# Строки

- Класс String инкапсулирует действия со строками. Объект типа String – строка, состоящая из произвольного числа символов, от 0 до $2*10^9$.
- Литерные константы типа String представляют собой последовательности символов, заключённые в двойные кавычки.
- В классе Object имеется метод toString(), обеспечивающий строковое представление любого объекта.
- Строки типа String являются неизменяемыми объектами – при каждом изменении содержимого строки создаётся новый объект-строка.
- Для того, чтобы сделать работу с многочисленными присваиваниями более эффективной, используются классы StringBuffer и StringBuilder.

# Методы класса

- String.valueOf(параметр)
- String.valueOf(charArray, index1,count)

# Методы объектов

- s1.charAt(i)
- s1.endsWith(subS)
- s1.equals(subS);
  s1.equalsIgnoreCase(subS)
- s1.getBytes(); s1.getBytes(charset)
- s1.indexOf(subS); s1.indexOf(subS,i)
- s1. lastIndexOf (subS);
  s1. lastIndexOf (subS,i)

# Методы объектов

- s1.length()
- s1.replaceFirst(oldSubS,newSubS)
- s1.replaceAll(oldSubS,newSubS)
- s1.split(separator); s1.split(separator, i)
- s1.startsWith(subS); s1.startsWith(subs, index1)
- s1.substring(index1); s1.substring(index1,index2)
- s1.toCharArray()
- s1.toLowerCase()
- s1.toUpperCase()
- s1.trim()

# Особенности строковых выражений

String s=1+2+3;

даст значение s ”6”.

String s=”Сумма=”+1+2+3;

даст s ”Сумма =123”.

String s=1+2+” не равно ”+1+2;

даст s ”3 не равно 12”.

# Классы-обертки и строки

- int Integer.parseInt(*строка*)
- Integer Integer.valueOf(*строка*)
- String Integer.toBinaryString(*число*)
- String Integer.toHexString(*число*)
- Integer Integer.decode(*строка*)

# Класс Character

- Character.isDigit(*символ*) –является ли символ цифрой.
- Character.isLetter(*символ*) –является ли символ буквой.
- Character.isLetterOrDigit(*символ*) –является ли буквой или цифрой.
- Character.isLowerCase(*символ*) –является ли символом в нижнем регистре.
- Character.isUpperCase(*символ*) –является ли символом в верхнем регистре.
- Character.isWhiteSpace(*символ*) –является ли “пробелом в широком смысле” – пробелом, символом табуляции и т.д.

# Классы StringBuffer и StringBuilder

```
StringBuffer sb=new StringBuffer();
sb.append("типа StringBuffer или String");
sb.insert(0," типа StringBuffer или String ");
System.out.println(sb);
```

- Буферизуемые и обычные строки можно сравнивать на совпадение содержания:

```
String s=”...”;
if (s.contentEquals(sb))
... ;
```

# Сравнение со String

```
String s = "a";
   for(int i = 0; i < 100; i++)
   {
         s+='a';
   }
```

```
StringBuilder s = new StringBuilder("a");
   for(int i = 0; i < 100;i++)   {
         s.append('a');
   }
```

# StringBuilder

s.deleteCharAt(i);*//удаляет символ в позиции i*

s.delete(i, j);*//удаляет подстроку с i - го по j - ый символ*

s.insert(i,s1);*//вставляет на i - ое место объект s1*

*как вернуться к String*

String ans = s.toString();

# StringBuilder

StringBuilder был добавлен в Java 1.5 и он во всем идентичен классу StringBuffer, за исключением того, что он не синхронизирован, что делает его более эффективным, но небезопасным.

Для класса **StringBuffer** не переопределены методы **equals()** и **hashCode()**, т.е. сравнить содержимое двух объектов невозможно, к тому же хэш-коды всех объектов этого типа вычисляются так же, как и для класса **Object**.

# StringTokenizer

java.util.StringTokenizer;

```
String s="Строка, для разделения – по словам";
StringTokenizer st = new StringTokenizer(s, "  \t\n\r,-");

while (st.hasMoreTokens()){
    System.out.println(st.nextToken());
}
```