Язык программирования Java. Начальный уровень

Мультимедийный курс

автор: Васильев А.Н.

www.vasilev.kiev.ua

Киев 2017

Лекция 7. Текст

- Знакомство с классом String
- Основные способы создания объектов класса String
- Операции с текстовыми объектами
- Методы класса String
- Метод toString()
- Знакомство с классами StringBuffer и StringBuilder

Если есть на этом Плюке гравицаппа, так достанем. Не такое доставали…

из к/ф "Кин-дза-дза"

Работа с текстом

Стандартный способ реализации текста - в виде объекта класса `String`

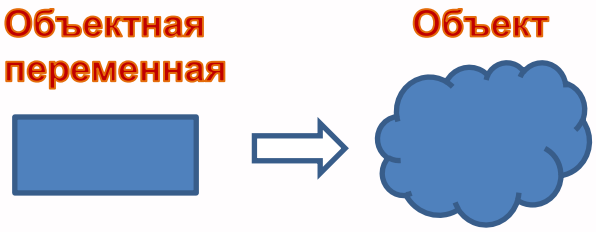
Основные способы создания:

```
String txt = "текстовый литерал";
```

```
char[] symbs={'J','a','v','a'};
String txt = new String(symbs);
```

```
String str="какой-то текст";
String txt = new String(str);
```

Реализация:




```
String txt="текст";
txt+="текст";
```

Этапы вычислений:

- Создается переменная `txt`
- Создается объект с текстом и ссылка на него записывается в переменную `txt`
- Создается новый объект с новым текстом и ссылка на него записывается в переменную `txt`

Операции с объектами класса String

Определение длины текста: используем метод `length()`

```
String str="Язык Java";
System.out.println(str.length());
```

Текстовые литералы:
Для текстовых литералов создаются объекты класса `String`

```
System.out.println("abc".length());
```

Конкатенация (объединение) текстовых строк
В Java запрещено использовать операторы с `String`-объектами, за исключением оператора “+” – выполняется объединение строк

```
String str="Сейчас "+2016+" год";
// "Сейчас 2016 год"

String txt="четыре: "+2+2;
// "четыре: 22"
```

После создания String-объект не может быть изменен!!!

Методы класса String (извлечение символов)

```
char charAt(int)
```

Получение символа по его индексу в строке

Индекс символа

Запись символов из строки в массив

```
void getChars(int, int, char[], int)
```

Индекс начала строки

Индекс конца строки+ 1

Массив-результат

Индекс в конечном массиве, начиная с которого выполняется заполнение

```
char[] toCharArray()
```

Преобразование символов из String-объекта в массив

Программа: извлечение символов

```
class GetCharsDemo{
    public static void main(String args[]){
        String s="Пример текстовой строки – язык Java";
        char  buf[]=new char[9];
        s.getChars(26,35,buf,0);
        System.out.println(buf);
        char symbol;
        symbol=s.charAt(31);
        System.out.println(symbol);
        char chars[];
        chars=s.toCharArray();
        for(int i=0;i<s.length();i++){
            System.out.print("*"+chars[i]);
        }
        System.out.println("*");
    }
}
```

Результат программы:

```
язык Java
J
*П*р*и*м*е*р* *т*е*к*с*т*о*в*о*й* *с*т*р*о*к*и* *–* *я*з*ы*к* *J*a*v*a*
```

Методы класса String (поиск элемента)

Поиск индекса первого вхождения символа или подстроки в строку

```
int indexOf(char или String)
```

Поиск индекса последнего вхождения символа или подстроки в строку

```
int lastIndexOf(char или String)
```

Вторым аргументом может указываться начальная точка (индекс) для начала поиска. В первом случае поиск выполняется от точки начала поиска к концу строки, а во втором случае - от точки начала поиска до начала строки. Если совпадений нет, возвращается значение -1

Программа: поиск элемента

```
class IndexOfDemo{
   public static void main(String args[]){
      String s="Now is the time for all good men "+
               "to come to the aid of their country.";
      System.out.println(s);
      System.out.println("1: "+s.indexOf('t'));
      System.out.println("2: "+s.lastIndexOf('t'));
      System.out.println("3: "+s.indexOf("the"));
      System.out.println("4: "+s.indexOf('t',10));
      System.out.println("5: "+s.indexOf("the",10));
   }
}
```

Результат программы:

```
Now is the time for all good men to come to the aid of their country.
1: 7
2: 65
3: 7
4: 11
5: 44
```

Методы класса String (сравнение строк)

`boolean equals(String)`

Результат равен `true` если одинаковые символы (с учетом регистра) в одинаковом порядке, иначе - `false`

`String`-объект для сравнения

`boolean equalsIgnoreCase(String)`

То же самое, что вметоде `equals()`, но без учета состояния регистра

`String`-объект для сравнения

При выполнении команды вида `str==txt` сравниваются ссылки на объекты, а не текстовые значения!!!

Программа: сравнение строк

```
class EqualsDemo{
   public static void main(String args[]){
      String s1="Hello";
      String s2=new String("Hello");
      String s3="Good-bye";
      String s4="HELLO";
      System.out.println(s1+" равно "+s2+" --> "+s1.equals(s2));
      System.out.println(s1+" равно "+s3+" --> "+s1.equals(s3));
      System.out.println(s1+" равно "+s4+" --> "+s1.equals(s4));
      System.out.println(s1+" равно "+s4+
         " --> "+s1.equalsIgnoreCase(s4));
      System.out.println(s1+" равно "+s2+" --> "+(s1==s2));
   }
}
```

Результат программы:

```
Hello равно Hello --> true
Hello равно Good-bye --> false
Hello равно HELLO --> false
Hello равно HELLO --> true
Hello равно Hello --> false
```

Методы класса String

Метод	Описание
`String substring(int,int)`	Метод возвращает подстроку. Аргументами передают начальный индекс и конечный индекс + 1 подстроки. Можна указывать только первый аргумент
`String concat(String)`	Возвращается строка, получающаяся объединением исходных строк
`String replace(char,char)`	Методом возвращается строка, которая получается заменой в исходной строке первого символа-аргумента на второй символ-аргумент
`String trim()`	Метод возвращает строку, которая получается удалением из исходной строки начальных и конечных пробелов
`String toLowerCase()`	Переводит символы в нижний регистр
`String toUpperCase()`	Переводит символы в верхний регистр

Программа: изменение строк

```
class StringReplace{
   public static void main(String args[]){
      String str="Мы программируем на C++";
      String s,s1,s2,s3,s4;
      s=str.substring(3,19);
      System.out.println(s);
      s1=str.concat(" и Java");
      System.out.println(s1);
      s2=s1.replace(' ','_');
      System.out.println(s2);
      s3=s1.toLowerCase();
      System.out.println(s3);
      s4=s1.toUpperCase();
      System.out.println(s4);
   }
}
```

Результат программы:

```
программируем на
Мы программируем на C++ и Java
Мы_программируем_на_C++_и_Java
мы программируем на c++ и java
МЫ ПРОГРАММИРУЕМ НА C++ И JAVA
```

Метод toString()

Для преобразования объектов в текст используется метод `toString()`. Метод `toString()` определен в классе `Object` (общий суперкласс Java)

- Метод `toString()` вызывается автоматически
- Метод `toString()` можно переопределить

```
class Box{
   double width, height, depth;
   Box(double w,double h,double d){
      width=w; height=h; depth=d;}
      public String toString(){
         return "Размеры: "+width+":"+height+":"+depth;}}
class ToStringDemo{
   public static void main(String args[]){
      Box obj=new Box(10,20,30);
      String s="Внимание! "+obj;
      System.out.println(s);
      System.out.println(obj);
   }
}
```

Результат:

```
Внимание! Размеры: 10.0:20.0:30.0
Размеры: 10.0:20.0:30.0
```

Домашнее задание

- Напишите программу с методом для отображения текстовой строки в обратном порядке.

- Напишите программу с методом для подсчета количества вхождений символа в текстовую строку.

- Напишите программу с методом, которым текст отображается в консоли, причем каждое слово из текста отображается в отдельной строке.

Дополнительные материалы: класс StringBuffer

Класс `StringBuffer` обеспечивает дополнительные возможности при работе с текстом.

- Объекты класса `String` – это неизменные последовательности фиксированной длины
- Объекты класса `StringBuffer` – это перезаписываемые символьные последовательности

В объекты класса `StringBuffer` можно вставлять символы и подстроки в середину и конец строки. Делается это за счет выделения дополнительной памяти

Конструктор

`StringBuffer()` - резервирует память для 16-ти дополнительных символов

`StringBuffer(int)` - явно указывается размер буфера

`StringBuffer(String)` - задается начальное значение и выделяется место еще для 16-ти символов

Методы класса StringBuffer

Метод	Описание
`int length()`	Текущая длина текстовой строки
`int capacity()`	Выделенный объем памяти (в символах) для данного объекта
`void ensureCapacity(int)`	Выделение памяти для уже созданного объекта
`void setLength(int)`	Устанавливается длина строки
`char charAt(int)`	Доступ к символу по индексу
`void setCharAt(int,char)`	Присваивание значения символу с заданным индексом
`void getChars(int,int,char[],int)`	Копирование текстовой строки в массив. Аргументы: индексы подстроки, массив и индекс в массиве для начала копирования
`StringBuffer append(String)`	Добавление строки в конец строки, из объекта которой вызывается метод
`StringBuffer insert(int,String)`	Вставка подстроки в строку. Аргументы: индекс и подстрока для вставки
`void reverse()`	Замена порядка следования символов в строке
`StringBuffer delete(int,int)`	Удаление последовательности символов в строке. Аргументы: индексы подстроки для удаления
`StringBuffer deleteCharAt(int)`	Удаление символа с указанным индексом
`StringBuffer replace(int,int,String)`	Замена одного набора символов другим. Аргументы: индексы заменяемой подстроки и текст замены

Программа: использование класса StringBuffer

```
class StringBufferDemo{
  public static void main(String args[]){
    StringBuffer str=new StringBuffer("Мы программируем на C++");
    System.out.println(str.length());
    System.out.println(str.capacity());
    str.insert(20,"Java и ");
    System.out.println(str);
    str.replace(27,30,"Pascal");
    System.out.println(str);
    str.reverse();
    System.out.println(str);
  }
}
```

Результат:

```
23
39
Мы программируем на Java и C++
Мы программируем на Java и Pascal
lacsaP и avaJ ан меуриммаргорп ыМ
```

Класс StringBuilder

Класс `StringBuilder` по функциональным возможностям аналогичен классу `StringBuffer`

Если в программе используется один поток - рекомендуется использовать `StringBuilder`

Если в программе используется несколько потоков - рекомендуется использовать `StringBuffer`