**Министерство науки и высшего образования Российской Федерации**

ФГАОУ ВО «УрФУ имени первого Президента России Б.Н.Ельцина»

ИРИТ-РТФ

Центр ускоренного обучения

**Отчет**

**о лабораторной работе №7**

**“Введение в наследование”**

Руководитель преподаватель Н.А. Архипов

Студент гр. РИЗ-100028у В.А. Кудрявцев

**Екатеринбург 2021**

ЦЕЛЬ РАБОТЫ

Цель работы получить представление о механизме наследования в языке программирования Java.

ОПИСАНИЕ ЗАДАЧИ

Составить 5 программ по представленным задачам, представить листинги программ, во вложении к отчёту приложить файлы готовых классов.

ХОД ВЫПОЛНЕНИЯ

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.***

**public class** example\_18\_01 {  
 **public static void** main(String[] args) {  
 example\_18\_01\_1 test = **new** example\_18\_01\_2(**"1String"**, **"2String"**);  
  
 System.***out***.println(test.toString());  
 }  
  
}

**public class** example\_18\_01\_1 {  
 **private** String **s**;  
  
 **public** example\_18\_01\_1(String s) {  
 **this**.**s** = s;  
 }  
  
 @Override  
 **public** String toString() {  
 **return** (getClass().getSimpleName() + **" string field = "** + **this**.**s**);  
 }  
  
}

**public class** example\_18\_01\_2 **extends** example\_18\_01\_1 {  
  
 *// public example\_18\_01\_2(String s) {  
 // super(s);  
 // }* **private** String **s**;  
  
 **public** example\_18\_01\_2(String s1, String s2) {  
 **super**(s1);  
 **this**.**s** = s2;  
 }  
  
 @Override  
 **public** String toString() {  
 **return super**.toString()  
 + **"\n"** + getClass().getSimpleName() + **" string field = "** + **this**.**s**;  
 }  
}

***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).***

**public class** example\_18\_02 {  
 **public static void** main(String[] args) {  
 example\_18\_02\_2 test = **new** example\_18\_02\_2(**"1String"**, 1);  
 System.***out***.println(test.getSLenght() + test.**i**);  
 test.setS(**"21String"**, 4);  
 System.***out***.println(test.getSLenght() + test.**i**);  
 }  
}

**public class** example\_18\_02\_1 {  
 **private** String **s**;  
  
 **public** example\_18\_02\_1(String s) {  
 setS(s);  
 }  
  
 **public void** setS(String s) {  
 **this**.**s** = s;  
 }  
 **public void** setS() {  
 **this**.**s** = **"nothing8"**;  
 }  
  
 **public int** getSLenght() {  
 **return s**.length();  
 }  
  
}

**public class** example\_18\_02\_2 **extends** example\_18\_02\_1{  
 **public** example\_18\_02\_2(String s) {  
 **super**(s);  
 }  
  
 **public int i**;  
  
 **public** example\_18\_02\_2(String s, **int** i) {  
 **super**(s);  
 **this**.**i** = i;  
 }  
  
 @Override  
 **public void** setS(String s) {  
 **super**.setS(s);  
 }  
  
 @Override  
 **public void** setS() {  
 **super**.setS();  
 }  
  
 **public void** setS(**int** i) {  
 **super**.setS();  
 **this**.**i** = i;  
 }  
 **public void** setS(String s, **int** i) {  
 **super**.setS(s);  
 **this**.**i** = i;  
 }  
  
}

***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.***

**public class** example\_18\_03 {  
 **public static void** main(String[] args) {  
 example\_18\_03\_3 test = **new** example\_18\_03\_3(123, **'Q'**, **"test"**);  
 System.***out***.println(test.toString());  
 }  
  
}

**public class** example\_18\_03\_1 {  
 **public int intClass1**;  
  
 **public void** setIntClass1(**int** intClass1) {  
 **this**.**intClass1** = intClass1;  
 }  
  
 **public** example\_18\_03\_1(**int** intClass1) {  
 **this**.**intClass1** = intClass1;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Class1\_example17\_03{"** +  
 **"intClass1="** + **intClass1** +  
 **'}'**;  
 }  
}

**public class** example\_18\_03\_2 **extends** example\_18\_03\_1{  
 **public char charClass2**;  
  
 **public** example\_18\_03\_2(**int** intClass1,**char** charClass2) {  
 **super**(intClass1);  
 **this**.**charClass2** = charClass2;  
 }  
  
 **public void** setCharClass2(**char** charClass2, **int** intClass1) {  
 **this**.**charClass2** = charClass2;  
 setIntClass1(intClass1);  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Class2\_example17\_03{charClass2="** +  
 **charClass2** +  
 **"}\n"** +  
 **super**.toString();  
 }  
}

**public class** example\_18\_03\_3 **extends** example\_18\_03\_2{  
 **public** String **stringClass3**;  
  
 **public** example\_18\_03\_3(**int** intClass1, **char** charClass2, String stringClass3) {  
 **super**(intClass1, charClass2);  
 setStringClass3(stringClass3);  
 }  
  
 **public void** setStringClass3(String stringClass3) {  
 **this**.**stringClass3** = stringClass3;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Class3\_example17\_03{"** +  
 **"stringClass3="** + **stringClass3** +  
 **'}'** + **"\n"** + **super**.toString();  
 }  
}

***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.***

**public class** example\_18\_04 {  
 **public static void** main(String[] args) **throws** CloneNotSupportedException {  
 example\_18\_04\_3 test = **new** example\_18\_04\_3(**'b'**,**"check"**, 123 );  
 example\_18\_04\_3 testClone = **new** example\_18\_04\_3(test);  
 System.***out***.println(testClone.**intClass3** + **" "** +  
 testClone.**charClass1** + **" "** +  
 testClone.**stringClass2**);  
 }  
  
}

**public class** example\_18\_04\_1 {  
 **public char charClass1**;  
  
 **public** example\_18\_04\_1(**char** charClass1) {  
 **this**.**charClass1** = charClass1;  
 }  
  
 **public** example\_18\_04\_1(example\_18\_04\_1 p) {  
 **this**.**charClass1** = p.**charClass1**;  
 }  
}

**public class** example\_18\_04\_2 **extends** example\_18\_04\_1{  
 **public** String **stringClass2**;  
  
 **public** example\_18\_04\_2(**char** charClass1) {  
 **super**(charClass1);  
 }  
 **public** example\_18\_04\_2(example\_18\_04\_2 p) {  
 **super**(p.**charClass1**);  
 **this**.**stringClass2** = p.**stringClass2**;  
 }  
  
 **public** example\_18\_04\_2(**char** charClass1, String stringClass2) {  
 **super**(charClass1);  
 **this**.**stringClass2** = stringClass2;  
 }  
}

**public class** example\_18\_04\_3 **extends** example\_18\_04\_2{  
 **public int intClass3**;  
  
  
 **public** example\_18\_04\_3(**char** charClass1, String stringClass2) {  
 **super**(charClass1, stringClass2);  
 }  
  
 **public** example\_18\_04\_3(**char** charClass1, String stringClass2, **int** intClass3) {  
 **super**(charClass1, stringClass2);  
 **this**.**intClass3** = intClass3;  
 }  
  
 **public** example\_18\_04\_3(example\_18\_04\_3 p){  
 **super**(p.**charClass1**, p.**stringClass2**);  
 **this**.**intClass3** = p.**intClass3**;  
 }  
}

***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.***

**public class** example\_18\_05 {  
 **public static void** main(String[] args) {  
 example\_18\_05\_1 testClass1 = **new** example\_18\_05\_1(**"testClass1"**);  
 testClass1.showVar();  
  
 example\_18\_05\_2 testClass2 = **new** example\_18\_05\_2(**"testClass2"**, 2);  
 testClass2.showVar();  
  
 example\_18\_05\_3 testClass3 = **new** example\_18\_05\_3(**"testClass3"**, **'b'**);  
 testClass3.showVar();  
  
 System.***out***.println(**"вызов метода через объектную переменную суперкласса на объект подкласса \"testClass2\""**);  
 example\_18\_05\_1 objA;  
 objA = testClass2;  
 objA.showVar();  
 }  
}

**public class** example\_18\_05\_1 {  
 **private** String **stringClass1**;  
  
 **public** example\_18\_05\_1(String stringClass1) {  
 **this**.**stringClass1** = stringClass1;  
 }  
  
 **public void** showVar() {  
 System.***out***.println(getClass().getSimpleName() +  
 **" stringClass1 = "** +  
 **stringClass1**);  
 }  
}

**public class** example\_18\_05\_2 **extends** example\_18\_05\_1{  
 **private int intClass2**;  
  
 **public** example\_18\_05\_2(String stringClass1, **int** intClass2) {  
 **super**(stringClass1);  
 **this**.**intClass2** = intClass2;  
 }  
  
 @Override  
 **public void** showVar() {  
 **super**.showVar();  
 System.***out***.println(getClass().getSimpleName() +  
 **" intClass2 = "** +  
 **intClass2**);  
 }  
}

**public class** example\_18\_05\_3 **extends** example\_18\_05\_1{  
 **private char charClass3**;  
  
 **public** example\_18\_05\_3(String stringClass1, **char** charClass3) {  
 **super**(stringClass1);  
 **this**.**charClass3** = charClass3;  
 }  
  
 @Override  
 **public void** showVar() {  
 **super**.showVar();  
 System.***out***.println(getClass().getSimpleName() +  
 **" charClass3 = "** +  
 **charClass3**);  
 }  
}

ВЫВОДЫ

В языке Java удобно связывать классы в единый проект для согласованной работы приложения.