





# Lesson 7

14.07.2025

```
public class Ex1 {  
    public static void main(String[] args) {  
        do {  
            int y = 1;  
            System.out.println(y++ + " ");  
        } while (y <= 10);  
    }  
}
```



```
public class Ex2 {  
    public static void main(String[] args) {  
        boolean keepGoing = true;  
        int result = 15;  
        int i = 10;  
        do {  
            i--;  
            if (i == 8) keepGoing = false;  
            result -= 2;  
        } while (keepGoing);  
        System.out.println(result);  
    }  
}
```



```
public class Ex3 {  
    public static void main(String[] args) { args) {  
        Foo foo = new Foo();  
        System.out.println(foo.a);  
        System.out.println(foo.b);  
        System.out.println(foo.c);  
    }  
}
```

```
class Foo {  
    int a = 5;  
    protected int b = 6;  
    public int c = 7;  
}
```

```
public class Ex5 {  
    public static void main(String[] args) {  
        int myGold = 7;  
        System.out.println(countGold, 6);  
    }  
}  
  
class Hobbit {  
    int countGold(int x, int y) {  
        return x + y;  
    }  
}
```



# Java Strings

**Рядок** є послідовністю символів. Для роботи з рядками в Java визначено клас String, який надає ряд методів для маніпуляції рядками. Фізично об'єкт String є посиланням на область у пам'яті, в якій розміщено символи. Для створення нового рядка ми можемо використовувати один із конструкторів класу String, або безпосередньо присвоїти рядок у подвійних лапках:

```
String str1 = "Java";  
String str2 = new String(); // пустая строка  
String str3 = new String(new char[] {'h', 'e', 'l', 'l', 'o'});  
String str4 = new String(new char[] {'w', 'e', 'l', 'c', 'o', 'm', 'e'}, 3, 4);
```

При роботі з рядками важливо розуміти, що об'єкт String є незмінним (**immutable**). Тобто за будь-яких операцій над рядком, які змінюють цей рядок, фактично буде створюватися новий рядок.



# Основні методи класу String

Основні операції з рядками розкривається через методи класу String, серед яких можна виділити такі:

**concat()**: об'єднує рядки

**valueOf()**: перетворює об'єкт у рядковий вигляд

**join()**: з'єднує рядки з урахуванням роздільника

**compareTo()**: порівнює два рядки

**charAt()**: повертає символ рядка за індексом

**getChars()**: повертає групу символів

**equals()**: порівнює рядки з урахуванням регістру

**equalsIgnoreCase()**: порівнює рядки без урахування регістру

**regionMatches()**: порівнює підрядки у рядках

**indexOf()**: знаходить індекс першого входження підрядка в рядок

**lastIndexOf()**: знаходить індекс останнього входження підрядка в рядок

**startsWith()**: визначає, чи починається рядок з підрядка

**endsWith()**: визначає, чи закінчується рядок на певний підрядок

**replace()**: замінює в рядку один підрядок на інший



**trim():** видаляє початкові та кінцеві пробіли

**substring():** повертає підрядок, починаючи з певного індексу до кінця або до певного індексу

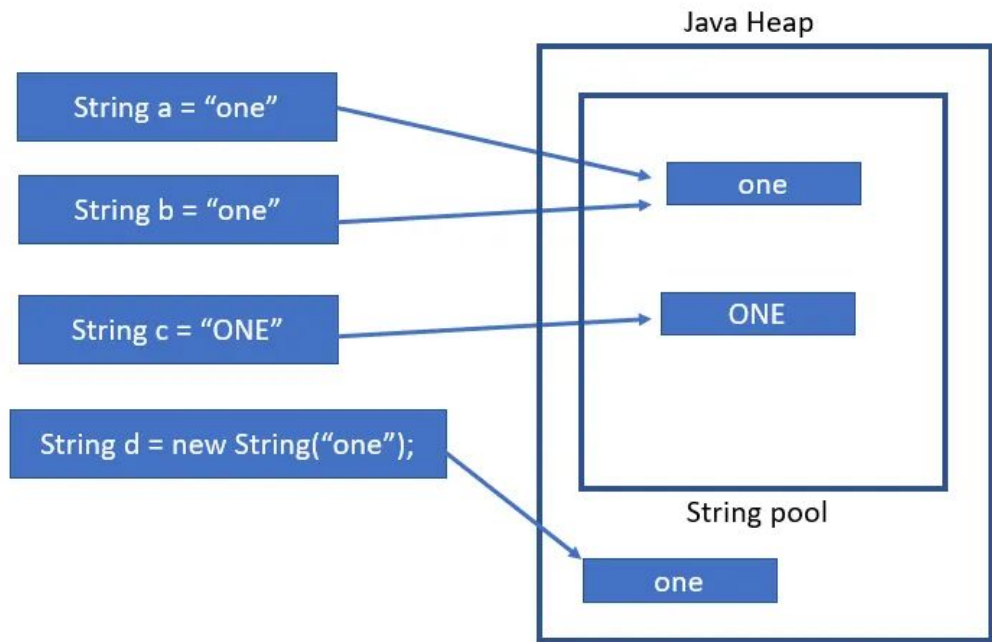
**toLowerCase():** перекладає всі символи рядка в нижній регістр

**toUpperCase():** перекладає всі символи рядка у верхній регістр



Пул рядків (String Pool) - це безліч рядків у купі (Java Heap Memory). Ми знаємо, що String - особливий клас у java, за допомогою якого ми можемо створювати рядкові об'єкти.

На діаграмі нижче ми бачимо, як саме рядковий пул розташований у пам'яті Java Heap. І як різні способи створення рядків впливають на розміщення їх у пам'яті.



```
a == b; // true
```

```
a == c; // false
```

```
a.equals(b); // true
```

```
a.equalsIgnoreCase(c); // true
```

```
a == d; // false
```

```
a.equals(d); // true
```



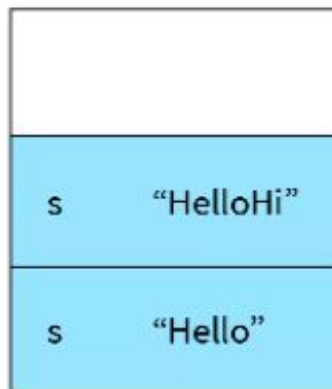
Коли ми використовуємо подвійні лапки, щоб створити новий рядок, то першим справа йде пошук рядка з таким же значенням в пулі рядків. Якщо java таку рядок знайшла, то повертає посилання, інакше створюється новий рядок у пулі, а потім повертається посилання.

Однак використання оператора `new` змушує клас `String` створити новий об'єкт `String`. Після цього можемо використовувати метод **`intern()`**, щоб помістити цей об'єкт в пул рядків або звернутися до іншого об'єкта з пула рядків, який має таке саме значення

## Immutable

- cannot be changed

```
String s=new String  
("Hello"); s+="Hi";
```



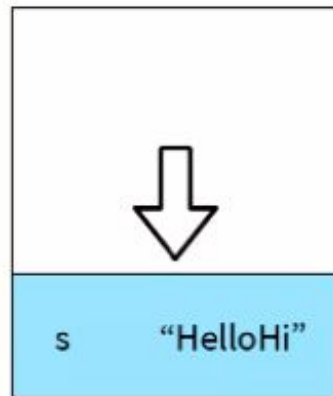
String

Memory

## Mutable

- can be changed

```
StringBuffer s=new  
StringBuffer("Hello");  
s.append("Hi");
```



StringBuffer

Memory



StringBuffer	StringBuilder
StringBuffer is a part of Java since its initial version 1.0	StringBuilder is introduced in Java 5.
All the methods of the StringBuffer class are synchronised.	Methods are not synchronised in the StringBuffer class.
StringBuffer is safe to use concurrently by multiple threads. It is thread-safe.	Instances of StringBuilder are not safe for use by multiple threads. It is not thread-safe.
The performance of StringBuffer is slow.	The performance of StringBuffer is fast.
StringBuffer objects are mutable.	StringBuilder objects are mutable too.
The storage area used by StringBuffer is heap memory.	The storage area used by StringBuilder is also heap memory.