Курс: «Разработка программного обеспечения на Java»

Тема: Строки, массивы одномерные, многомерные

План

1. Работа со строками
2. Что такое массивы? Работа с массивами. Примеры использования
3. Многомерные массивы на примере двумерных массивов

1. Работа со строками

Строка – это объект, который представляет собой последовательность символов. Для работы со строками используйте класс java.lang.String. Класс String является неизменяемым (immutable), а значит, объект данного класса не может быть изменен, поэтому все методы, которые проводят изменения в строке возвращают новый экземпляр класса String.

Для проведения изменений в строке без пересоздания нового экземпляра следует использовать специальные классы: StringBuffer и StringBuilder, которые допускают изменения в строке. Указанные классы, включая класс String, реализуют интерфейс CharSequence.

Благодаря неизменяемости строк, платформа Java предоставляет возможность использовать строки в многопоточных средах и в качестве Map-ключей.

Создание строки

1) Создание нового экземпляра пустой строки
String s = new String();

2) Создание нового экземпляра строки
String s = new String("abc");

3) Сокращенное создание строки
String s ="abc";

4) Создание строки из массива символов
char chars[] = { 'a', 'b', 'c' }:
String s = new String(chars);

Оператор конкатенации строк

В Java нет возможности переопределять операторы для классов, исключением является класс String. Для него определены операторы объединения срок: + и +=.

String part1 = "abc";
String part2 = "def";
String str = part1 + part2; // str = abcdef

part1 += part2; // part1 = abcdef

Оператор += объединяет обе строки, при этом инициализируя новую строку и присваивая ее первому операнду.

Основные методы String-класса

Описание: Объединение строк
Метод: String concat (String string)
Пример:
String str1 = "abc";
String str2 = str1.concat("def"); // str2 = abcdef

Описание: Получение символа строки по номеру позиции
Метод: char charAt (int index)
Пример:
String str = "abcdef";
char ch = str.charAt(3); // ch = d

Описание: Содержит ли строка некоторую последовательность символов
Метод: boolean contains (CharSequence cs)
Пример:
String str = "автомобиль";
boolean b = str.contains("авто"); // b = true

Описание: Возвращает номер первой встречной позиции указанного символа или строки
Метод: int indexOf(String str)
Пример:
String str = "he**l**lo world!";
int position = str.indexOf("l"); // position = 2

Описание: Возвращает номер последней встречной позиции указанного символа или строки
Метод: int lastIndexOf(String str))
Пример:
String str = "hello wor**l**d!";
int position = str.lastIndexOf("l"); // position = 9

Описание: Является ли строка пустой
Метод: boolean isEmpty()

Описание: Возвращает длину строки
Метод: int length()

Описание: Удаляет пробелы в начале и в конце строки
Метод: String trim()

Сравнение строк

В языке Java можно использовать для сравнения строк перегруженный оператор == иди метод **equals()**. Важно понимать разницу между этими способами.

Оператор == сравнивает ссылки на объекты, т.е. у двух строках, которые имеют одинаковые символы могут быть разные ссылки, результат будет false.

Метод equals() сравнивает конкретно содержимое строки, если строки содержат одинаковые символы, тогда результатом сравнение будет true.

В Java присутствует особенность, благодаря которой разные строки могут содержать одинаковые ссылки. Данная особенность проявляется тогда, когда программист объявляет примитивные строки. Такие строки индексируются во время выполнения и, когда объявляется строка с аналогичным содержимым, ей присваивается ранее созданная ссылка.

Рассмотрим на примере:

```
String s1 = "hello";
String s2 = "hello";
s1 == s2; // true
s1.equals(s2); // true

String s1 = new String("hello");
String s2 = new String("hello");
s1 == s2; // false
s1.equals(s2); // true
```

2. Что такое массивы? Работа с массивами. Примеры использования

Массивы используются для удобной работы с большим количеством однотипных данных. Доступ к каждому элементу массива осуществляется по его индексу.

Размерностью массива называют общее количество элементов массива. Количество элементов массива определено и является конечным, а значит размер массива задаётся при инициализации массива и не может быть изменён в дальнейшем. Нумерация элементов массива начинается с 0, последний элемент массива имеет индекс n-1, где n - размер массива.

В языке Java массивы являются объектами. Имя массива указывает на адрес фрагмента данных в памяти, а размерность определяет смещение от нулевого элемента.

Синтаксис объявления массива:

<тип> [] <имя>;
<тип> <имя>[];

Для инициализации массива необходимо вызвать оператор new или перечислить элементы массива в фигурных скобках, тогда программа автоматически подсчитает их количество и инициализирует массив.

<имя массива> = new int[n]; // n - размерность массива
<имя массива> = {1, 2, 3, 4, 5};

Для определения размера массива следует использовать свойство lenght (int, возвращает длину массива).

Обращение к элементу массива происходит через имя массива и индекс элемента в квадратных скобках.

Пример:

```
int [] array = {10, 20, 30, 40, 50};
int element = array[2]; // element = 30
```

Если указать неверный индекс, произойдет Exception:

```
int [] array = {10, 20, 30, 40, 50};
int element = array[7]; // ArrayIndexOutOfBoundsException
```

Для работы с массивом удобно использовать циклы. К примеру, выведем на экран все элементы массива:

```
int [] array = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

for(int i = 0; i <= array.length - 1; i++) {
    System.out.print(array[i] + "  ");
}
```

Заполним элементы массива десятью случайными числами от 0 до 100:

```
int [] array = new int [10];
Random rand = new Random();

for(int i = 0; i < array.length; i++)
    array[i] = rand.nextInt(100);
```

Посчитаем сумму всех элементов массива:

```
int [] array = {50, 20, 60, 70, 30};
int sum = 0;

for(int i = 0; i < array.length; i++)
   sum += array[i];
```

Для гибкой работы с массивами используйте класс Arrays. У него есть ряд методов, которые позволяют сравнивать, копировать, заполнять массивы, осуществлять сортировку и поиск элементов.

Пример №1

```
int [] array1 = new int[10];
int [] array2 = new int[10];

Arrays.fill(array1, 1); // Заполним массив единицами
Arrays.fill(array2, 1); // Заполним массив единицами
Arrays.equals(a1, a2)); // Результат сравнения - true
```

Пример №2

```
int [] array = {2, 7, 8, 9, 10, 1, 3, 5, 2, 4};
Arrays.sort(array); // Быстрая сортировка массива, в результате array = 1, 2, 2, 3, 4, 5, 7, 8, 9, 10
```

3. Многомерные массивы на примере двумерных массивов

В предыдущем вопросе мы рассмотрели работу с одномерным массивом. Таким же образом в языке Java можно работать и с многомерными массивами. По сути, массив в массиве является двумерным массивом, тройная вложенность задает трехмерный массив и так далее.

Рассмотрим работу с многомерными массивами на примере двумерных массивов. Двумерный массив можно представить в виде матрицы, которая состоит из m x n элементов по высоте и ширине.

Объявление двумерного массива:

```
int [][] array;
```

Инициализация двумерного массива:

```
array = new int [m][n]; // m и n - целые числа
```

Упрощенная инициализация двумерного массива:

```
array = {{1, 2, 3},{4, 5, 6},{7, 8, 9}};
```

Обращение к элементу двумерного массива:

```
array[i][j];
```

Работа с двумерным массив осуществляется при помощи вложенных циклов. Заполним массив случайными числами:

```
int [][] array = new int [5][5];
Random rand = new Random();

for(int i = 0; i < 5; i++){
    for(int j = 0; j < 5; j++)
        array[i][j] = rand.nextInt(10);
}
```

Выведем элементы массива в консоль:

```
for(int i = 0; i < 5; i++){
    for(int j = 0; j < 5; j++){
        System.out.print(array[i][j]);
    }System.out.println();
}
```