

Java Basics Procedural Decomposition Lesson 05

OLGA SMOLYAKOVA

План конспекта

1. Декомпозиция кода с помощью статических методов

2. Основы работы с типом String

3. Практическая работа

1

Декомпозиция кода с помощью статических методов

- 1.1 Решим задачу. Напишите приложение „Калькулятор”, выполняющее 4 действия: сложение, вычитание, умножение, деление.

```
public static void main(String[] args) {
    double x, y;
    double sum;
    double difference;
    double composition;
    double quotient;

    Scanner sc = new Scanner(System.in);

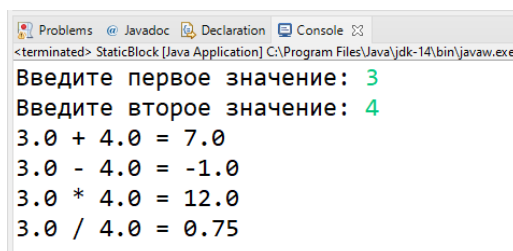
    System.out.print("Введите первое значение: ");
    while(!sc.hasNextDouble()) {
        sc.nextLine();
        System.out.print("Неверный ввод. Введите первое значение: ");
    }
    x = sc.nextDouble();

    System.out.print("Введите второе значение: ");
    while(!sc.hasNextDouble()) {
        sc.nextLine();
        System.out.print("Неверный ввод. Введите второе значение: ");
    }
    y = sc.nextDouble();

    sum = x + y;
    difference = x - y;
    composition = x * y;
    quotient = x / y;

    System.out.println(x + " + " + y + " = " + sum);
    System.out.println(x + " - " + y + " = " + difference);
    System.out.println(x + " * " + y + " = " + composition);
    System.out.println(x + " / " + y + " = " + quotient);
}
```

Результат:



```
<terminated> StaticBlock [Java Application] C:\Program Files\Java\jdk-14\bin\javaw.exe
Введите первое значение: 3
Введите второе значение: 4
3.0 + 4.0 = 7.0
3.0 - 4.0 = -1.0
3.0 * 4.0 = 12.0
3.0 / 4.0 = 0.75
```

Как видно из кода программы переиспользовать какие-либо части примера можно только с применением копирования, что является неудобным способом в работе и приводит к размножению ошибок и необходимости исправлять код в нескольких местах.

При написании кода нужно же стремиться к его переиспользованию, возможности несложной поддержки, расширяемости и простому чтению кода.

Одним из инструментов, позволяющих повысить переиспользование кода, являются методы.

1.2 Разберемся, как методы реализуются и используются. Сократим код программы „Калькулятор” до работы только с возможностью сложения.

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

    double x, y;
    double sum;

    Scanner sc = new Scanner(System.in);

    System.out.print("Введите первое значение: ");
    while(!sc.hasNextDouble()) {
        sc.nextLine();
        System.out.print("Неверный ввод. Введите первое значение: ");
    }
    x = sc.nextDouble();

    System.out.print("Введите второе значение: ");
    while(!sc.hasNextDouble()) {
        sc.nextLine();
        System.out.print("Неверный ввод. Введите второе значение: ");
    }
    y = sc.nextDouble();

    sum = x + y;

    System.out.println(x + " + " + y + " = " + sum);
}
```

1.3 Перепишем код следующим образом, выделив действия по сложению двух чисел в отдельный метод.

```
public class Task {

    public static void main(String[] args) {
        double x, y;
        double sum;

        Scanner sc = new Scanner(System.in);

        System.out.print("Введите первое значение: ");
        while(!sc.hasNextDouble()) {
            sc.nextLine();
            System.out.print("Неверный ввод. Введите первое
значение: ");
        }
        x = sc.nextDouble();

        System.out.print("Введите второе значение: ");
        while(!sc.hasNextDouble()) {
            sc.nextLine();
            System.out.print("Неверный ввод. Введите второе
значение: ");
        }
        y = sc.nextDouble();

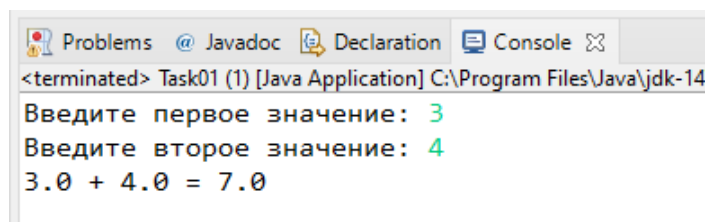
        addition(x, y);
    }

    public static void addition(double a, double b) {
        double sum;

        sum = a + b;

        System.out.println(a + " + " + b + " = " + sum);
    }
}
```

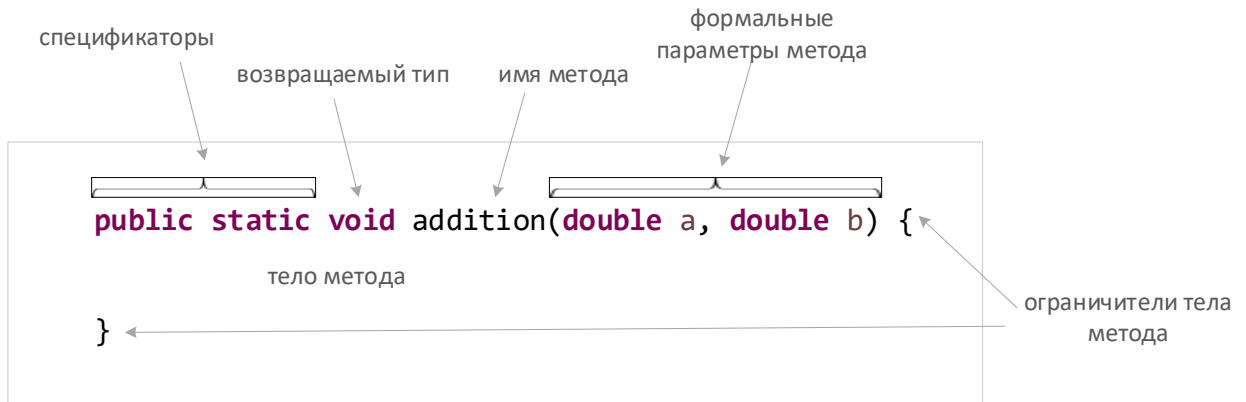
Результат:



```
<terminated> Task01 (1) [Java Application] C:\Program Files\Java\jdk-14
Введите первое значение: 3
Введите второе значение: 4
3.0 + 4.0 = 7.0
```

Объявление метода имеет следующую сигнатуру:

[спецификаторы] **возвращаемый_тип**
имя_метода([аргументы]) {/*тело метода*/} | ;



В данном модуле мы работаем только со статическими методами.

- 1.4 Напишем и разберем код, который передает в метод целое число, а метод это число увеличивает.

```
public class Task {  
  
    public static void main(String[] args) {  
  
        int x = 7;  
  
        System.out.println("in main: x = " + x);  
  
        method(x);  
  
        System.out.println("in main: x = " + x);  
    }  
  
    public static void method(int x) {  
        System.out.println("  in method: x = " + x);  
        x++;  
        System.out.println("  in method: x = " + x);  
    }  
}
```

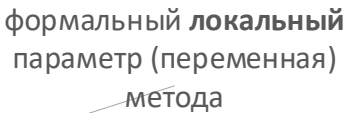
Результат:

The screenshot shows the output of the program in an IDE console. The output is as follows:
<terminated> Task02 (1) [Java Application] C:\Program Files\Java\jdk-14\bin
in main: x = 7
 in method: x = 7
 in method: x = 8
in main: x = 7

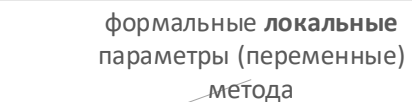
Методы в Java можно определить только в классе. При объявлении методы следуют друг за другом, нельзя написать метод внутри другого метода.

```
public class Example {  
    public static void method1() {  
  
    }  
    public static void method2(int x) {  
  
    }  
    public static void method3(String s) {  
  
    }  
    public static void method4(double[] mas, String s) {  
  
    }  
}
```

- 1.5 Объявленные в методе переменные являются локальными для метода, т.е. они создаются при вызове метода и уничтожаются автоматически после его завершения. Локальные переменные метода создаются и уничтожаются столько раз, сколько вызывается метод.

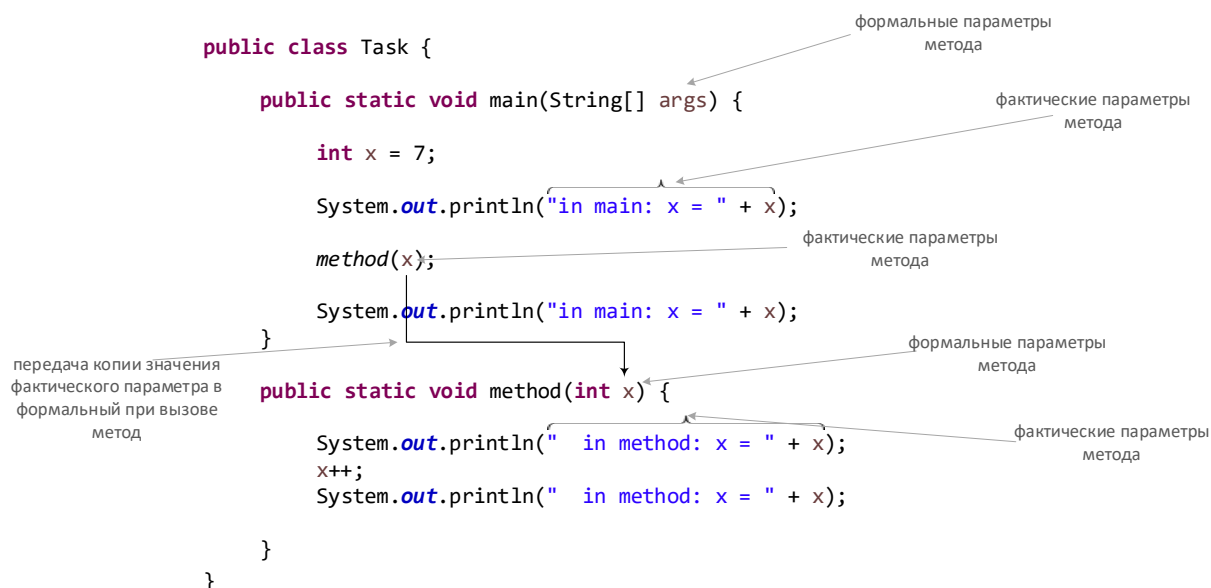


```
public static void method(int x) {  
    System.out.println("  in method: x = " + x);  
    x++;  
    System.out.println("  in method: x = " + x);  
}
```



```
public static void addition(double a, double b) {  
    double sum;  
    sum = a + b;  
    System.out.println(a + " + " + b + " = " + sum);  
}
```

- 1.6 Все параметры в методы **передаются только по значению**. Это значит, что при вызове метода создаются локальные формальные переменные метода и в них присваивается(передается) копия значения переменных, используемых при вызове этого метода (фактических переменных).



При передаче в метод параметров по значению изменение формальных параметров **не приводит к изменению** соответствующих им фактических параметров.

- 1.7 Теперь перепишем метод сложения программы „Калькулятор” таким образом, чтобы метод не только принимал параметры и выполнял ряд операторов, но и возвращал в точку вызова результат своей работы.

```
public static void main(String[] args) {  
    double x, y;  
    double sum;  
  
    Scanner sc = new Scanner(System.in);  
  
    System.out.print("Введите первое значение: ");  
    while(!sc.hasNextDouble()) {  
        sc.nextLine();  
        System.out.print("Неверный ввод. Введите первое значение: ");  
    }  
    x = sc.nextDouble();  
  
    System.out.print("Введите второе значение: ");  
    while(!sc.hasNextDouble()) {  
        sc.nextLine();  
    }  
}
```



```

        System.out.print("Неверный ввод. Введите второе значение: ");
    }
    y = sc.nextDouble();

    sum = addition(x, y);

    System.out.println(x + " + " + y + " = " + sum);
}

public static double addition(double a, double b) {
    double sum;

    sum = a + b;

    return sum;
}

```

Результат:

```

<terminated> Task01 (1) [Java Application] C:\Program Files\Java\jdk-14\
Введите первое значение: 3
Введите второе значение: 4
3.0 + 4.0 = 7.0

```

Для завершения работы метода и возврата значения в точку вызова метода используется оператор **return**. При реализации метода необходимо, чтобы любой возможный путь выполнения метода оканчивался оператором **return**.

```

public static void main(String[] args) {
    double x, y;
    double sum;

    ...
    y = sc.nextDouble();
    sum = addition(x, y);
    System.out.println(x + " + " + y + " = " + sum);
}

public static double addition(double a, double b) {
    double sum;

    sum = a + b;

    return sum;
}

```

точка вызова метода `nextDouble()`
 точка вызова метода `addition()`
 передача копий значений фактических параметров в формальные при вызове метод
 результатом работы метода является какое-то значение типа **double**
 возврат значения локальной переменной `sum` в точку вызова метода `addition`

В методах с возвращаемым типом **void** оператор **return** также можно использовать для завершения работы метода.

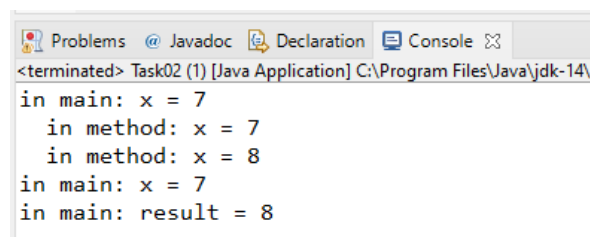
```
public static void division(int a, int b) {  
    if(b == 0) {  
        System.out.println("Деление на ноль невозможно.");  
        return;  
    }  
  
    // ...  
}
```

Оператор **return** – это оператор безусловного выхода из метода. При его выполнении происходит возврат в точку вызова без каких-либо условий.

- 1.8 Изменив возвращаемый тип метода можно написать код, возвращающий измененное состояние локальной переменной метода, и передать новое значение в точку вызова метода.

```
public class Task {  
    public static void main(String[] args) {  
  
        int x = 7;  
        int result;  
  
        System.out.println("in main: x = " + x);  
  
        result = method(x);  
  
        System.out.println("in main: x = " + x);  
        System.out.println("in main: result = " + result);  
    }  
  
    public static int method(int x) {  
        System.out.println("in method: x = " + x);  
        x++;  
        System.out.println("in method: x = " + x);  
        return x;  
    }  
}
```

Результат:



```
<terminated> Task02 (1) [Java Application] C:\Program Files\Java\jdk-14\  
in main: x = 7  
in method: x = 7  
in method: x = 8  
in main: x = 7  
in main: result = 8
```

```

public class Task {

    public static void main(String[] args) {

        int x = 7;

        System.out.println("in main: x = " + x);

        x = method(x);

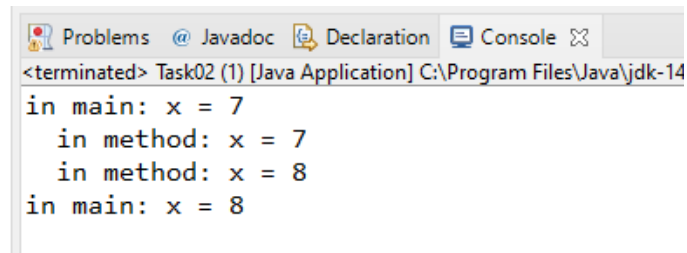
        System.out.println("in main: x = " + x);

    }

    public static int method(int x) {
        System.out.println("  in method: x = " + x);
        x++;
        System.out.println("  in method: x = " + x);
        return x;
    }
}

```

Результат:



```

<terminated> Task02 (1) [Java Application] C:\Program Files\Java\jdk-14
in main: x = 7
  in method: x = 7
  in method: x = 8
in main: x = 8

```

- 1.9 Рассмотрим ряд модификаций программы „Калькулятор”. Сначала выделим в отдельный метод код, читающий значения с клавиатуры, а потом код, выводящий результат вычисления на консоль.

```

public class Task{

    public static void main(String[] args) {
        double x, y;
        double sum;

        x = inputDoubleFromConsole();
        y = inputDoubleFromConsole();

        sum = addition(x, y);
    }
}

```

```

        System.out.println(x + " + " + y + " = " + sum);
    }

    public static double inputDoubleFromConsole() {
        double value;
        Scanner sc = new Scanner(System.in);
        System.out.print("Введите значение: ");
        while (!sc.hasNextDouble()) {
            sc.nextLine();
            System.out.print("Неверный ввод. Введите значение: ");
        }
        value = sc.nextDouble();

        return value;
    }

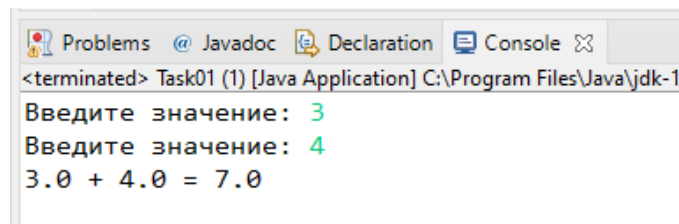
    public static double addition(double a, double b) {
        double sum;

        sum = a + b;

        return sum;
    }
}

```

Результат:



```

<terminated> Task01 (1) [Java Application] C:\Program Files\Java\jdk-1
Введите значение: 3
Введите значение: 4
3.0 + 4.0 = 7.0

```

```

public class Task{

    public static void main(String[] args) {
        double x, y;
        double sum;

        x = inputDoubleFromConsole("Введите первое значение: ");
        y = inputDoubleFromConsole("Введите второе значение: ");

        sum = addition(x, y);

        System.out.println(x + " + " + y + " = " + sum);
    }
}

```

```

    }

    public static double inputDoubleFromConsole(String message) {
        double value;
        Scanner sc = new Scanner(System.in);
        System.out.print(message);
        while (!sc.hasNextDouble()) {
            sc.nextLine();
            System.out.print("Неверный ввод. " + message);
        }
        value = sc.nextDouble();

        return value;
    }

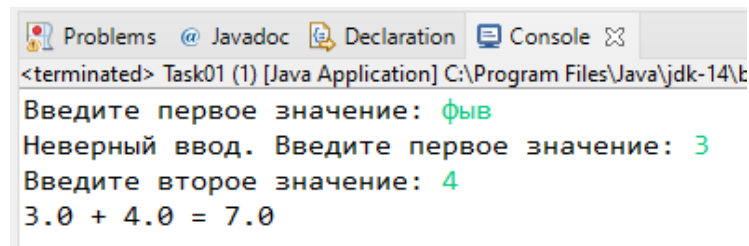
    public static double addition(double a, double b) {
        double sum;

        sum = a + b;

        return sum;
    }
}

```

Результат:



```

<terminated> Task01 (1) [Java Application] C:\Program Files\Java\jdk-14\bin\java.exe
Введите первое значение: 3.0
Неверный ввод. Введите первое значение: 3
Введите второе значение: 4
3.0 + 4.0 = 7.0

```

```

public class Task {

    public static void main(String[] args) {
        double x, y;
        double sum;

        x = inputDoubleFromConsole("Введите первое значение: ");
        y = inputDoubleFromConsole("Введите второе значение: ");

        sum = addition(x, y);

        simplePrint(x, y, sum, '+');
        richPrint(x, y, sum, '+');
    }
}

```

```

    }

    public static double inputDoubleFromConsole(String message) {
        double value;
        Scanner sc = new Scanner(System.in);
        System.out.print(message);
        while (!sc.hasNextDouble()) {
            sc.nextLine();
            System.out.print("Неверный ввод. " + message);
        }
        value = sc.nextDouble();

        return value;
    }

    public static double addition(double a, double b) {
        double sum;

        sum = a + b;

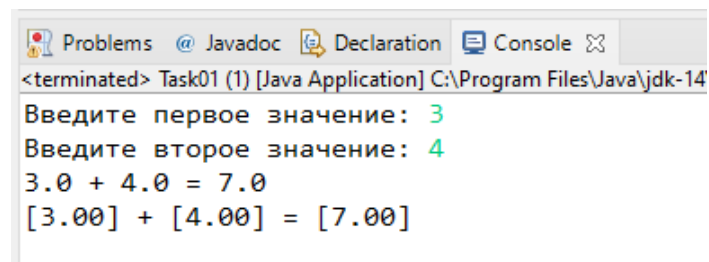
        return sum;
    }

    public static void simplePrint(double a, double b, double rez, char sign) {
        System.out.println(a + " " + sign + " " + b + " = " + rez);
    }

    public static void richPrint(double a, double b, double rez, char sign) {
        System.out.printf("[%2f] %c [%2f] = [%2f]", a, sign, b, rez);
    }
}

```

Результат:



```

<terminated> Task01 (1) [Java Application] C:\Program Files\Java\jdk-14
Введите первое значение: 3
Введите второе значение: 4
3.0 + 4.0 = 7.0
[3.00] + [4.00] = [7.00]

```

2.1 Объявим две строки в приложении.

```
public static void main(String[] args) {  
    String str1 = new String("Java");  
    String str2 = "Java";  
}
```

Строки в Java можно объявлять в двух видах: полном и сокращенном. Полный – это с использованием классического вида объявления ссылки и создания объекта и вызовом конструктора.

```
String str1 = new String("Java");
```

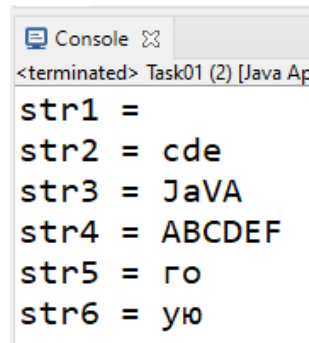
Но так как строки являются самым частым используемым объектом, для его объявления ввели “синтаксический сахар” – разрешили присваивать ссылке литерал-строку, и явно не вызывать конструктор.

```
String str2 = "Java";
```

2.2 Создать объект класса String можно и другими способами. Рассмотрим пример.

```
public static void main(String[] args) throws  
    UnsupportedEncodingException {  
    String str1 = new String();  
  
    char[] data1 = { 'a', 'b', 'c', 'd', 'e', 'f' };  
    String str2 = new String(data1, 2, 3);  
  
    char[] data2 = { '\u0041', '\u0061', 'V', 'A' };  
    String str3 = new String(data2);  
  
    byte ascii[] = { 65, 66, 67, 68, 69, 70 };  
    String str4 = new String(ascii);  
  
    byte[] data3 = { (byte) 0xE3, (byte) 0xEE };  
    String str5 = new String(data3, "CP1251");  
    String str6 = new String(data3, "CP866");  
  
    System.out.println("str1 = " + str1);  
    System.out.println("str2 = " + str2);  
    System.out.println("str3 = " + str3);  
    System.out.println("str4 = " + str4);  
    System.out.println("str5 = " + str5);  
}
```

```
        System.out.println("str6 = " + str6);
    }
    Результат:
```



```
<terminated> Task01 (2) [Java Ap
str1 =
str2 = cde
str3 = JaVA
str4 = ABCDEF
str5 = го
str6 = ую
```

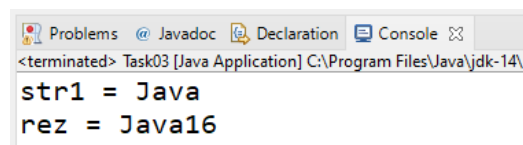
2.3 Строки в языке Java – это константные объекты, т.е. объекты, состояние которых нельзя изменить. При попытке изменить строку виртуальной машиной создается новый объект строки с измененным состоянием. Исследуем поведение строки при конкатенации.

```
public static void main(String[] args) {
    String str1 = "Java";
    int x = 16;

    String rez = str1 + x;

    System.out.println("str1 = " + str1);
    System.out.println("rez = " + rez);
}
```

Результат:



```
Problems @ Javadoc Declaration Console
<terminated> Task03 [Java Application] C:\Program Files\Java\jdk-14\
str1 = Java
rez = Java16
```

В Java можно конкатенировать (объединить) строку с переменной любого другого типа данных. Для этого используется единственный перегруженный оператор "+". Результатом конкатенации со строкой всегда является другая строка (исходная строка при этом не изменяется).

Исследуем поведение строки при применении метода `replace` – заменяющего в строке одни последовательности символов на другие.


```

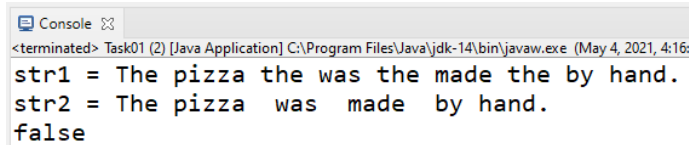
public static void main(String[] args) {

    String str1 = "The pizza the was the made the by hand.";
    String str2;

    str2 = str1.replace("the", "");
    System.out.println("str1 = " + str1);
    System.out.println("str2 = " + str2);
    System.out.println(str1 == str2);
}

```

Результат:



```

Console
<terminated> Task01 (2) [Java Application] C:\Program Files\Java\jdk-14\bin\javaw.exe (May 4, 2021, 4:16:
str1 = The pizza the was the made the by hand.
str2 = The pizza was made by hand.
false

```

Обязательно нужно отличать операторы:

`String str = null;` - ссылка не ссылается ни на какой объект, а при попытке вызвать любой метод будет сгенерировано исключение `java.lang.NullPointerException`.

`String str = "";` - ссылка ссылается на объект строки, но строка не содержит ни одного символа – это пустая строка. И вызов метода `length()` на такой ссылке (`int size = str.length();`) вернет значение 0.

2.4 При выполнении конкатенации также следует учитывать приоритет операций.

```

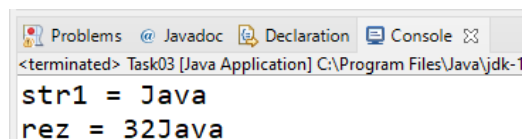
public static void main(String[] args) {
    String str1 = "Java";
    int x = 16;

    String rez = x + x + str1;

    System.out.println("str1 = " + str1);
    System.out.println("rez = " + rez);
}

```

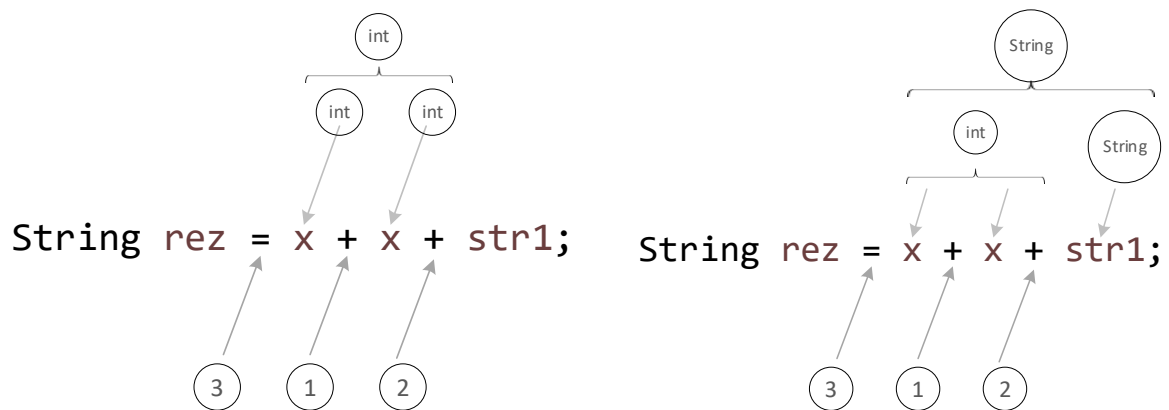
Результат:



```

Problems @ Javadoc Declaration Console
<terminated> Task03 [Java Application] C:\Program Files\Java\jdk-1
str1 = Java
rez = 32Java

```



2.5 Исследуем ряд методов, которые часто применяются при работе со строкой.

```
public static void main(String[] args) {
    String str1 = "Java";
    String str2 = "JAVA";

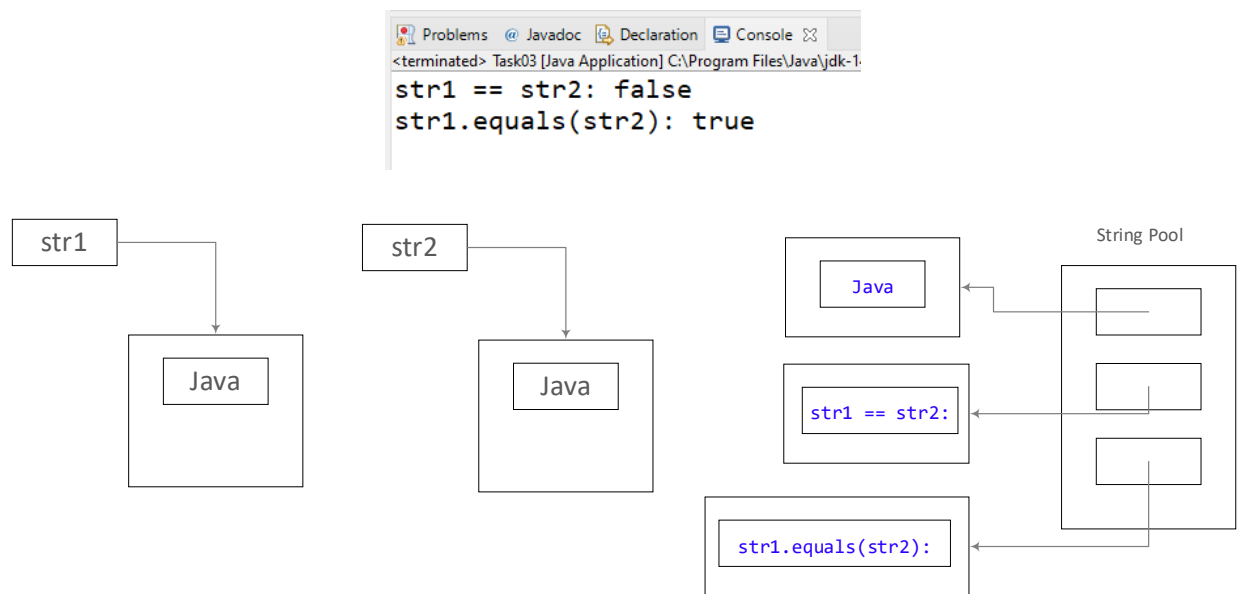
    System.out.println("Длина строки: " + str1.length());
    System.out.println("Возвращение символа по индексу: " +
        str1.charAt(0));
    System.out.println("Сравнение строк не лексикографическое равенство: "
        + str1.equals(str2));
    System.out.println("Сравнение строк <> 1: " + str1.compareTo(str2));
    System.out.println("Сравнение строк <> 2: " + str2.compareTo(str1));
    System.out.println("Сравнение строк <> 3: " + "Java".compareTo(str1));
    System.out.println("Сравнение без учета регистра символов: " +
        str1.compareToIgnoreCase(str2));
    System.out.println("Проверка строки на \"пустоту\": " +
        str1.isEmpty());
    System.out.println("Приведение символов строки к верхнему регистру: " +
        str1.toUpperCase());
    System.out.println("Приведение символов строки к нижнему регистру: " +
        str1.toLowerCase());
}
```

2.6 Исследуем особенности сравнения строк в Java.

```
public static void main(String[] args) {
    String str1 = new String("Java");
    String str2 = new String("Java");

    System.out.println("str1 == str2: " + (str1 == str2));
    System.out.println("str1.equals(str2): " + (str1.equals(str2)));
}
```

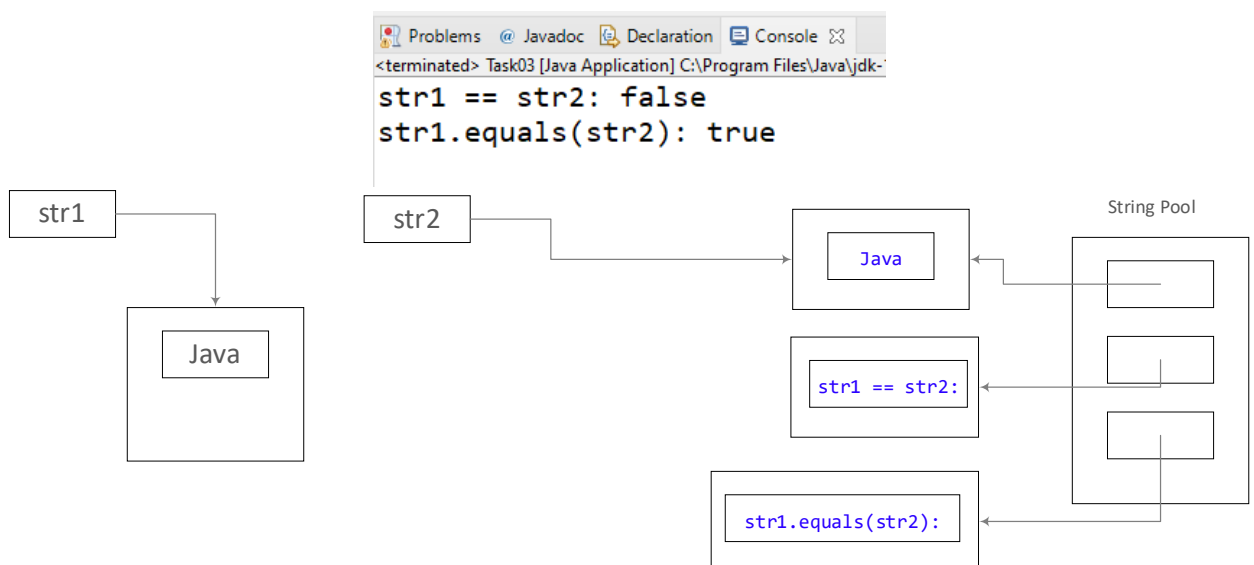
Результат:



2.7 Исследуем особенности сравнения строк в Java.

```
public static void main(String[] args) {  
    String str1 = new String("Java");  
    String str2 = "Java";  
  
    System.out.println("str1 == str2: " + (str1 == str2));  
    System.out.println("str1.equals(str2): " + (str1.equals(str2)));  
}
```

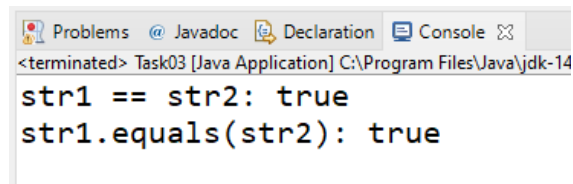
Результат:



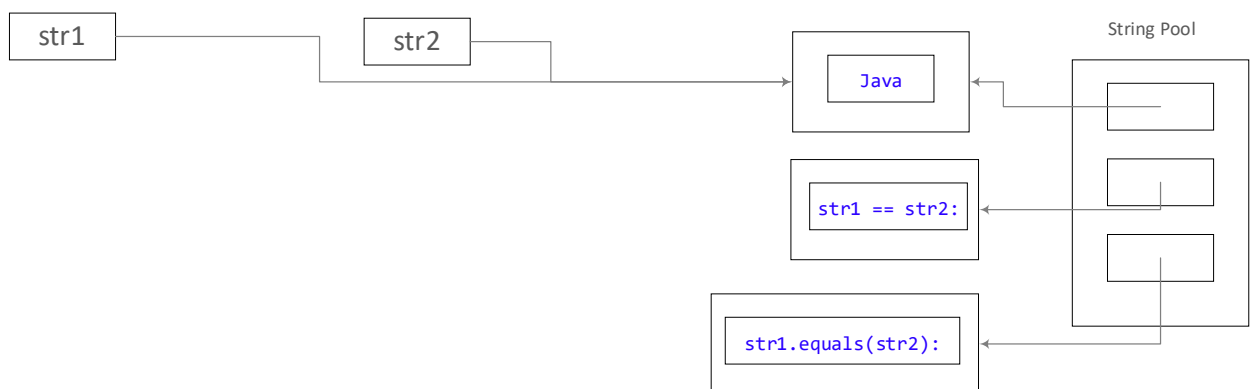
2.8 Исследуем особенности сравнения строк в Java.

```
public static void main(String[] args) {  
    String str1 = "Java";  
    String str2 = "Java";  
  
    System.out.println("str1 == str2: " + (str1 == str2));  
    System.out.println("str1.equals(str2): " + (str1.equals(str2)));  
}
```

Результат:



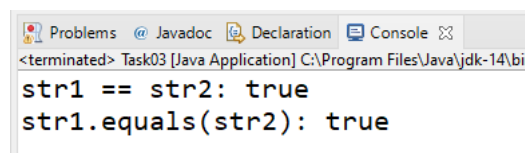
```
<terminated> Task03 [Java Application] C:\Program Files\Java\jdk-14  
str1 == str2: true  
str1.equals(str2): true
```



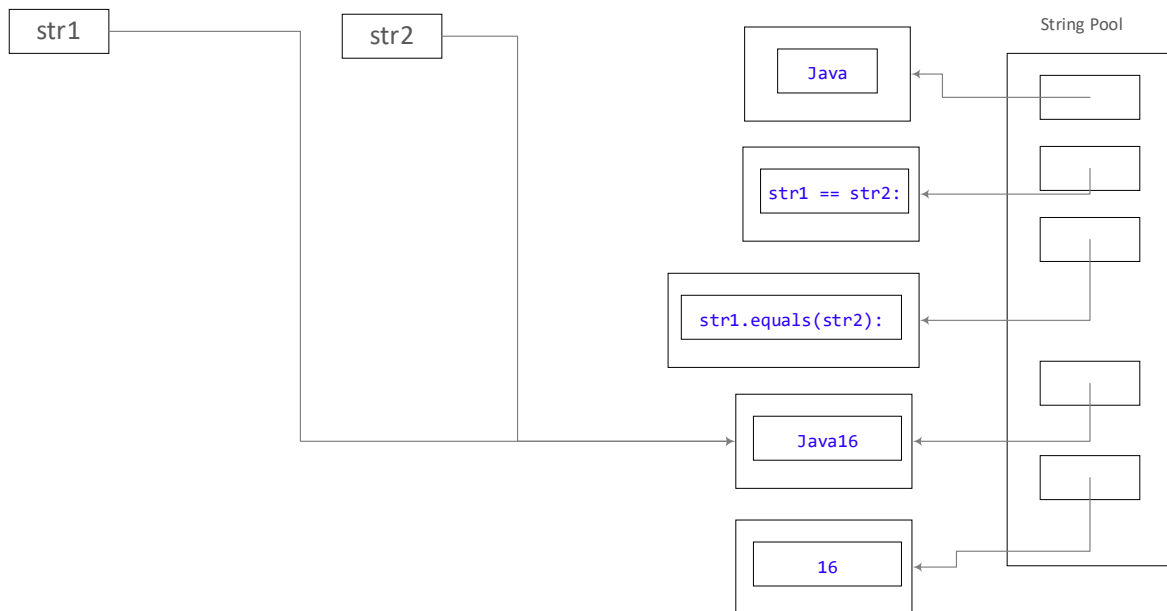
2.9 Исследуем особенности сравнения строк в Java.

```
public static void main(String[] args) {  
    String str1 = "Java16";  
    String str2 = "Java" + "16";  
  
    System.out.println("str1 == str2: " + (str1 == str2));  
    System.out.println("str1.equals(str2): " + (str1.equals(str2)));  
}
```

Результат:



```
<terminated> Task03 [Java Application] C:\Program Files\Java\jdk-14\bin  
str1 == str2: true  
str1.equals(str2): true
```



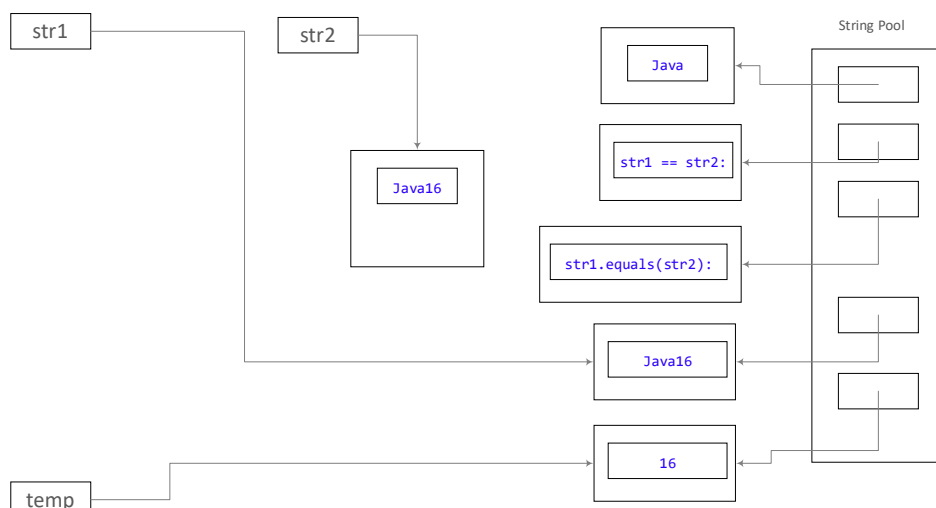
2.10 Исследуем особенности сравнения строк в Java.

```
public static void main(String[] args) {
    String str1 = "Java16";
    String temp = "16";
    String str2 = "Java" + temp;

    System.out.println("str1 == str2: " + (str1 == str2));
    System.out.println("str1.equals(str2): " + (str1.equals(str2)));
}
```

Результат:

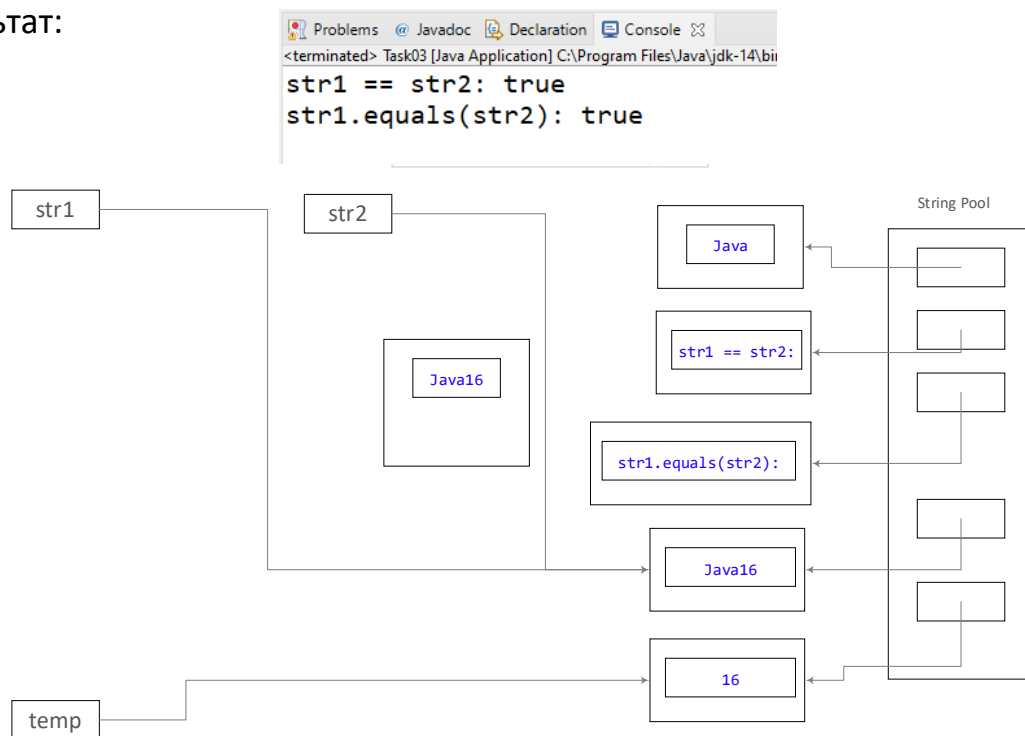
```
<terminated> Task03 [Java Application] C:\Program Files\Java\jdk-14\
str1 == str2: false
str1.equals(str2): true
```



2.11 Исследуем особенности сравнения строк в Java.

```
public static void main(String[] args) {  
    String str1 = "Java16";  
  
    String temp = "16";  
    String str2 = "Java" + temp;  
  
    str2 = str2.intern();  
    System.out.println("str1 == str2: " + (str1 == str2));  
    System.out.println("str1.equals(str2): " + (str1.equals(str2)));  
}
```

Результат:

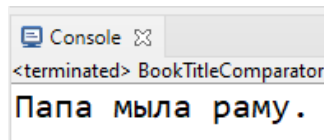


2.12 Исследуем ряд методов, которые часто применяются при работе со строкой.

String replace(CharSequence target, CharSequence replacement) - замена одной подстроки другой

```
public static void main(String[] args) {  
    String str = "Мама мыла раму."  
    str = str.replace("Мама", "Папа");  
    System.out.println(str);  
}
```

Результат:

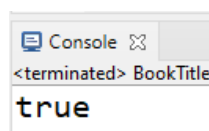


```
Console
<terminated> BookTitleComparator
Папа мыла раму.
```

boolean contains(CharSequence cs) - проверяет, входит ли указанная последовательность символов в строку

```
public static void main(String[] args) {
    String str = "Мама мыла раму.";
    System.out.println(str.contains("мыла"));
}
```

Результат:

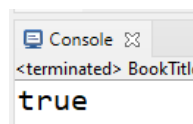


```
Console
<terminated> BookTitle
true
```

boolean endsWith(String suffix) - заканчивается ли String суффиксом suffix

```
public static void main(String[] args) {
    String str = "Мама мыла раму.";
    System.out.println(str.endsWith("y."));
}
```

Результат:

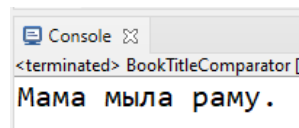


```
Console
<terminated> BookTitle
true
```

String trim() – отсекает на концах строки пустые символы

```
public static void main(String[] args) {
    String str = "    Мама мыла раму.    ";
    System.out.println(str.trim());
}
```

Результат:

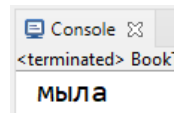


```
Console
<terminated> BookTitleComparator
Мама мыла раму.
```

String substring(int startIndex, int endIndex) – возвращает подстроку с beginIndex до endIndex

```
public static void main(String[] args) {  
    String str = "Мама мыла раму."  
    System.out.println(str.substring(4, 9));  
}
```

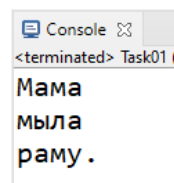
Результат:



String[] split(String regexStr) - разбивает строку на части, границами разбиения служит выражение, передаваемое в качестве входного параметра

```
public static void main(String[] args) {  
    String str = "Мама мыла раму."  
    String[] parts;  
  
    parts= str.split(" ");  
    for(String s : parts) {  
        System.out.println(s);  
    }  
}
```

Результат:



Для того, чтобы в качестве разделителя учитывать любое количество пробельных символов, идущих подряд, следует использовать регулярное выражение `\s+`. Помните, что символ `\` в Java задает управляющий символ, и чтобы при обработке строки получить “\” в строке нужно написать `\\`.

```
public static void main(String[] args) {  
    String str = "Мама    мыла        раму."  
    String[] parts;  
  
    parts= str.split("\\s+");
```

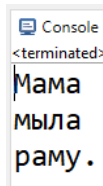


```

    for(int i=0; i<parts.length; i++) {
        System.out.println(parts[i]);
    }
}

```

Результат:



```

Console
<terminated>
Мама
мыла
раму.

```

3

Практическая работа

Подготовка рабочего пространства для решения задач

1. Создайте проект с именем Unit05[Surname] (например, Unit05Ivanov).
2. Далее создайте класс Task01 и поместите его в пакет It.lhu.unit05.main.
3. Аналогично создайте еще 3 класса Task02 ... Task04.
4. В каждом классе создайте метод main и пишите в нем код решения задачи из списка задач с таким же номером, как и у класса.

Список задач

Задача 1.

Написать метод(методы) для нахождения наибольшего общего делителя и наименьшего общего кратного двух натуральных чисел:

$$\left(HOK(A, B) = \frac{A \cdot B}{НОД(A, B)} \right)$$

Задача 2.

Написать метод(методы) для нахождения суммы большего и меньшего из трех чисел.

Задача 3.

Написать метод(методы), определяющий, в каком из данных двух чисел больше цифр.

Задача 4.

Задан массив D. Определить следующие суммы: $D[1] + D[2] + D[3]$; $D[3] + D[4] + D[5]$; $D[4] + D[5] + D[6]$.

Пояснение. Составить метод(методы) для вычисления суммы трех последовательно расположенных элементов массива с номерами от k до m.

