



Массивы

Базовые конструкции языка Java. Отличительные особенности

Объявление:

<тип> <идентификатор> [];

<тип> []<идентификатор>;

Например, int[] B;

Описание:

Все элементы получают значения по умолчанию

<идентификатор> = **new** <тип> [выражение];

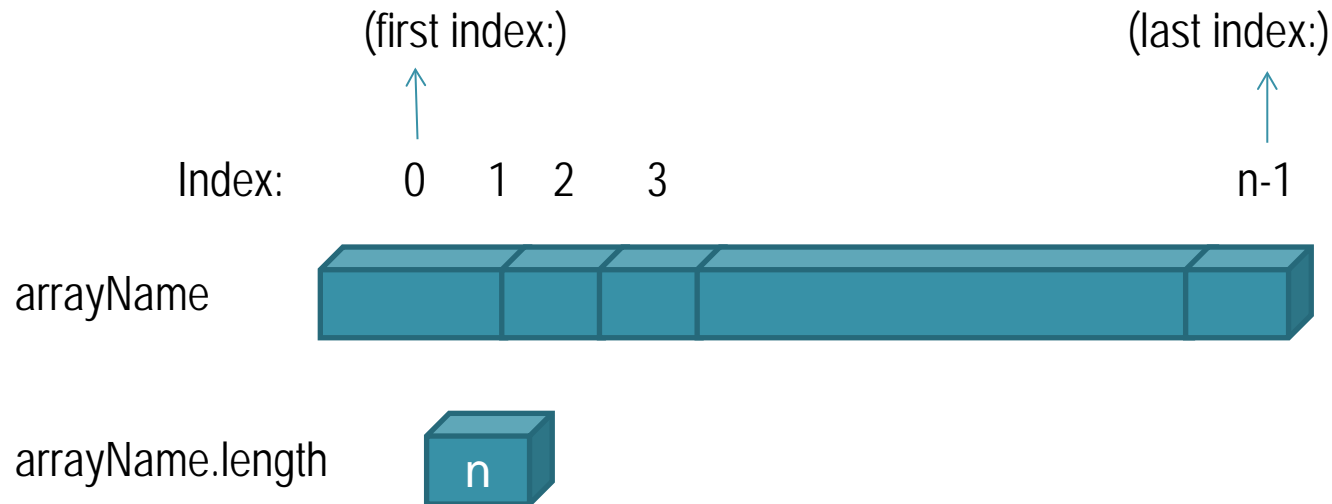
Например, B = new int [5];

Инициализация при описании:

<тип>[] <идентиф.> = {<литерал> {,<литерал>}₀^N};

Базовые конструкции языка Java. Отличительные особенности

- ❑ **Массив** – это объект-контейнер, который содержит фиксированное количество значений одного типа. А также длину (поле *length*).



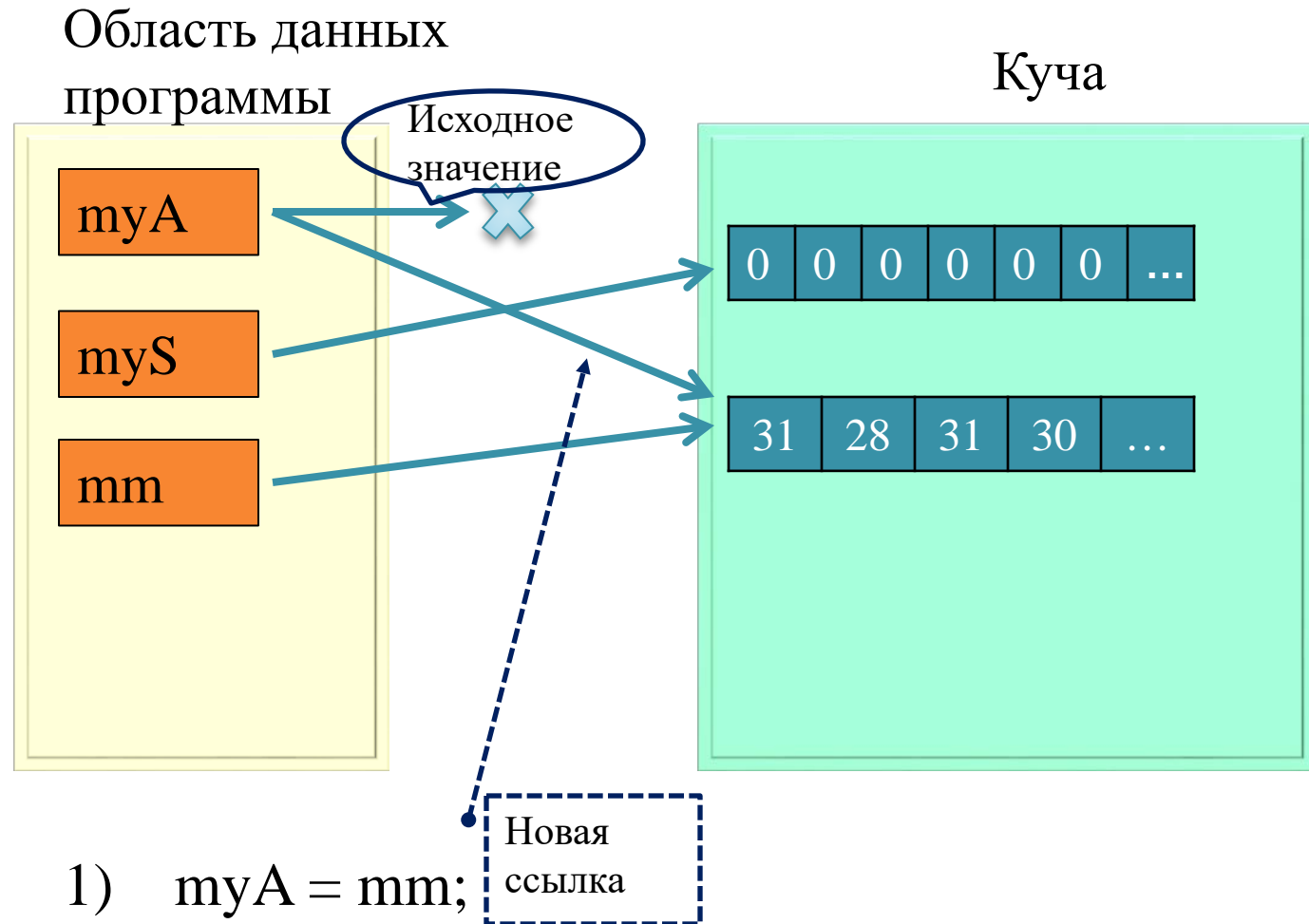
<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html>

Базовые конструкции языка Java. Отличительные особенности

Пример 1:

```
class Array {  
    public static void main(String [] args) {  
        int[] myA;  
        int[] myS = new int [100];  
        int[] mm = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};  
        System.out.println("Апрель содержит " + mm[3] + " дней");  
        for (int i=0; i < mm.length; i++) {  
            if (mm[i] < 31) {  
                mm[i] = 0;  
            }  
            myS[i] = mm[i];  
        }  
        myA = mm;  
    }  
}
```

Базовые конструкции языка Java. Отличительные особенности



Базовые конструкции языка Java. Отличительные особенности

- ❑ С Java 5 к операторам был добавлен новый улучшенный цикл **for-each** для итерирования элементов массива (набора данных):

```
int[] arr1 = { 1, 2, 3, 4, 5, 6 };
```


```
for ( int element : arr1 ) {  
    System.out.println(element);  
}
```

Нельзя изменить значение
элемента массива, только
ИСПОЛЬЗОВАТЬ

Базовые конструкции языка Java. Отличительные особенности

Пример 2:

```
public class FindReplace {  
    public static void main(String[] args){  
        int a[] = {5,10,0,-5,16,-2};  
        int max = a[0];  
        for (int value : a)  
            if (max < value)  
                max = value;  
        for (int i = 0; i < a.length; i++) {  
            if( a[i] < 0 )  
                a[i] = max;  
            System.out.println("a[" + i + "] = " + a[i]);  
        }  
    }  
}
```



Допускаем, что 1-й элемент имеет максимальное значение

Результат:

a[0]= 5
a[1]= 10
a[2]= 0
a[3]= 16
a[4]= 16
a[5]= 16

Многомерные массивы

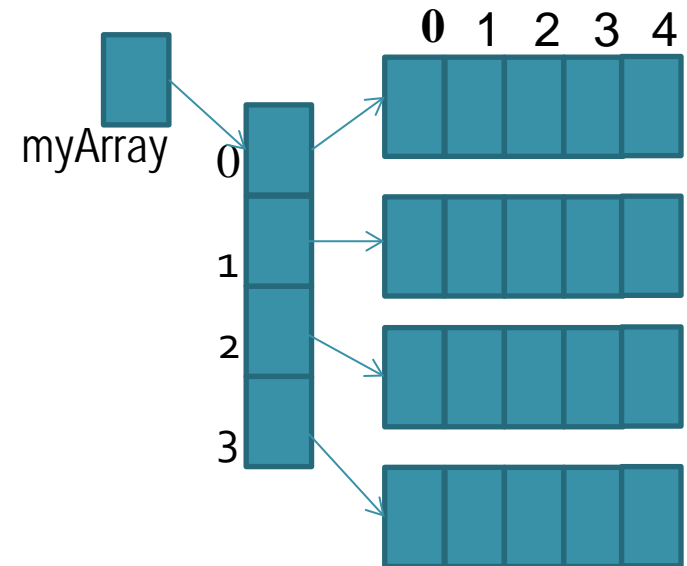
Описание двумерного массива:

<тип>[][] <идентификатор> =
new <тип>[выражение][выражение];

Выражение
должно иметь тип
int

Например:

int[][] myArray = **new** **int**[4][5];

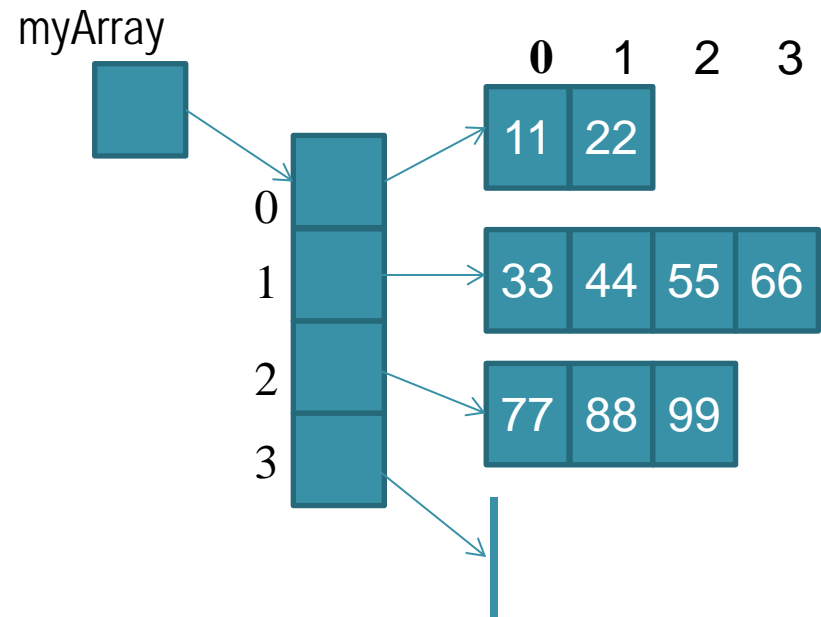


Рваные многомерные массивы

Например:

```
int[][] marr = {{11,22},{33,44,55,66},{77,88,99},{}};
```

```
int[][] arr = new int [4][];  
arr[0] = new int [5];  
arr[1] = new int [4];  
arr[2] = new int [3];  
arr[3] = new int [2];
```





Операции над массивом

Операции над массивом

- ❑ В библиотеке Java есть класс **Arrays**, который предоставляет статические методы для общих манипуляций с массивом:
 - *sort(array)* – располагает элементы массива в порядке возрастания;
 - *binarySearch(array , element)* – определяет содержит ли массив указанное значение и если да, то возвращает место размещения значения;
 - *equal(array)* – сравнивает массивы;
 - *fill(array , element)* – размещает значение во всем массиве;
 - *toString()* – преобразует массив в тип **String**.
- ❑ Для использования класса **Arrays** необходимо импортировать его: **import java.util.Arrays;**

Операции над массивом

- Также в классе **Arrays** содержится метод, выполняющий быстрое копирование массива: *copyOf()*, который аналогичен методу *arrayCopy()* класса **System**.

Например,

```
//исходный массив
int[] arr1 = { 1, 2, 3, 4, 5, 6 };
// другой массив
int[] arr2 = { 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 };
// копирование всех элементов arr1 в массив
// arr2, начиная с нулевого индекса
System.arrayCopy(arr1, 0, arr2, 0, arr1.length);
```

Ссылка на
исходный массив

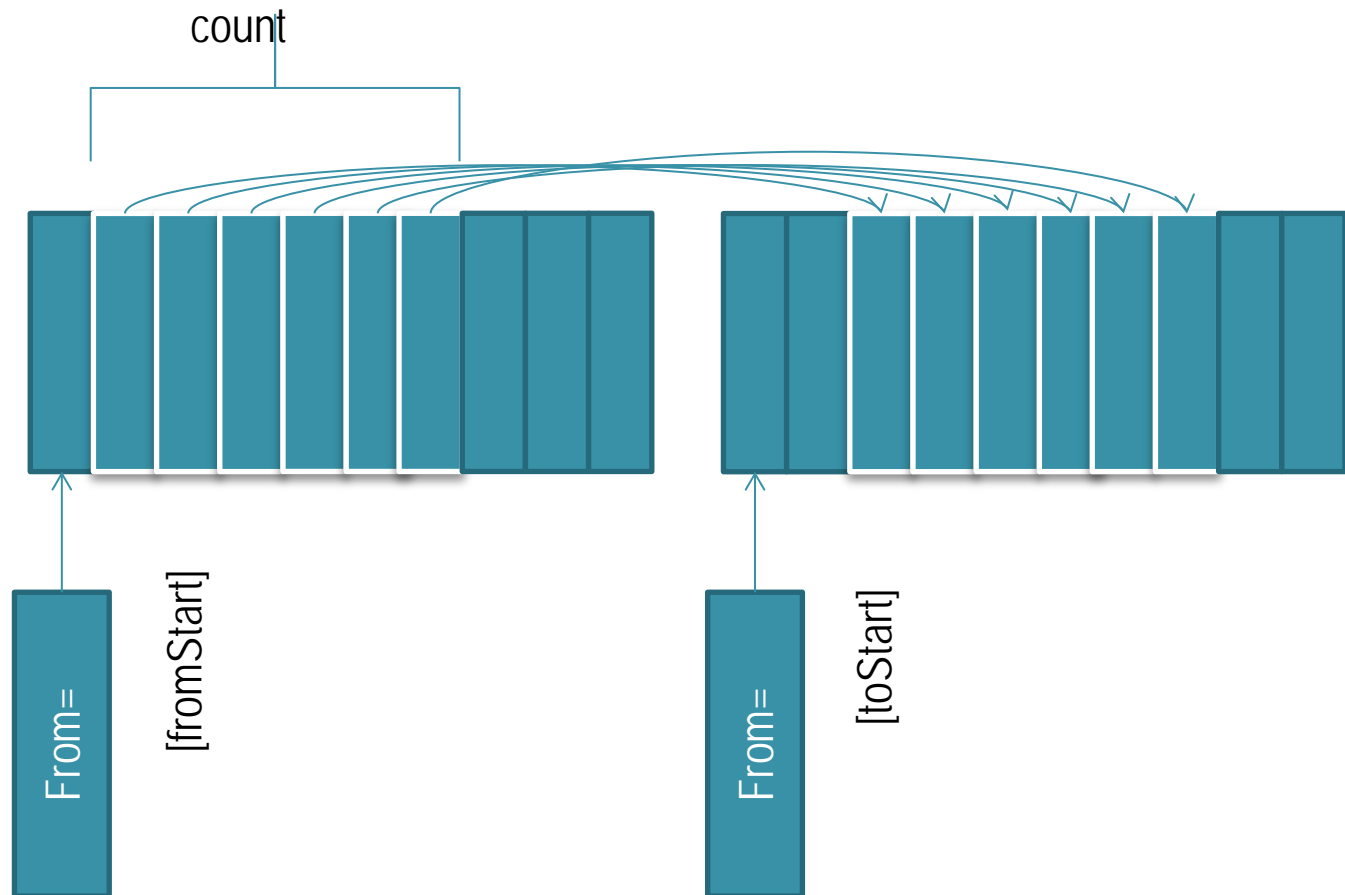
Индекс, с которого
начинается
копирование

Ссылка на массив
назначения

Индекс, в который
начинается
копирование

Количество
копируемых
элементов

Копирование массивов



Операции над массивом

- ❑ Для успешной работы метода *arrayCopy()* класса **System** необходимо:
 - Ссылки на исходный массив (*src*) и на целевой (*dest*) должны существовать (т.е. не быть **NULL**);
 - Значения индексов (*srcPos*, *destPos*) и длина (*length*) копируемых элементов не должны быть отрицательными;
 - Последний индекс копируемого элемента в исходном массиве (*srcPos+length-1*) и последний индекс в массиве целевом (*destPos+length-1*) должны существовать.:
 - ✓ В противном случае будет ошибка во время исполнения типа **ArrayIndexOutOfBoundsException**

Сравнение массивов

- ❑ Два массива считаются равными, если оба массива содержат то же самое количество элементов, и все соответствующие пары элементов в двух массивах равны.

```
int[] arr1 = {1,2,3,4,5,6,7,8,9};
```

```
int[] arr2 = {1,2,3,4,5,6,7,8,9};
```

```
int[] arr3 = {1,2,5,5,5,5,5,8,9};
```

```
// сравнение ссылок
```

```
System.out.println(arr1 == arr2);
```

```
// сравнение по элементам
```

```
System.out.println(arr1.equals(arr2));
```

```
// проверки тождественности
```

```
System.out.println(Arrays.equals(arr1, arr3));
```

```
System.out.println(Arrays.equals(arr2, arr3));
```

Сортировка и поиск

```
int intArr[] = {55, 57, 61, 66, 18, 19, 27, 38, 10, 11, 15, 39, 51, 82, 83, 95};
```

```
// сортировка вставками или быстрая по возрастанию
```

```
Arrays.sort(intArr);
```

```
// вывод массива
```

```
System.out.println("The sorted int array is:");
```

```
System.out.println(Arrays.toString(intArr));
```

```
// поиск значения
```

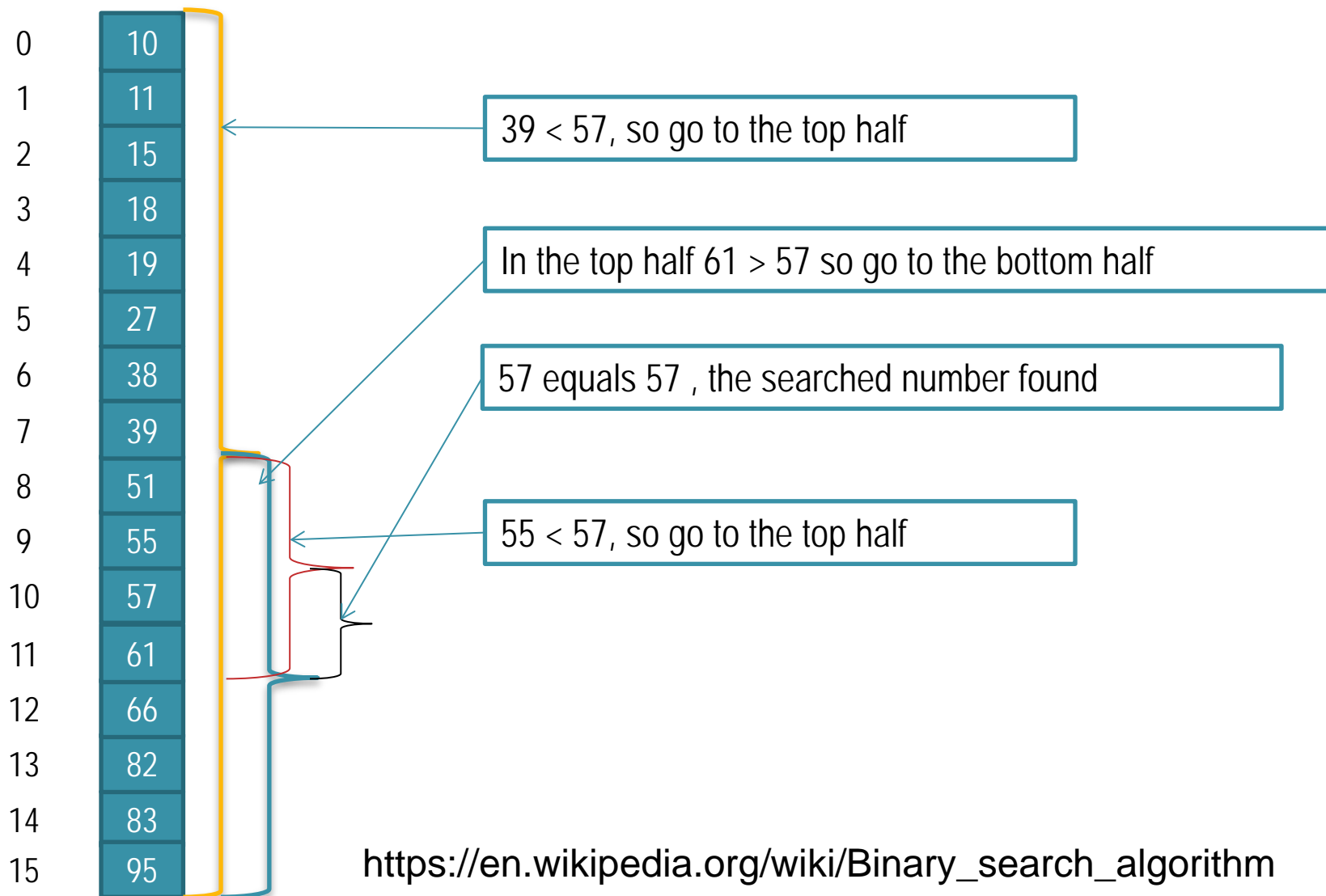
```
int searchVal = 57;
```

```
int retVal = Arrays.binarySearch(intArr, searchVal);
```

```
System.out.println("The index of element 57 is : " + retVal);
```

- ❑ Перед выполнением поиска массив должен быть отсортирован, иначе результат не определен.

Алгоритм бинарного поиска






Обработка строка

Класс String

- ❑ **Строка** – это последовательность символов, представляемая в Java как объект (пакет ***java.lang***).


Наиболее простой способ создания *строки* – использование *строкового литерала*:

```
String str = "text";
```



Строковый
литерал

```
String str_1 = "H";
```



Строка может
содержать
один символ

Создание строки с помощью конструктора

String()	- пустая строка;
String(String str)	- строка как копия другой строки;
String(byte[] ach)	- строка 8-разрядных символов;
String(char[] c)	- строка, инициализированная массивом символов;

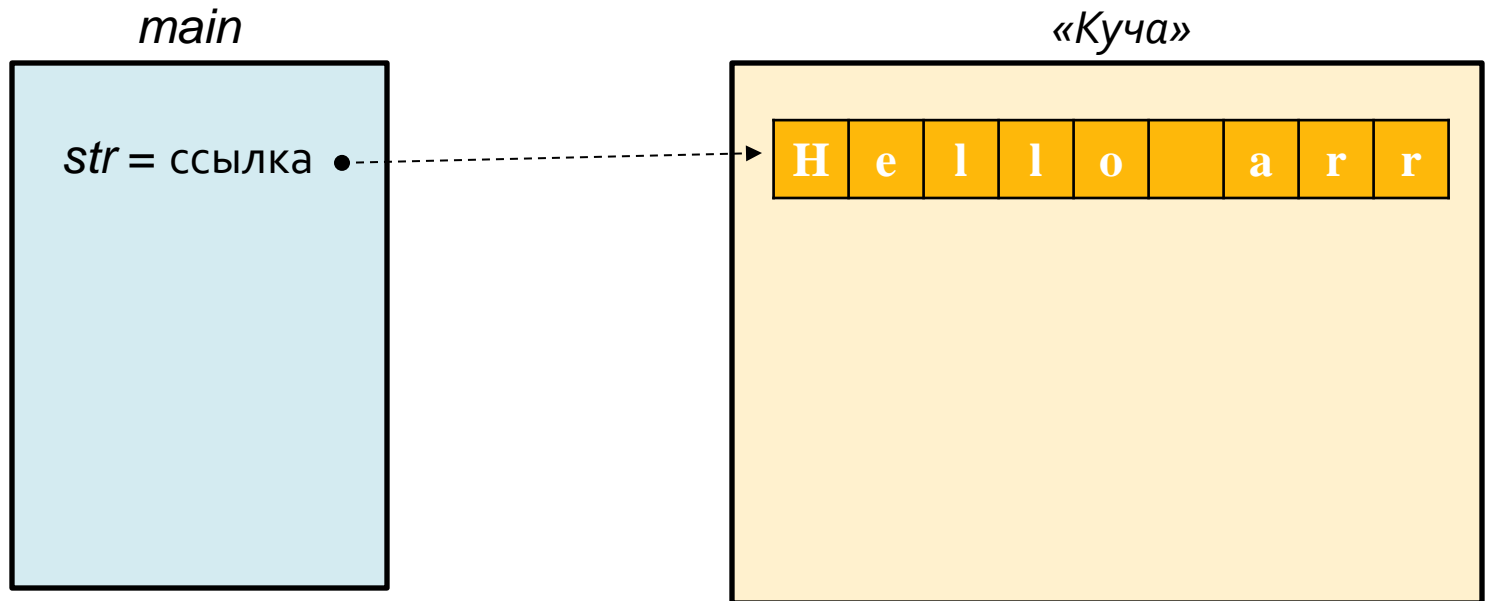
Например,

```
String s = new String();  
String s1 = new String(s);  
String s2 = new String("Student");
```

Обработка строк

Пример 3:

```
public static void main(String[] arg) {  
    char[] helloArray = {'H', 'e', 'l', 'l', 'o', ' ', 'a', 'r', 'r'};  
    String str = new String(helloArray);  
    System.out.println(str);  
}
```



Методы класса String

<i>Прототип метода</i>	<i>Назначение метода</i>
length()	Определение длины строки
concat(String s) или «+»	Соединение двух строк
equals(Object ob)	Сравнение двух строк с учетом регистра
equalsIgnoreCase(String s)	Сравнение двух строк без учета регистра
substring(int n, int m)	Извлечение из строки подстроки длиной (m-n), начиная с позиции n
substring(int n)	Извлечение из строки подстроки, начиная с позиции n
valueOf(<значение>)	Преобразование переменной базового типа к строке
toUpperCase()	Преобразование всех символов строки в верхний регистр
toLowerCase()	Преобразование всех символов строки в нижний регистр
replace(char c1, char c2)	Замена в строке всех вхождений первого символа вторым символом
trim()	Удаление всех пробелов в начале и конце строки
charAt(int pos)	Получение символа по указанной позиции pos
getChars(<параметры>)	Получение строки символов в виде массива двухбайтных значений
indexOf(int ch)	Определить позицию первого вхождения указанного символа в строке
lastIndexOf(char c)	Определить позицию последнего вхождения указанного символа в строке

Объединение строк

```
String str1 = "Learning ";
```

```
String str2 = "java!";
```

```
String str3 = str1.concat(str2);
```

```
String str4 = str1 + str2;
```

Первый способ

Второй способ

```
System.out.println(str3);
```

```
System.out.println(str4);
```

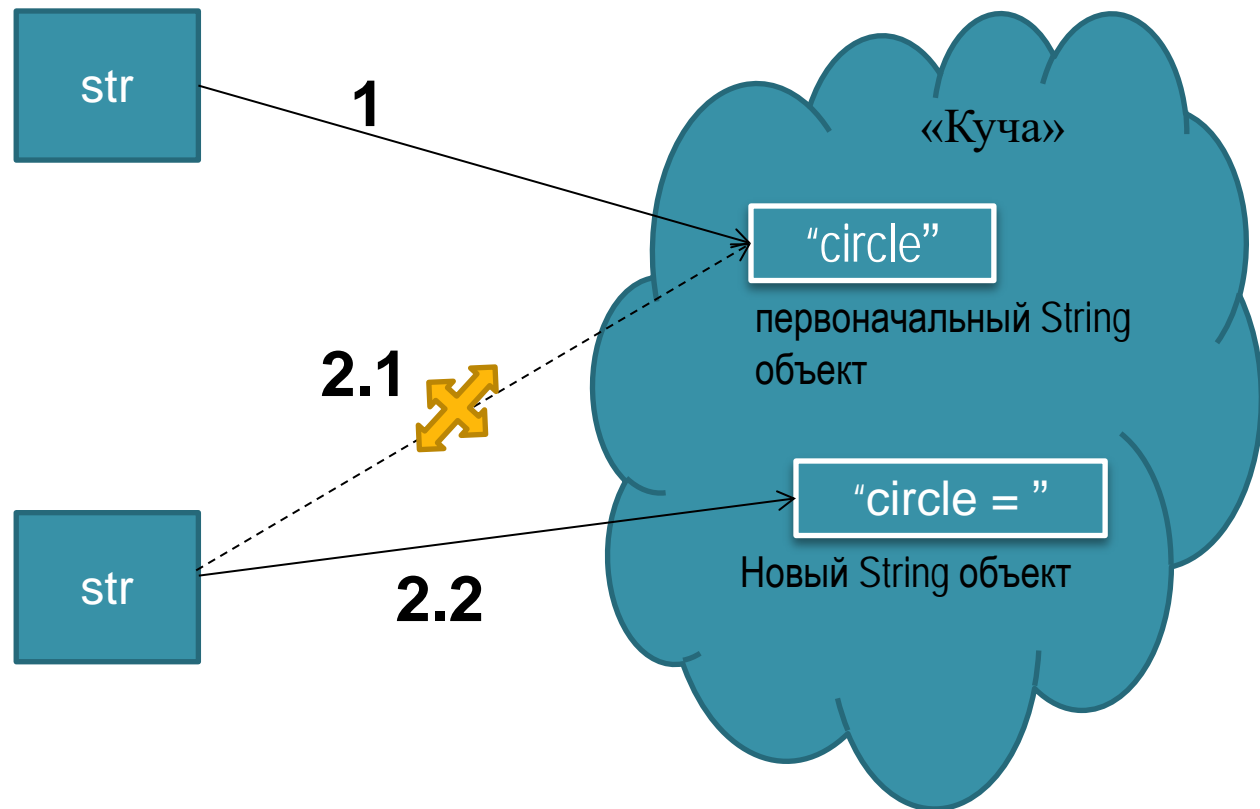
Вывод в консоли:

Learning java!

Learning java!

Неизменяемость строк

1. String str = "circle";
2. str = str + " = ";



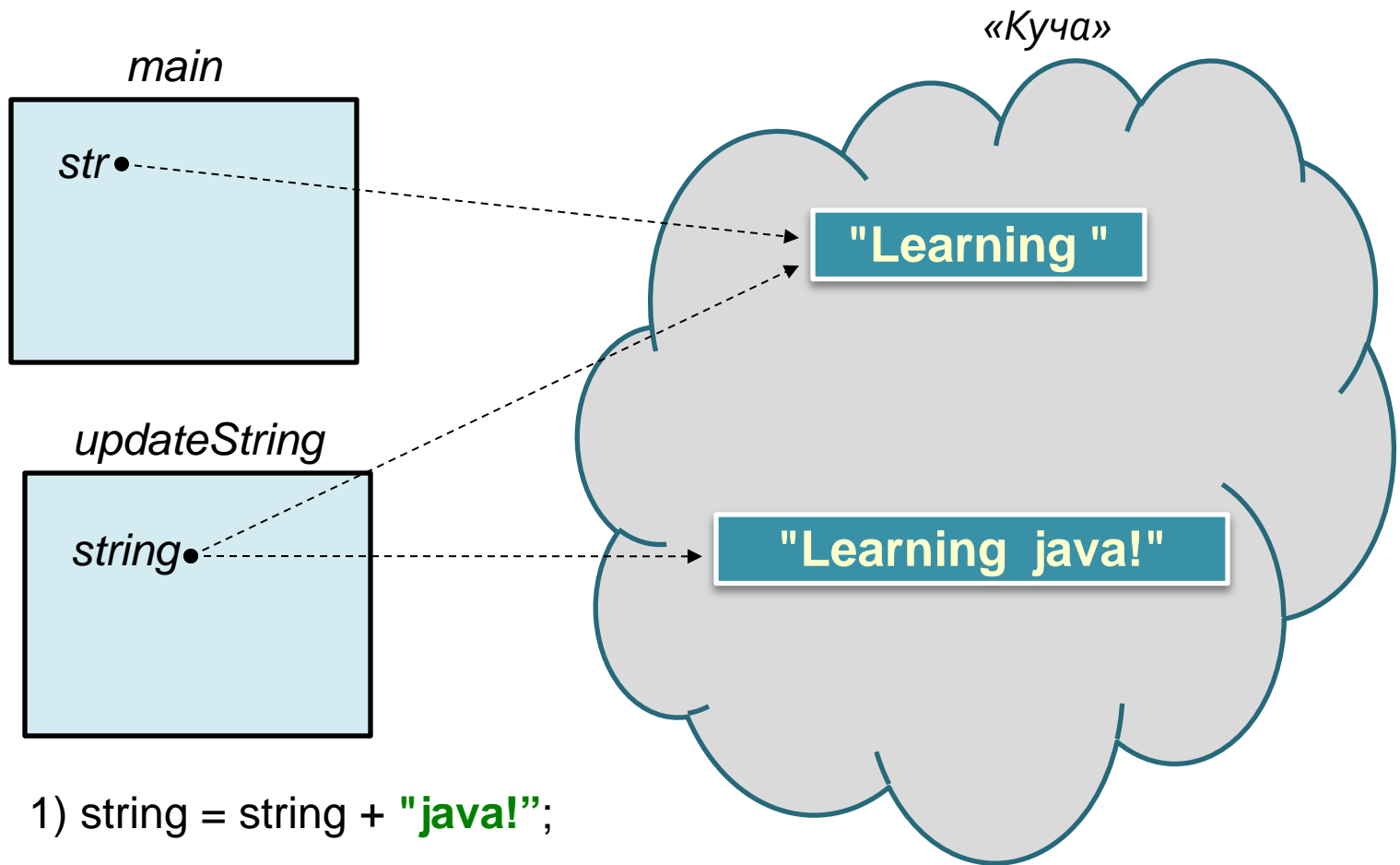
Работа с ссылками на строки типа String

Пример 4:

```
public class ChangeString {  
    public static void main(String [] args) {  
        String str = "Learning";  
        updateString(str);  
        System.out.println(str);  
    }  
    static void updateString(String string) {  
        string = string + "java!";  
    }  
}
```

Вывод в консоли:
Learning

Обработка строк



Обработка строк

Сравнение строк

Пример 5:

```
class EqualsNotEqualTo {  
    public static void main(String args[]) {  
        String s1 = "Привет";  
        String s2 = new String(s1);  
  
        System.out.println(s1 + " equals " + s2 + " -> " +  
                           s1.equals(s2));  
  
        System.out.println(s1 + " == " + s2 + " -> " + (s1 == s2));  
    }  
}
```

Вывод в консоли:

```
Привет equals Привет -> true  
Привет == Привет -> false
```

Обработка строк

Работа с индексами в строке

Пример 6:

```
String str = "Software And Hardware!";
```

```
char aChar0 = str.charAt(0);
```

```
char aChar9 = str.charAt(9);
```

```
char aCharEnd = str.charAt( str.length() - 1);
```

```
System.out.println(aChar0);
```

```
System.out.println(aChar9);
```

```
System.out.println(aCharEnd);
```

Вывод в консоли:

S

A

!

- Если *указанный индекс* не входит в границы строки, то будет ошибка типа **StringIndexOutOfBoundsException**

Получение подстроки

Пример 7:

```
String str = "Software And Hardware!";
```

```
String substr1 = str.substring(13);
```

От указанного индекса до
конца строки

```
String substr2 = str.substring(0, 8);
```

```
String substr3 = str.substring(13, 17);
```

Начиная с первого
индекса и до второго
индекса

```
System.out.println(substr1);
```

```
System.out.println(substr2);
```

```
System.out.println(substr3);
```

Вывод в консоли:

Hardware!

Software

Hard

Обработка строк

Поиск символа в строке

Пример 8:

```
String str = "Software And Hardware!";
```

```
int i1 = str.indexOf('a');
```

```
int i2 = str.lastIndexOf('a');
```

```
int i3 = str.indexOf('x');
```

```
System.out.println(i1);
```

```
System.out.println(i2);
```

```
System.out.println(i3);
```

От начала и до первого
встреченного

С конца и до первого
встреченного

Вывод в консоли:

5

18

-1


- Если указанного символа, то возвращается -1, а также можно искать от указанного индекса.

Обработка строк

Поиск подстроки в строке

Пример 9:

```
String str = "String in java is a sequence of characters java";  
int i1 = str.indexOf("java");  
int i2 = str.lastIndexOf("java");  
int i3 = str.indexOf("java", 20);  
System.out.println(i1);  
System.out.println(i2);  
System.out.println(i3);
```



Вывод в консоли:

10

44

44

Пул общих строк

- ❑ Для уменьшения использования памяти виртуальной машины и оптимизации работы с часто используемыми строками в Java существует специальный механизм хранения строковых литералов в отдельной области памяти – пул общих строк (*string common pool*).
- ❑ Если существует два или более одинаковых строковых литерала, то для них в этой области выделяется место только для одного, но ссылки на этот объект могут быть присвоены любому количеству строковых переменных.

Обработка строк

Пример 10:

```
String s1 = "Hello";  
String s2 = "Hello";  
String s3 = s1;  
String s4 = new String("Hello");  
String s5 = new String("Hello");
```

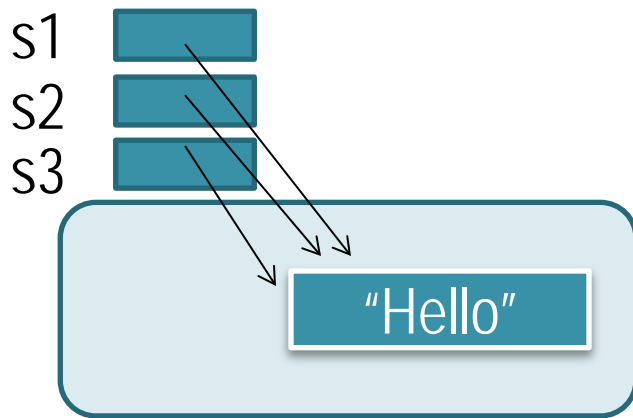
// строковый литерал

// строковый литерал

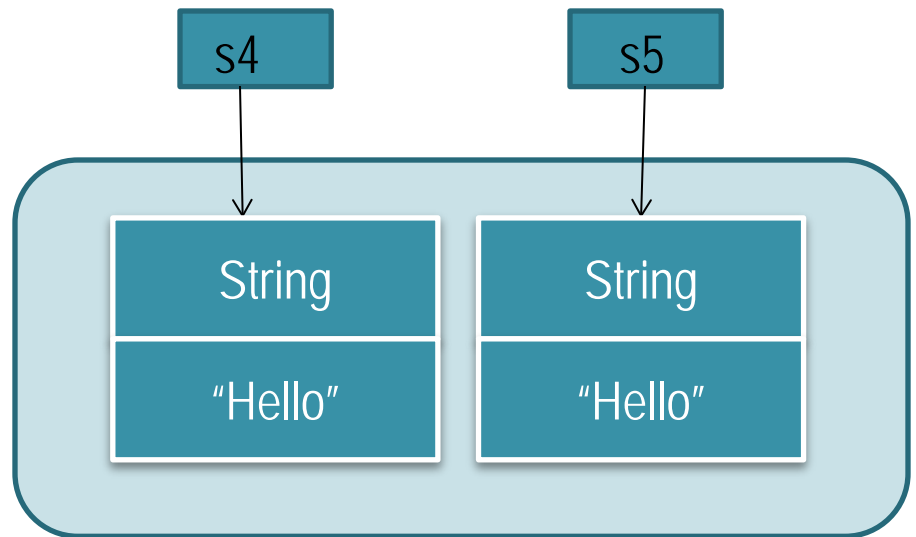
// та же ссылка

// новый объект

// новый объект



Пул строковых литералов



"Куча"

Классы **StringBuilder** и **StringBuffer**

- ❑ Потеря эффективности работы с данными типа **String** при частой их модификации:

```
String str = "S0";  
for (int i = 1; i <= 6; i++) {  
    str += "m" + i;  
}  
System.out.println(str);
```

Выход: использование класса
StringBuilder или **StringBuffer**

Объекты в «Куче»:

S0

S0m1

S0m1m2

S0m1m2m3

S0m1m2m3m4

S0m1m2m3m4m5

S0m1m2m3m4m5m6

Обработка строк

- ❑ Класс **StringBuffer/StringBuilder** представляет расширяемые и доступные для изменений последовательности символов.

Отличие: Класс **StringBuffer** является потоко-безопасным, т.к. все методы синхронизированы.

Конструкторы класса **StringBuffer**:

StringBuffer(); → пустой буфер размерностью 16

StringBuffer(int size); → буфер размерность size

StringBuffer(String obj); → размерность: длина *obj* + 16
дополнительных

<https://docs.oracle.com/javase/10/docs/api/java/lang/StringBuffer.html>

<https://docs.oracle.com/javase/10/docs/api/java/lang/StringBuilder.html>

Методы классов StringBuffer/StringBuilder

<i>Прототип метода</i>	<i>Назначение метода</i>
setLength(int n)	Установить размер буфера для объекта
capacity()	Определить размер буфера объекта
append(...)	Добавить символы, значения базовых типов, массивы и строки
insert(...)	Вставить символ, объект или строку в указанную позицию
deleteCharAt(int index)	Удалить символ по указанному индексу
delete(int start, int end)	Удалить подстроку с указанными позициями
reverse()	Перевертывание строки символов

Обработка строк

Пример 11:

```
StringBuffer sb = new StringBuffer(10);  
sb.append("Hello...");  
char c = '!';  
sb.append(c);  
sb.insert(8, " Java");  
  
sb.delete(5, 8);  
  
System.out.println(sb);
```

// добавить символ
// вставить строку в
// позицию 8
// удалить подстроку с 5
// до 8 позиции

Обработка строк

1) `StringBuilder sb = new StringBuilder(10);`

0	1	2	3	4	5	6	7	8	9

2) `sb.append("Hello...");`

0	1	2	3	4	5	6	7	8	9
H	e	l	l	o	.	.	.		

3) `sb.append('!');`

0	1	2	3	4	5	6	7	8	9
H	e	l	l	o	.	.	.	!	

4) `sb.insert(8, " Java");`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
H	e	l	l	o	.	.	.		J	a	v	a	!	

5) `sb.delete(5,8);`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
H	e	l	l	o		J	a	v	a	!				

Работа со ссылками на строки типа StringBuffer

Пример 12:

```
public class ChangeStringB {  
    public static void main(String [] args) {  
        StringBuffer str = new StringBuffer("Learning ");  
        updateString(str);  
        System.out.println(str);  
    }  
    static void updateString(StringBuffer string) {  
        string.append("java!");  
    }  
}
```

Вывод в консоли:
Learning java!