Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Уральский государственный университет имени первого президента России
Б. Н. Ельцина»

Институт радиоэлектроники и информационных технологий
Центр ускоренного обучения

Отчёт по лабораторной работе №7

«Введение в наследование»

Руководитель ст. преподаватель — Н. А. Архипов

Студент гр. РИЗ-100028у — И. С. Арсентьев

Екатеринбург 2021

Лабораторная работа №6.

«Введение в наследование»

Цель: 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 задач:

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_02_71
{
   public static void main(String[] args)
   {
      Test02_01 ts1 = new Test02_01("Programming");
      System.out.println(ts1.toString());
      Test02_012 ts2 = new Test02_012("Programming", "science");
      System.out.println(ts2.toString());
   }
}
class Test02_01//текстовое поле
{
   private String textArea;

   Test02_01(String text)
   {
      this.textArea = text;
   }

   public String getName ()
   {
      return this.textArea;
   }

   public String toString ()
```

```
   {
      return "Название класса: " + getClass().getSimpleName() + ", название переменной: " +
this.textArea;
   }
}
class Test02_012 extends Test02_01
{
   private String textArea;
   private String comment;
   Test02_012(String text, String textComment)
   {
      super(text);
      this.comment = textComment;
   }
   public String toString ()
   {
      return "Название класса: " + getClass().getSimpleName() + ", название переменной из
Первого класса: " + this.getName() + "\n" +
            "Переменная класса: " + this.comment;
   }
}
```

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_02_72
{
   public static void main(String[] args)
   {
   }}

class Test
{
   private String textArea;

   Test(String text)
   {
      this.textArea = text;
   }
```

```
    public String word ()
    {
        return this.textArea;
    }

    public int lengthWord ()
    {
        return this.textArea.length();
    }
}

class Test1 extends Test
{
    private int lenght = 0;
    Test1(String text)
    {
        super(text);
        lenght = super.lengthWord();
    }

    void display ()
    {
        System.out.println("Размер: " + lenght + " слово: " + word());
    }
}
```

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_02_73
{
    public static void main(String[] args)
    {
        Test07_03 test07_03 = new Test07_03(20);
        System.out.println(test07_03.toString());

        Test07_031 test10_031 = new Test07_031('A', 23);
        System.out.println(test10_031.toString());
```

```
        Test07_032 test10_032 = new Test07_032("Programming",'A', 231);
        System.out.println(test10_032.toString());
    }
}
class Test07_03
{
    public int num = 0;
    public Test07_03(int val)
    {
        this.num = val;
    }
    public void setVal (int val)
    {
        this.num = val;
    }
    public String toString()
    {
        return  "Имя класса: " + getClass().getSimpleName() +
                "Значение переменной: " + this.num;
    }
}
class Test07_031 extends Test07_03
{
    public char symbol;

    public Test07_031(char ch, int val)
    {
        super(val);
        this.symbol = ch;
    }

    public void setFirst(char ch, int val)
    {
        super.setVal(val);
    }

    public String toString()
    {
        return  "Имя класса: " + getClass().getSimpleName() +
                "Значение переменных: " + this.symbol + " : " + super.num;
    }
}
class Test07_032 extends Test07_031
{
    public String symb;
    public Test07_032(String str, char ch, int val)
    {
        super(ch, val);
        this.symb = str;
```

```
   }
   public void setValueString (String str, char ch, int val)
   {
      super.setFirst(ch, val);
   }
   public String toString()
   {
      return  "Имя класса: " + getClass().getSimpleName() +
            "Значение переменных: " + this.symb + " : " + super.symbol + " : " + super.num;
   }
}
```

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_02_74
{
   public static void main(String[] args)
   {

      Test02_04 test02_04 = new Test02_04('A');

      Test02_041 test02_041 = new Test02_041("Programming", 's');

      Test02_042 test02_042 = new Test02_042(20, "Hello,World", 'o');

   }
}
class Test02_04
{
   public char symbol;

   Test02_04(char ch)
   {
      this.symbol = ch;
   }

  public Test02_04(Test02_04 t04)
   {
      this(t04.symbol);
   }
}

class Test02_041 extends Test02_04
{
```

```
    public String textArea;

    Test02_041(String text, char ch)
    {
        super(ch);
        this.textArea = text;
    }
    public Test02_041(Test02_041 t041)
    {
        this(t041.textArea, t041.symbol);
    }
}
class Test02_042 extends Test02_041
{
    public int perem;
    Test02_042(int val, String text, char ch)
    {
        super(text, ch);
        this.perem = val;
    }

    public Test02_042(Test02_042 t042)
    {
        this(t042.perem, t042.textArea, t042.symbol);
    }

    public void display()
    {
        System.out.println(this.perem + " : " + this.symbol + " : " + this.textArea);
    }
}
```

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_02_75{
  public static void main(String[] args) {
        example_02_75_SuperClass superClassObject = new example_02_75_SuperClass("передал в конструктор суперкласса");
        String className = superClassObject.toString();
        System.out.println(className);

     example_02_75_SubOneClass subClassObject = new example_02_75_SubOneClass("передал в конструктор подкласса");
        String ClassName = subClassObject.toString();
        System.out.println(ClassName);
     }
  }
//подкласс
public class example_02_75_SubOneClass extends example_02_75_SuperClass{
     private int Int1;
     example_02_75_SubOneClass(String Str1) {
        super(Str1);
     }
     example_02_75_SubOneClass (int Int1){
        super();
        this.Int1 = Int1;
     }
     public String toString (){
        String superClassNameAndFieldValue;
        superClassNameAndFieldValue = "sub" + "\n" +
              " Class name: " + this.getClass().getSimpleName() + "\n" ;
        return superClassNameAndFieldValue ;
     }
  }
//суперкласс
public class example_02_75_SuperClass {
  private String Str1;
  example_02_75_SuperClass(String Str1){
     System.out.println(Str1);
  }

  public example_02_75_SuperClass() {
  }
  public String toString (){
     String superClassNameAndFieldValue;
     superClassNameAndFieldValue = "super" + "\n" +
           " Class name: " + this.getClass().getSimpleName() + "\n" ;
     return superClassNameAndFieldValue ;
  }
}
```

Вывод:

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

Выполненные задания в ходе лабораторной работы демонстрируют разные способы сочетания классов, как в теле одного, так и отдельно написанные.

Связи между классами обусловлены условиями наследования и совместного использования ресурсов.

Каждое решение задания сопровождено листингами программ, в архиве с классами будут добавлены соответствующие файлы.