Строки

Строка представляет собой последовательность символов. Для работы со строками в Java определен класс String, который предоставляет ряд методов для манипуляции строками. Физически объект String представляет собой ссылку на область в памяти, в которой размещены символы. В языке Java массив символов char[] и класс String являются различными типами – их значения могут быть легко конвертированы друг в друга с помощью специальных методов, но они все же не идентичны.

Рассмотрим основные характеристики и особенности класса String:

- класс String входит в пакет java.lang, поэтому его не нужно импортировать;
- класс String в Java – это final класс, который не может иметь потомков;
- класс String – неизменяемый класс: его объекты не могут быть изменены после создания. Любые операции над объектом класса String, результатом которых должен быть объект класса String, приведут к созданию нового объекта;
- благодаря своей неизменности объекты класса String являются потокобезопасными и могут быть использованы в многопоточной среде;
- в Java реализован механизм конкатенации (соединения) строк при помощи оператора +, который позволяет складывать друг с другом и строковые переменные, и строковые литералы;
- любой объект в Java может быть преобразован в строку через метод toString(), унаследованный всеми Java-классами от базового класса Object.

Перечислим методы создания строк и управления ими:

Тип	Метод	Описание
char	charAt(int index)	Возвращает char со значением, равным символу с указанным индексом.
int	compareTo(String str)	Сравнивает строку с указанной строкой с учётом регистра символов.
int	compareToIgnoreCase(String str)	Сравнивает строку с указанной строкой без учёта регистра символов.
String	concat(String str)	Присоединяет строку str к концу строки.
boolean	contains(CharSequence s)	Возвращает true, если строка содержит указанную последовательность символов.
boolean	contentEquals(CharSequence cs)	Сравнивает строку с указанным значением CharSequence.
boolean	contentEquals(StringBuffer sb)	Сравнивает строку с указанным значением StringBuffer .

boolean	endsWith(String str)	Проверяет, заканчивается ли строка указанной подстрокой.
boolean	equals(Object anObject)	Сравнивает строки по значению с учетом регистра символов.
boolean	equalsIgnoreCase(String str)	Сравнивает строки по значению без учета регистра символов.
String	format(String format, Object ... args)	Возвращает отформатированную строку, используя специальный формат строки и аргументы.
byte[]	getBytes()	Возвращает представление строки в виде массива байт.
int	indexOf(int ch)	Возвращает индекс первого вхождения символа в строку.
int	indexOf(int ch, int fromIndex)	Возвращает индекс первого вхождения символа в строку, начиная с указанной позиции.
int	indexOf(String str)	Возвращает индекс первого символа подстроки в строке с её начала.
int	indexOf(String str, int index)	Возвращает индекс первого символа подстроки в строке с указанной позиции.
boolean	isEmpty()	Проверяет, пустая ли строка.
int	lastIndexOf(int ch)	Поиск последнего вхождения символа в строку.
int	lastIndexOf(int ch, int fromIndex)	Поиск последнего вхождения символа в строку. Осуществляется назад, от указанной позиции до начала строки.
int	lastIndexOf(String str)	Поиск последнего вхождения строки.
int	lastIndexOf(String str, int index)	Поиск последнего вхождения строки. Осуществляется назад, от указанной позиции до начала строки.
int	length()	Возвращает количество символов в строке.
boolean	matches(String regex)	Проверяет, удовлетворяет ли строка указанному регулярному выражению.
String	replace(char oldChar, char newChar)	Заменяет в строке один символ на другой.
String	replace(CharSequence target, CharSequence replacement)	Заменяет одну подстроку другой.
String	replaceAll(String regex, String replacement)	Заменяет каждую подстроку данной строки, соответствующую данному регулярному выражению, на заданную строку.

String	replaceFirst(String regex, String replacement)	Заменяет первую подстроку данной строки, которая соответствует данному регулярному выражению на заданную строку.
boolean	startsWith(String str)	Проверяет, начинается ли строка с указанной строки.
boolean	startsWith(String str, int offset)	Проверяет, начинается ли строка в указанной позиции с указанной строки.
String	substring(int index)	Возвращает подстроку, начиная с указанной позиции (включительно) до конца строки.
String	substring(int beginIndex, int endIndex)	Возвращает подстроку, начиная с beginIndex (включительно) до endIndex (не включительно).
char[]	toCharArray()	Преобразует строку в массив символов.
String	toLowerCase()	Конвертирует все символы строки в нижний регистр.
String	toString()	Возвращает сам объект, который уже является строкой.
String	toUpperCase()	Конвертирует все символы строки в верхний регистр.
String	trim()	Удаляет пробелы в начале и конце строки.
String	valueOf(boolean b)	Возвращает значение переменной типа boolean в виде строки.
String	valueOf(char c)	Возвращает значение переменной типа char в виде строки.
String	valueOf(char[] data)	Возвращает значение массива символов в виде строки.
String	valueOf(char[] data, int offset, int count)	Возвращает значение массива символов в виде строки.
String	valueOf(double d)	Возвращает значение переменной типа double в виде строки.
String	valueOf(float f)	Возвращает значение переменной типа float в виде строки.
String	valueOf(int i)	Возвращает значение переменной типа int в виде строки.
String	valueOf(long l)	Возвращает значение переменной типа long в виде строки.
String	valueOf(Object obj)	Возвращает строковое представление объекта.

Объекты String являются неизменяемыми, поэтому все операции, которые изменяют строки, фактически приводят к созданию новой строки, что сказывается на производительности приложения. Для решения этой проблемы, чтобы работа со строками проходила с меньшими издержками, в Java были добавлены классы StringBuffer и StringBuilder. По сути, они представляют собой расширяемую строку, которую можно изменять без ущерба для производительности.

Эти классы практически идентичны, они имеют одинаковые конструкторы и одни и те же методы, которые одинаково используются. Единственное их различие состоит в том, что класс StringBuffer синхронизированный и потокобезопасный. Таким образом, класс StringBuffer удобнее использовать в многопоточных приложениях, где объект данного класса может изменяться в различных потоках. Если же речь о многопоточных приложениях не идет, то лучше использовать класс StringBuilder, который не является потокобезопасным, но при этом работает быстрее, чем StringBuffer в однопоточных приложениях.

Перечислим методы данных классов на примере StringBuffer:

Тип	**Метод**	**Описание**
StringBuffer	append(String str)	Обновляет текущее значение StringBuffer, добавляя к нему заданную строку.
int	capacity()	Возвращает текущую вместимость StringBuffer.
char	charAt(int index)	Возвращает символ с указанным индексом из текущего значения StringBuffer.
void	delete(int beginIndex, int endIndex)	Удаляет указанные символы из текущего значения StringBuffer.
void	ensureCapacity(int minimumCapacity)	Изменяет вместимость StringBuffer, гарантируя её равенство указанному минимуму.
void	getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)	Указанные символы копируются из текущего значения StringBuffer в указанный символьный массив, начиная с заданной позиции.
int	indexOf(String str)	Возвращает индекс первого вхождения указанной подстроки в строке, представленной текущим значением StringBuffer.
int	indexOf(String str, int fromIndex)	Возвращает индекс первого вхождения указанной подстроки в строке, представленной текущим значением StringBuffer, начиная с указанного индекса.

void	insert(int offset, int i)	Вставляет строку в текущее значение StringBuffer на заданную позицию с указанным смещением.
int	lastIndexOf(String str)	Возвращает индекс последнего вхождения указанной подстроки в строке, представленной текущим значением StringBuffer.
int	lastIndexOf(String str, int fromIndex)	Возвращает индекс последнего вхождения указанной подстроки в строку, представленную текущим значением StringBuffer, начиная с указанного индекса.
int	length()	Возвращает текущую длину StringBuffer (количество символов).
void	replace(int beginIndex, int endIndex, String str)	Заменяет символы в заданной подстроке, представленной текущим значением StringBuffer, символами указанной строки.
void	reverse()	Изменяет порядок символов, содержащихся в StringBuffer, на обратный.
void	setCharAt(int index, char ch)	Устанавливает заданное значение для символа с указанным индексом, содержащегося в текущий момент в StringBuffer.
void	setLength(int newLength)	Устанавливает длину StringBuffer.
CharSequence	subSequence(int start, int end)	Возвращает последовательность символов, содержащихся в StringBuffer, которая ограничена указанными индексами.
String	substring(int start)	Возвращает подстроку, состоящую из символов, содержащихся в текущий момент в StringBuffer, которая начинается с указанного индекса.
String	substring(int start, int end)	Возвращает подстроку, состоящую из последовательности символов, содержащихся в текущий момент в StringBuffer, которая ограничена указанными индексами.
String	toString()	Возвращает строку, состоящую из последовательности символов, содержащихся в текущий момент в StringBuffer.