Программирование

Тема 7.2 Строки в Java

Класс String в Java

Пример 1: Способы создания строк

```
package primer;
public class Primer
{public static void main(String[] args)
  { String s = ""; //пустая строка
    // String s = new String (); //пустая строка
    String s1 = "hello"; String s2 = "hello";
    //  String s1 = new String ("hello");
     String s3 = "hel" + "lo";
     String s4 = "hel"; s4 = s4 +"lo";
     char chars[] = { 'a', 'b', 'c', 'd', 'e', 'f' };
     String s5 = new String(chars);
      System.out.println(s1); System.out.println(s2);
      System.out.println(s3); System.out.println(s4);
      System.out.println(s5);
      if (s1 == s2) System.out.println("Ссылки равны");
          else System.out.println("Ссылки не равны");
      if (s1 == s3) System.out.println("Ссылки равны");
          else System.out.println("Ссылки не равны");
      if (s1 == s4) System.out.println("Ссылки равны");
          else System.out.println("Ссылки не равны");
     System.out.println(s5.charAt(0));
  }
}
```

```
run:
hello
hello
hello
hello
abcdef
Ссылки равны
Ссылки равны
Ссылки не равны
a
```

Класс String в Java

- Строка в языке Java – это не массив символов типа char, а объект соответствующего класса.
- Для работы со строками в Java используются классы String, StringBuffer и StringBuilder пакета java.lang.
- Символы в строках хранятся в кодировке Unicode, в которой каждый символ занимает два байта. Тип каждого символа char. Каждый символ строки имеет индекс. Индекс первого символа строки равен нулю.
- Строку String можно создать с использованием оператора new. Конструкторы класса String позволяет инициализировать строки различными способами, например, пустой строкой, строковой константой, массивом символов.Также каждая строковая константа порождает объект класса String (пример 1).
- Класс String позволяет создавать строковые объекты, которые нельзя изменить. Методы класса, модифицирующие объект String, реально не изменяют его, а создают дополнительный объект String.
- Ссылку на объект класса String можно изменить так, чтобы она указывала на другой объект и тем самым на другую строку.
- Классы StringBuffer и StringBuilder допускают изменения в строке.
- Строки можно объединять, используя операцию конкатенации (+), это единственная операция, применимая для строк. Все остальные манипуляции со строками выполняются с помощью методов.
- Поскольку объект класса String неизменяем, то результатам сложения s4 = s4 + “lo” (пример 1) будет новый объект. Одинаковые строковые константы размещаются в одном пуле строк (пример 1, ссылки s1, s2 и s3 – ссылаются на одну и ту же область памяти (объект) с константой “hello”).
- Если использовать операцию сравнения (==) со строками, то она будет сравнивать ссылки на строки, а не сами строки (пример 1).
- Для чтения символа строки с указанным номером используется метод charAt(). Символы строки нумеруются с нуля.

Пример 2: Объединение и сравнение строк

```
run:
Привет мир!!!
2014/10/28
Строки не равны
Строки равны
Строки не равны
Строки равны
```

```
package primer;
public class Primer
{ public static void main(String[] args)
  {    String s1 = "Привет"; String s2 = "мир";
       String s3 = s1 + " " + s2;
       String s4 = s3.concat("!!!");
       System.out.println(s4);
       String joined = String.join("/", "2014", "10", "28" );
       System.out.println(joined);
       String s5 = "Мир";
       if (s1.equals(s2))System.out.println("Строки равны");
          else System.out.println("Строки не равны");
       if (s2.equalsIgnoreCase(s5))System.out.println("Строки равны");
          else System.out.println("Строки не равны");
       if (s1.compareTo(s2) == 0)System.out.println("Строки равны");
          else System.out.println("Строки не равны");
       if (s2.compareToIgnoreCase(s5) == 0)
           System.out.println("Строки равны");
         else System.out.println("Строки не равны");
  }
}
```

Класс String в Java

Пример 3: Поиск в строке, разделение строки

```
package primer;
public class Primer
{public static void main(String[] args)
  { String s1 = "hello world";
    char ch = 'o';
    int indexOfChar = s1.indexOf(ch); // равно 4
    System.out.println(indexOfChar);
    int indexOfChar2 = s1.lastIndexOf(ch);
    System.out.println(indexOfChar2);
    String subString = "wor";
    int indexOfSubstring = s1.indexOf(subString); // равно 6
    System.out.println(indexOfSubstring);
    String text = "И поэтому, все   так произошло.";
    String[] words = text.split("[\\s,-;.]+");
       for(String s : words)
           System.out.println(s);
  }
 }
```

```
run:
4
7
6
И
поэтому
все
так
произошло
```

Класс String в Java

- В примере 2 показано использование методов класса System.String, которые используются для объединения (конкатенации) и сравнения строк.
- Метод concat выполняет сцепление (конкатенацию) строк.
- Метод join объединяет несколько строк через символ разделитель.
- Метод compareTo сравнивает две строки и возвращают число. Если первая строка по алфавиту стоит выше второй, то возвращается число больше нуля. В противном случае возвращается число меньше нуля. И третий случай - если строки равны, то возвращается число 0.
- Метод compareTo предназначен в основном для использования при сортировке строк.
- Для сравнения строк удобно использовать метод equals. Метод equals возвращает true, если строки равны и false – в противном случае. Метод equalsIgnoreCase сравнивает строки без учета регистра.
- С помощью метода indexOf мы можем определить индекс первого вхождения отдельного символа или подстроки в строке (пример 3).
- Подобным образом действует метод lastIndexOf, только находит индекс последнего вхождения символа или подстроки в строку.
- С помощью метода Split можно разделить строку на массив подстрок. В качестве параметра функция Split принимает регулярное выражение, которое задает разделители подстрок в строке (пример 3).

Пример 4: Обрезка начальных и конечных пробелов строки, извлечение подстроки

```
package primer;
public class Primer
{public static void main(String[] args)
  { String text = " hello world ";
    text = text.trim();
    System.out.println(text);
    text = "Хороший день";
    // обрезаем начиная с третьего символа
    text = text.substring(2);
     System.out.println(text);
    // обрезаем сначала до последних двух символов
     text = text.substring(0, text.length() - 2);
     System.out.println(text);
  }
}
```

```
run:
hello world
роший день
роший де
```

Пример 5: Замена в строке

```
package primer;
public class Primer
{ public static void main(String[] args)
  {  String text2 = "плохой день. плохой день.";
    System.out.println(text2);
    text2 = text2.replace("плохой", "хороший ");
    System.out.println(text2);
    text2 = text2.replaceFirst("[^.]+", "???");
    System.out.println(text2);
    text2 = text2.replaceAll("[.]", "!!!");
    System.out.println(text2);
    text2 = text2.replace('?', '.');
    System.out.println(text2);
    text2 = text2.replace("о", "");
    System.out.println(text2);
  }
}
```

```
run:
плохой день. плохой день.
хороший  день. хороший  день.
???. хороший  день.
???!!! хороший  день!!!
...!!! хороший  день!!!
...!!! хрший  день!!!
```

Пример 6.1: Форматирование строк

```
package primer;
 public class Primer
 { public static void main(String []args)
    { String stringVar = "abc";
      float floatVar = 2.12354f;
      int intVar = 25;
      String str1,str2;
      System.out.format("floatVar = %f, intVar =  %d, " +
        "stringVar = %s\n", floatVar, intVar, stringVar);
      str1 = String.format("floatVar = %-10.3f, intVar =  %d, " +
        "stringVar = %s\n", floatVar, intVar, stringVar);
      System.out.println(str1);
     }
}
```

```
run:
floatVar = 2,123540, intVar =  25, stringVar = abc
floatVar = 2,124     , intVar =  25, stringVar = abc
```

Класс String в Java

- Пакет java.io содержит класс PrintStream, который содержит методы printf и format.
- Объект System.out — это объект PrintStream, поэтому вы можете вызывать методы PrintStream используя System.out. Например: System.out.format(.....);
- Синтаксис методов одинаков: public PrintStream format(String format, Object... args)
- где format — это строка которая определяет шаблон, согласно которому будет происходить форматирование, args — это список переменных, для печати по заданному шаблону.
- Строка format содержит обычный текст и специальные форматирующие символы. Эти символы начинаются со знака процента (%) и заканчиваются символом, который определяет тип переменной для форматирования.
- Следующая таблица содержит некоторые форматирующие символы и флаги:

d		Десятичное целое.
f		Float.
n		Символ новой строки в зависимости от платформы, на которой запущена программа. Вместо \n лучше использовать %n.
tB		Дата и время — полное название месяца в зависимости от языка.
td, te		Дата и время — 2 цифры дня месяца. td — с ведущими нулями, te — без.
ty, tY		Дата и время — ty = год из 2х цифр, tY = год из 4х цифр.

Класс String в Java

tl		Дата и время — часы в 12 часовом формате.
tM		Дата и время — минуты из 2х цифр с ведущими нулями.
tp		Дата и время — am/pm в зависимости от языка(в нижнем регистре).
tm		Дата и время — месяц — 2 цифры с ведущими нулями.
tD		Дата и время — дата в формате %tm%td%ty
	08	Восемь символов с ведущими нулями при необходимости.
	+	Включить знак (положительный или отрицательный).
	-	По левому краю
	.3	Три символа после запятой
	10.3	Десять символов до запятой и три — после.

Пример 6.2: Форматирование строк

```
package primer;
import java.util.Calendar;
 public class Primer
 { public static void main(String []args)
     { String str2;
       Calendar c = Calendar.getInstance();
       str2 = String.format("%te %tB, %tY ", c, c, c) +
              String.format("%tl:%tM %tp%n", c, c, c);
       System.out.print(str2);
       String str3 = String.format("%tD%n", c);
       System.out.println(str3);
      }
}
```

```
run:
14 мая, 2015 9:17 pm
05/14/15
```

Пример 6.3: Использование класса Formatter

```
package primer;
import java.util.Formatter;
public class Primer
 { public static void main(String args[])
   { double data[] = { 12.3, 45.5764, -0.09, -18.0, 1232.01 };
     Formatter fmt = new Formatter();
     // Создать таблицу
     fmt.format("%12s %12s\n", "Значение", "Куб. корень");
     for(double v : data)
        fmt.format("%12.4f %12.4f\n", v, Math.cbrt(v));
     // Отобразить форматированные данные.
     System.out.println(fmt);
   }
}
```

```
run:
    Значение  Куб. корень
     12,3000       2,3084
     45,5764       3,5720
     -0,0900      -0,4481
    -18,0000      -2,6207
   1232,0100      10,7202
```

Класс StringBuilder в Java

Пример 7: Создание строк StringBuilder

```
package primer;
public class Primer
{static void Print(StringBuilder sb)
     {  System.out.println("Длина строки: " + sb.length());
        System.out.println("Емкость строки: " + sb.capacity());
     }
  public static void main(String[] args)
  {    StringBuilder sb = new StringBuilder();
       sb.append("Название: "); Print(sb);
       sb.append(" Руководство по C#"); Print(sb);
       System.out.println();
       StringBuilder sb1 = new StringBuilder(40);
       sb1.append("Название: "); Print(sb1);
       sb1.append(" Руководство по C++"); Print(sb1);
       System.out.println();
       StringBuilder sb2 = new StringBuilder("Название: ");
       Print(sb2); sb2.append(" Руководство"); Print(sb2);
       sb2.append(" по C++"); Print(sb2);
       sb2.append(" и C#"); Print(sb2);
       System.out.println();
       StringBuilder sb3 = new StringBuilder("Название: Руководство");
       Print(sb3); sb3.append(" по Java"); Print(sb3);
       System.out.println();
       String str = "Руководство по программированию";
       StringBuilder sb4 = new StringBuilder(str); Print(sb4);
  }
}
```

```
run:
Длина строки: 10
Емкость строки: 16
Длина строки: 28
Емкость строки: 34

Длина строки: 10
Емкость строки: 40
Длина строки: 29
Емкость строки: 40

Длина строки: 10
Емкость строки: 26
Длина строки: 22
Емкость строки: 26
Длина строки: 29
Емкость строки: 54
Длина строки: 34
Емкость строки: 54

Длина строки: 21
Емкость строки: 37
Длина строки: 29
Емкость строки: 37

Длина строки: 31
Емкость строки: 47
```

Пример 8: Использование методов класса StringBuilder

```
package primer;
public class Primer
{ public static void main(String[] args)
  {    StringBuilder sb = new StringBuilder(30);
       sb.append("Привет мир!");
       sb.insert(7, "компьютерный ");
       System.out.println(sb + " " + sb.length());
       sb.replace(20,23, "world");
       System.out.println(sb + " " + sb.length());
       sb.delete(7, 19);
       System.out.println(sb + " " + sb.length());
       sb.deleteCharAt(6);
       String s = sb.toString();
       System.out.println(s+ " " + sb.length());
  }
}
```

```
run:
Привет компьютерный мир! 24
Привет компьютерный world! 26
Привет  world! 14
Привет world! 13
```

Класс StringBuilder в Java

- Объекты типа StringBuilder отличаются от объектов типа String тем, что они могут быть изменены. По сути они представляют из себя массивы переменной длины, содержащие последовательность символов. Длина и содержание последовательности могут быть изменены вызовом соответствующих методов.
- StringBuilder необходимо использовать в случаях, когда это поможет упростить код или если это необходимо для лучшей производительности. Если вам необходимо объединить большое количество строк, использование StringBuilder будет более эффективным.
- Класс StringBuilder как и класс String, имеет метод length(), который возвращает длину последовательности символов объекта.
- В отличие от строк String, в StringBuilder, помимо длины есть capacity — количество символов под которое выделена память. Емкость (capacity), возвращаемая методом capacity(), всегда больше или равна длине строки (обычно больше) и автоматически будет увеличина при необходимости.
- Конструкторы StringBuilder:

`StringBuilder()`	Создает пустой stringbuilder, с емкостью 16 символов.
`StringBuilder(CharSequence cs)`	Создает stringbuilder, содержащий заданную последовательность символов плюс дополнительно 16 пустых символов.
`StringBuilder(int initCapacity)`	Создает stringbuilder определенной емкости.
`StringBuilder(String s)`	Создет stringbuilder с заданной строкой.

Класс StringBuilder в Java

- Класс StringBuilder содержит несколько методов относящихся к длине и емкости, которых нет в классе String:

`void setLength(int newLength)`	Задает длину последовательности символов. Если newLength меньше length(), последние символы будут отсечены. Если newLength больше length(), будут добавлены null-символы в конец последовательности.
`void ensureCapacity(int minCapacity)`	Гарантирует, что емкость будет не меньше указанного минимума.

- Некоторые методы (например, append(), insert(), или setLength()) могут увеличить длину последовательности символов в stringbuilder'е , так что длина может оказаться больше текущей емкости. Когда это происходит — емкость автоматически увеличивается.
- Основные функции StringBuilder, которых нет в String: append() и insert(), которые перегружены для приема разного типа данных. Каждый конвертирует аргумент в строку и добавляет её в строку stringbuilder'а. Метод append всегда добавляет символы в конец существующей последовательности, тогда как метод insert добавляет символы в указанное место.

Класс StringBuilder в Java

- Некоторые методы класса StringBuilder:

```
StringBuilder append(boolean b)
StringBuilder append(char c)
StringBuilder append(char[] str)
StringBuilder append(char[] str, int offset, int len)
StringBuilder append(double d)
StringBuilder append(float f)
StringBuilder append(int i)
StringBuilder append(long lng)
StringBuilder append(Object obj)
StringBuilder append(String s)
```

Добавляет аргумент в конец строки stringbuilder'а. Предварительно данные конвертируются в строку.

```
StringBuilder insert(int offset, boolean b)
StringBuilder insert(int offset, char c)
StringBuilder insert(int offset, char[] str)
StringBuilder insert(int index, char[] str, int
offset, int len)
StringBuilder insert(int offset, double d)
StringBuilder insert(int offset, float f)
StringBuilder insert(int offset, int i)
StringBuilder insert(int offset, long lng)
StringBuilder insert(int offset, Object obj)
StringBuilder insert(int offset, String s)
```

Вставляет второй аргумент в stringbuilder. Первый аргумент задает позицию символа перед которым будет вставлены данные. Перед вставкой данные конвертируются в строку.

Класс StringBuilder в Java

`StringBuilder delete(int start, int end)` `StringBuilder deleteCharAt(int index)`	Первый метод удаляет подпоследовательность начиная с from до end-1 (включительно) в последовательности символов StringBuilder'а. Второй – удаляет символ, индекс которого равен index.
`StringBuilder replace(int start, int end, String s)` `void setCharAt(int index, char c)`	Заменяет заданные символы в строке.
`StringBuilder reverse()`	Переворачивает строку в stringbuilder'е.
`String toString()`	Возвращает строку, которая хранится в билдере.

- Вы можете использовать любой метод класса String на объекте класса StringBuilder предварительно сконвертировав его в строку методом toString(). Затем из строки можно опять сделать stringbuilder, используя конструктор: StringBuilder(String str).

Пример 9: Извлечение всех больших букв из строки

```
package primer;
public class Primer
{ public static String extractCapitals(String s)
   { StringBuilder result = new StringBuilder();
     for (int i = 0; i < s.length(); i++) {
       char ch = s.charAt(i);
         if (Character.isUpperCase(ch)) {
           result.append(ch);
    }
 }
 return result.toString();
}
 public static void main(String[] args)
  {   String str1 = "Abcd aBCd";
      String str2 = extractCapitals(str1);
      System.out.println(str2);
   }
 }
```

```
run:
ABC
```

Контрольные вопросы

1. Строки класса String в Java: понятие, способы создания, допустимые операции, основные методы класса String. Примеры.
2. Строки класса StringBuider в Java: понятие, способы создания, методы класса StringBuider. Примеры.