# Робота з рядками.

Для зберігання та обробки рядків у Java визначено два класи:

**String** – для незмінних рядків;
**StringBuffer** – для рядків, які можуть змінюватись.

Обидва класи розширюють клас **Object**. Вони знаходяться в пакеті **java.lang**, тому для їхнього підключення оператор **import** не потрібен.

**Рядкові літерали в Java укладаються в подвійні апострофи.**

**"abc"** задає рядковий літерал ***abc***.

Якщо всередині рядкового літерала необхідно задати символ апострофа, він задається символами \"

**"it\"s"** задає рядковий літерал ***it"s***.

# Ініціалізація об'єкта класу String

**Може виконуватися:**

- за допомогою оператора привласнення змінної класу **String** значення рядкової змінної або рядкового літералу

```
String str = "Строка 1";
```

- при створенні об'єкта класу **String** за допомогою оператора **new** з використанням одного з наступних конструкторів:

- **String()** – створюється порожній рядок;
- **String (String original)** – вміст рядка **original** копіюється в інший рядок;
- **String (StringBuffer buffer)** – вміст рядка buffer класу **StringBuffer** перетворюється на рядок класу **String**;
- **String(byte[] bytes)** – рядок створюється з байтового масиву **bytes** з використанням кодування на даному комп'ютері за замовчуванням;

• **String(byte[] bytes, int offset, int length)** – рядок створюється з частини масиву байт, що починається з індексу **offset** і містить **length** байт;

• **String (char [] value)** – рядок створюється з масиву **value** символів **Unicode**;

• **String(char[] value, int offset, int length)** — рядок створюється з частини масиву символів **Unicode**, що починається з індексу **offset** і містить **length** символів.

```
String str = " abc ";
String str1 = new String(str);              // Рядок str1 прийме значення"abc"
```

```
char[] charArray =
{"0", "1", "2", "3", "4"};
String str2 = new String(charArray);        // Рядок str2 прийме значення " 01234 "
String str3 = new String(charArray,1,3);    // Рядок str2 прийме значення "123"
```

Довжина рядка може бути визначена за допомогою методу **public int length()**

Для рядків можна використовувати операцію **зчеплення (конкатенація)** двох або більше рядків – **"+".**

```
int strLength =«Рядок 1".length();       // Значення strLength буде равно 8.
```

```
String S = "Перший" + « рядок";       // Рядок S получить значення: «Перший рядок"
```

**Операцію конкатенації використовують при перенесенні довгого рядка на інший рядок.**

Рядки класу **String** можна змінювати, але при кожній зміні довжини рядка створюється новий екземпляр рядка

Клас **StringBuffer** схожий на клас **String**, але рядки, створені за допомогою цього класу, можна модифікувати.

При зміні рядка класу **StringBuffer** програма не створює новий рядковий об'єкт, а працює безпосередньо з вихідним рядком, всі методи оперують безпосередньо з буфером, що містить рядок.

Клас **StringBuffer** зазвичай використовується, коли рядок часто доводиться модифікувати зі зміною його довжини.

**Розміщення рядків в об'єкті StringBuffer**:

- для об'єкта **StringBuffer** задається **розмір** чи **ємність** (*capacity*) *буферної* пам'яті для рядка;
- рядок символів в об'єкті **StringBuffer**, характеризується також своєю ***довжиною***, яка може бути меншою або дорівнює ємності буфера;
- якщо ***довжина рядка менше ємності буфера***, то довжина рядка, що залишилася, заповнюється символом **Unicode** "\u0000";
- якщо в результаті модифікації рядка її ***довжина стане більшою за ємність буфера***, ємність буфера автоматично збільшується.

**В класі StringBuffer є три конструктори:**

```
StringBuffer();
StringBuffer(int length);
StringBuffer(String str).
```

***Перший конструктор*** створює порожній об'єкт **StringBuffer** з ***ємністю*** буферної пам'яті 16 символів.

***Другий конструктор*** задає буфер із ***ємністю*** **length** для зберігання рядка.

***Третій конструктор*** створює об'єкт **StringBuffer** з об'єкта **String** з ємністю буфера, що дорівнює довжині рядка класу **String**.

Довжину рядка в об'єкті **StringBuffer** можна так само, як і для рядка класу **String**, отримати за допомогою методу

```
public int length()
```

Поточну ємність буферної області можна отримати за допомогою методу

```
public int capacity()
```

Ємність буферної пам'яті можна також встановити за допомогою методу

```
public void ensureCapacity(int minimumCapacity)
```

**Приклад:**

```
StringBuffer str = new StringBuffer("String buffer");
str.ensureCapacity(512);
```

Довжина буферної пам'яті для рядка **str** буде встановлена рівною 512 байтам.

Довжина рядка StringBuffer встановлюється за допомогою методу
**public void setLength(int newLength)**

```
StringBuffer str = new StringBuffer("String buffer");
str.setLength(40) ;
```

Якщо ***нова довжина більша за стару***, збільшуються довжини рядка та буфера, а нові символи заповнюються нулями.

Якщо ***нова довжина менша за стару***, символи в кінці рядка відкидаються, а розмір буфера не змінюється.

Для перетворення рядка **StringBuffer** на рядок **String** використовується метод**public String toString()**

# Порівняння рядків

Оскільки Java рядки є об'єктами, для порівняння рядків можна використовувати оператор"= ="

**Використання оператора "==" для порівняння рядків може призвести до невірного результату, якщо порівнювані рядки – різні об'єкти**, тому кращим є використання методу **equals()**

```
public boolean equals(Object anObject)
```

Метод порівнює рядок, на який викликається метод, з об'єктом **anObject**. Результат виклику методу буде **true**, тільки якщо **anObject** є рядком і значення порівнюваних рядків дорівнюють.

```
String str1 = new String("Рядок");
String str2 = new String("Рядок");
String str3 = str2 + "1";
int x=0, y=0;
if (str1.equals(str2))     x = 1;
if (str1.equals(str3))     y = 1;
```

Внаслідок виконання операторів змінна **x** отримає значення 1, а значення **y** залишиться рівним 0.

**Для порівняння рядків класу String визначено методи (public):**

- **boolean equalsIgnoreCase(String anotherString)** – порівняння значень рядків без урахування регістру літер;
- **boolean startsWith(String prefix)** – перевірка, чи міститься рядок **prefix** на початку рядка, що перевіряється;
- **boolean startsWith(String prefix, int toffset)** – перевірка, чи міститься підрядок рядка **prefix**, починаючи з позиції **toffset** на початку рядка, що перевіряється;
- **boolean endsWith(String suffix)** – перевірка, чи міститься рядок **prefix** в кінці рядка, що перевіряється;
- **boolean regionMatches(int toffset, String other, int ooffset, int len)** – порівнює **len** символів у двох рядках, причому в першому рядку порівнювані символи починаються з позиції **toffset**, а в другому рядку other – з позиції **ooffset**;

**Для порівняння рядків класу String визначено методи (public):**

• **boolean regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len)** – виконує ту ж операцію, що й попередній метод, але якщо значення **ignoreCase** при виклику методу дорівнює **true**, то регістри літер у підрядках, що порівнюються, ігноруються.

• **int compareTo(String anotherString)** – лексикографічно порівнює два рядки та повертає значення:

- **0**, якщо рядки дорівнюють за довжиною і мають однакове значення;
- **менше 0**, якщо в першій позиції, в якій символи рядків не рівні, код символу в першому рядку менший за код символу в другому рядку або довжина першого рядка менша за довжину другого рядка і всі символи першого рядка рівні символам в тих же позиціях другого рядка;
- **більше 0**, якщо в першій позиції, в якій символи рядків не рівні, код символу в першому рядку більший за код символу в другому рядку або довжина першого рядка більша за довжину другого рядка.

## Приклад

```
String str1 = "abc";
String str2 = "aBc";
boolean comp12 = str1.equalsIgnoreCase(str2);
```

змінна **comp12** отримає значення ***true***.

```
String str3 = "bcde";
String str4 = "abc";
boolean comp34 = str3.startsWith(str4,1);
```

змінна **comp34** отримає значення ***false***

```
String str5 = "abc";
String str6 = "abcde";
int comp56 = str5.compareTo(str6);
```

змінна **comp56** отримає значення ***менше 0***

```
String str7 = new String("Рядок1");
String str8 = new String("Новий рядок");
int x;
if (str7.regionMatches(true, 0, str8, 5, 5))
x = 1;
```

Значення виразу в дужках дорівнюватиме ***true***, оскільки перші 5 символів рядка **str7** збігаються з 5 символами рядка **str8**, починаючи з індексу 6 без урахування регістру символів.

# Пошук в рядках

**Для пошуку символів або послідовностей символів** (тільки в рядках класу **String**) використовуються такі методи, що перевантажуються**indexOf()** (public):

- **int indexOf(int ch)** – повертає першу позицію у рядку, в якому зустрічається символ **ch**;
- **public int indexOf(int ch, int fromIndex)** – повертає першу позицію в рядку, починаючи з позиції **fromIndex**, в якій зустрічається символ **ch**;
- **int indexOf(String str)** – повертає першу позицію у рядку, в якому зустрічається рядок **str**;
- **public int indexOf(String str, int fromIndex)** – повертає першу позицію в рядку, починаючи з позиції **fromIndex**, в якій зустрічається рядок **str**.

Для кожного методу **indexOf()** є відповідний метод **lastIndexOf(),** який шукає символ або рядок не з початку, а з кінця рядка.

Якщо символ або рядок не знайдено у рядку, в якому здійснюється пошук, методи**indexOf()** и **lastIndexOf()** повертають значення**-1.**

**Приклад**

```
String str1 = new String(“Рядок1");
String str2 = new String("Новий рядок");
int x, y, z;
x = str1.indexOf('к', 1);                    // x = 3
y = str2.indexOf("рядок");                   // y = 6
z = str2.indexOf("рядок1");                  // z = -1
```

# Вилучення з рядків

**Вилучення символів та підстрок із рядків String і StringBuffer** виконується за допомогою наступних методів (public):

- **char charAt(int index)** – повертає значення символу рядка у позиції **index**;
- **char[] toCharArray() –** повертає масив символів – копію рядка;
- **String substring(int beginIndex)** – повертає рядок, що починається з позиції **beginIndex** вихідного рядка і до кінця рядка;
- **String substring(int beginIndex, int endIndex)** – повертає рядок, що починається в позиції **beginIndex** і закінчується в позиції, на одиницю меншої **endIndex** у вихідному рядку;
- **void getChars (int srcBegin, int srcEnd, char[] dst, int dstBegin)** – копіює частину рядка, починаючи з символу позиції **srcBegin** і закінчуючи символом позиції **dstBegin** + (**srcEnd** - **srcBegin**) - 1 в символьний масив **dst**, починаючи з позиції **dstBegin**.

## Приклад

```
String str1 = new String("Рядок 1");
char firstSymbol = str1.charAt(0);
```

Змінна **firstSymbolstr** отримає значення **'P'**.

```
String str2 = new String("Новий рядок 2");
String substr21 = str2.substring(6);
String substr22 = str2.substring(6,11);
```

Змінна **substr21** отримає значення "рядок 2", а змінна **substr22** - значення "рядок".

```
StringBuffer str3 = new StringBuffer("String buffer");
char ch[] = new char[20];
str3.getChars(7, 10, ch, 0);
```

Вилучення символів рядка **str3** символьний масив **ch**.

# Модифікація рядків

Для **модифікації рядків** класу **String** визначено такі методи (public):

- **String concat(String str)** – повертає вихідний рядок, в кінець якого додано рядок **str**;
- **String toLowerCase()** – повертає рядок, у якому всі літери переведені в нижній регістр;
- **String toUpperCase()** – повертає рядок, у якому всі літери переведені у верхній регістр;
- **String trim()** – повертає рядок, в якому видалені всі пробілові символи (символи з кодами, що не перевищують '\u0020') на початку і в кінці рядка;
- **String replace(char oldChar, char newChar)** – замінює у рядку всі символи **oldChar** на **newChar**.

## Приклад

```
String str1 = new String("Рядок 1");
String upperStr1 = str1.toUpperCase();
```

Змінна **upperStr1** отримає значення "РЯДОК 1".

```
String str2 = new String("  Рядок 2  ");
str2 = str2.trim();
```

Змінна **str2** отримає значення "Рядок2".

```
String str3 = new String("a:b:c:d");
str3 = str3. replace(':',',');
```

Змінна **str3** отримає значення "a,b,c,d".

**Для створення рядків із примітивних типів даних** у класі **String** використовуються перевантажені статичні методи **valueOf(),** як аргумент яких задається константа, змінна або вираз примітивного типу(**boolean, char, int, long, float**, или **double**).

Значення методу, що повертається, є рядок класу **String** – рядкове подання аргументу.

```
String.valueOf(15);
```

повертає рядок"15"

- **static String valueOf(char[] data)** – повертає рядкову виставу символьного масиву **data**;

- **static String valueOf(char[] data, int offset, int count)** – повертає рядкове представлення символьного масиву **data**, починаючи з позиції **offset** та розміром **count**.

## Методи класу StringBuffer можуть безпосередньо модифікувати рядок

• **public void setCharAt(int index, char ch)** поміщає символ **ch** у вказаній позиції **index** рядка.

```
StringBuffer str =
new StringBuffer("String buffer");
str.setCharAt(3, 'X');
```

Змінна **str** після виконання методу **setCharAt()** міститиме символи "StrXng buffer""

• **public void deleteCharAt(int index)** видаляє символ у заданій позиції **index** рядка.

```
StringBuffer str = new StringBuffer("String
buffer");
str.deleteCharAt(2);
```

Після видалення символу 'r' довжина рядка зменшиться на 1 і рядок str матиме значення "Sting buffer".

• **public StringBuffer replace(int start, int end, String str)** замінює підрядок у рядку, починаючи з символу позиції **start** і до символу позиції **end - 1** рядком **str**.

```
StringBuffer str = new StringBuffer("String buffer");
str.replace(0,6,"Array");
```

Після заміни рядок матиме вигляд Array buffer і довжина рядка зменшиться на 1.

•**public StringBuffer append (тип-параметра ім'я-параметра)** додає символи до кінця рядка.

•**public StringBuffer insert(int offset, тип-параметра ім'я-параметра)** вставляє символи в будь-якому місці рядка, починаючи з позиції **offset**.

Обидва методи мають кілька версій, що дозволяють обробляти різні типи даних. Для **append** і **insert** **тип-параметра** може приймати значення:
**Object, String, char[], boolean, char, int, long, float, double**

**Приклад:**

Додавання рядкового подання цілого числа до кінця рядка

```
StringBuffer str1 = new StringBuffer("String buffer");
int value = 15;
str1.append(value);                    // str1="String buffer15"
```

Вставка символів

```
StringBuffer str2 = new StringBuffer("String buffer");
int value = 15;
str.insert(6, value);                  // str2="String15 buffer"
```

- **public StringBuffer append(char[] str, int offset, int len)** - метод додавання
- **public StringBuffer insert(int offset0, char[] str, int offset1, int len)** - метод вставки

Як аргумент передається частина масиву **char**, починаючи з індексу **offset** і довжиною **len**.

## Лектор:

Старший викладач кафедри Електроніки и комп'ютерної техніки Сумського державного університету

**Горячев О. Є.**


## В лекції використано матеріали авторів:

**Шонін В.А.**
**Монахов В.В.**