }  
  
 public void print() {  
 System.*out*.println("StringBoxB{str=" + str + ", chr=" + chr + '}');  
 }  
}  
  
public class Task5 {  
 public static void main(String[] args) {  
 var box = new StringBox("str");  
 box.print();  
 var boxA = new StringBoxA("str", 42);  
 boxA.print();  
 var boxB = new StringBoxB("str", 'a');  
 boxB.print();  
  
  
 StringBox someBox = new StringBoxB("maybe str", '7');  
 someBox.print();  
 }  
}

**Вывод**

Во время выполнения работы были отработаны навыки работы с классами и наследованием классов.