Объявленные в методе переменные являются локальными переменными метода, а не членами классов, и не инициализируются значениями по умолчанию при создании объекта класса или вызове метода.

Статические методы и поля

Поля данных, объявленные в классе как **static**, являются общими для всех объектов класса и называются переменными класса. Если один объект изменит значение такого поля, то это изменение увидят все объекты. Для работы со статическими атрибутами используются статические методы, объявленные со спецификатором **static**. Такие методы являются методами класса, не привязаны ни к какому объекту и не содержат указателя **this** на конкретный объект, вызвавший метод. Статические методы реализуют парадигму «раннего связывания», жестко определяющую версию метода на этапе компиляции. По причине недоступности указателя **this** статические поля и методы не могут обращаться к нестатическим полям и методам напрямую, так как для обращения к статическим полям и методам достаточно имени класса, в котором они определены.

```
// пример # 4 : статические метод и поле: Mark.java
package chapt03;

public class Mark {
      private int mark = 3;
      public static int coeff = 5;

      public double getResult() {
            return (double)coeff*mark/100;
      }
      public static void setCoeffFloat(float c) {
            coeff = (int)coeff*c;;
      }
      public void setMark(int mark) {
            this.mark = mark;
      }
//из статического метода нельзя обратиться к нестатическим полям и методам
      /*public static int getResult() {
            setMark(5);//ошибка
            return coeff*mark/100;//ошибка
      }*/
}
```

При создании двух объектов

```
Mark ob1 = new Mark();
Mark ob2 = new Mark();
```

Значение **ob1.coeff** и **ob2.coeff** и равно **5**, поскольку располагается в одной и той же области памяти. Изменить значение статического члена можно прямо через имя класса:

```
Mark.coeff = 7;
```

Вызов статического метода также следует осуществлять с помощью указания: **ClassName.*methodName*()**, а именно:

```
Mark.setCoeffFloat();
```

```
float z = Math.max(x, y); // определение максимума из двух значений
System.exit(1);  // экстренное завершение работы приложения
```

Статический метод можно вызывать также с использованием имени объекта, но такой вызов снижает качество кода и не будет логически корректным, хотя и не приведет к ошибке компиляции.

Переопределение статических методов класса не имеет практического смысла, так как обращение к статическому атрибуту или методу осуществляется по большей части посредством задания имени класса, которому они принадлежат.

Модификатор final

Модификатор **final** используется для определения констант в качестве члена класса, локальной переменной или параметра метода. Методы, объявленные как **final**, нельзя замещать в подклассах, для классов – создавать подклассы. Например:

```
/* пример # 5 : final-поля и методы: Rector.java: ProRector.java */
package chapt03;

public class Rector {

        // инициализированная константа
        final int ID = (int)(Math.random()*10);
        // неинициализированная константа
        final String NAME_RECTOR;

        public Rector() {
        // инициализация в конструкторе
                NAME_RECTOR = "Старый";// только один раз!!!
        }
        // {NAME_RECTOR = "Новый";} // только один раз!!!

        public final void jobRector() {
                // реализация
                // ID = 100; //ошибка!
        }
        public boolean checkRights(final int num) {
                // id = 1; //ошибка!
                final int CODE = 72173394;
                if (CODE == num) return true;
                else return false;
        }
        public static void main(String[] args) {
                System.out.println(new Rector().ID);
        }
}
package chapt03;

public class ProRector extends Rector {
        // public void jobRector(){} //запрещено!
}
```