String

String в Java

String (рядок) - це об’єкт, в якому зберігається масив символів.

```
1  String s1 = "String #1"; // зі строкового літералу
2  String s2 = new String("String #2"); // як копію іншого рядка
3
4  char[] chars = {'S', 't', 'r', 'i', 'n', 'g', ' ', '#', '3'};
5  String s3 = new String(chars); // з масиву символів
6
7  byte[] bytes = {83, 116, 114, 105, 110, 103, 32, 35, 54};
8  String s4 = new String(bytes); // з масиву байт (кодів символів)
```

Операції з рядками

Основні методи для роботи з рядками:

- length() – вертає довжину рядка;
- charAt() – вертає **char** за вказаним індексом;
- concat() або “+” – додає (конкатенує) два рядки;
- contains() – перевіряє чи міститься в рядку вказаний підрядок;
- indexOf() – знаходить позицію першого співпадіння символа/підрядка;
- substring() – повертає підрядок між вказаними позиціями.
- equals() – перевіряє чи рядки однакові;

Детальніше (та більше методів):
https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html

length() і charAt()

```java
// введемо "Hello, world!"
String str = sc.nextLine();

// виведе 13
System.out.println("Length is: " + str.length());

// виведе 'H'
System.out.println("First symbol: " + str.charAt(0));

// виведе '!'
System.out.println("Last symbol: " + str.charAt(str.length() - 1));
```

toCharArray()

```
1  String s = sc.nextLine();
2
3  // отримання копії масиву символів з рядка
4  // !!! зміни в цьому масиві не впливають на існуючий рядок s
5  char[] chars = s.toCharArray();
6
7  for (int i = 0; i < chars.length; i++) {
8     if (Character.isLowerCase(chars[i])) {
9         chars[i] = Character.toUpperCase(chars[i]);
10    }
11 }
12
13 // створення нового рядку зі зміненого масива символів
14 String s1 = new String(chars);
15
16 System.out.println(s1);
```

concat()

```
1 String s1 = "Hello";
2 String s2 = "world";
3
4 // result1 = "Hello, world!"
5 String result1 = s1.concat(", ").concat(s2).concat("!");
6
7 // result2 = "Hello, world!"
8 String result2 = s1 + ", " + s2 + "!";
```

contains() та indexOf()

```
1  String s = "quick brown fox jumps over the lazy dog";
2
3  System.out.println(s.contains("fox")); // true
4  System.out.println(s.contains("wolf")); // false
5
6  System.out.println(s.indexOf('o'));     // 8 (in "brown")
7  System.out.println(s.lastIndexOf('o')); // 37 (in "dog")
8  System.out.println(s.indexOf('o', 9));  // 13 (in "fox")
9
10 System.out.println(s.indexOf('й')); // -1
11
12 System.out.println(s.indexOf("the lazy ")); // 27
```

substring()

```
1 String s = "Hello, world!";
2
3 System.out.println(s.substring(7)); // "world!"
4
5 System.out.println(s.substring(4, 9)); // "o, wo"
```

split()

```
1  String sentence = "Quick brown fox jumps over the lazy dog";
2
3  // split() розбиває рядок на масив рядків за певним розділювачем
4  String[] words = sentence.split(" ");
5
6  // [Quick, brown, fox, jumps, over, the, lazy, dog]
7  System.out.println(Arrays.toString(words));
```

equals()

```java
String s1 = new String("quack");
String s2 = new String("hello");
String s3 = new String("hello");

// порівняння по equals() (порівнюється вміст)
System.out.println(s1.equals(s2)); // false
System.out.println(s2.equals(s3)); // true

// порівняння по "==" (порівнюються посилання)
System.out.println(s1 == s2); // false
System.out.println(s2 == s3); // false

// регістр враховується при порівнянні
System.out.println(s2.equals("Hello")); // false
System.out.println(s2.equalsIgnoreCase("Hello")); // true
```

String Pool

Всі об’єкти String, які створюються з літералів, зберігаються в окремій структурі даних - String Pool’і. При створенні нового рядка з літералу, Java перевіряє чи є такий самий рядок у пулі: якщо немає - створює, якщо є - повертає посилання на існуючий. Новий об’єкт в такому випадку не створюється.

При створенні рядка за допомогою оператора **new**, кожен раз створюється новий об’єкт поза межами пула. Під кожен такий рядок виділяється пам’ять.

```
1  String s1 = "hello";
2  String s2 = "hello";
3  String s3 = new String("hello");
4
5  System.out.println(s1 == s2); // true
6  System.out.println(s1 == s3); // false
7
8  System.out.println(s1.equals(s2)); // true
9  System.out.println(s1.equals(s3)); // true
```

Immutability

Об’єкти String в Java - immutable, тобто стан об’єкта не змінюється ніколи після його створення. Кожна спроба змінити об’єкт призводить до створення нового об’єкту в пам’яті вже зі зміненим станом.

```
1  // в пулі з'являється "hello"
2  String s1 = "hello";
3
4  // в пулі з'являється "world"
5  String s2 = "world";
6
7  // в пулі з'являється " "
8  // в heap з'являється "hello "
9  s1 = s1 + " ";
10
11 // в heap з'являється "hello world"
12 // посилання на "hello " втрачено, але в пам'яті об'єкт залишився
13 s1 = s1 + s2;
```

StringBuilder та StringBuffer

Класи StringBuilder та StringBuffer забезпечують більш зручний інтерфейс для створення та зміни рядків, але всередині вони так само представляють собою масив символів.

Об’єкти класів StringBuilder та StringBuffer, на відміну від String, є змінюваними (mutable). Їх зміна не призводить до створення нових об’єктів в пам’яті. Тому якщо рядок буде часто змінюватись - краще скористатись одним із цих класів.

Єдина різниця між StringBuilder та StringBuffer у тому, що StringBuffer є thread-safe (потокобезпечним), тобто може безпечно використовуватись в багатопоточній середі за рахунок того що всі його методи синхронізовані.

StringBuilder та StringBuffer

```java
int n = sc.nextInt();

StringBuilder sb = new StringBuilder("string: ");
for (int i = 0; i < n; i++) {
   sb.append(i);
   sb.append(" ");
}

String s = sb.toString();

// “string: 0 1 2 3 4 5 6 7 8 9 10 11 12 ...”
System.out.println(s);
```