

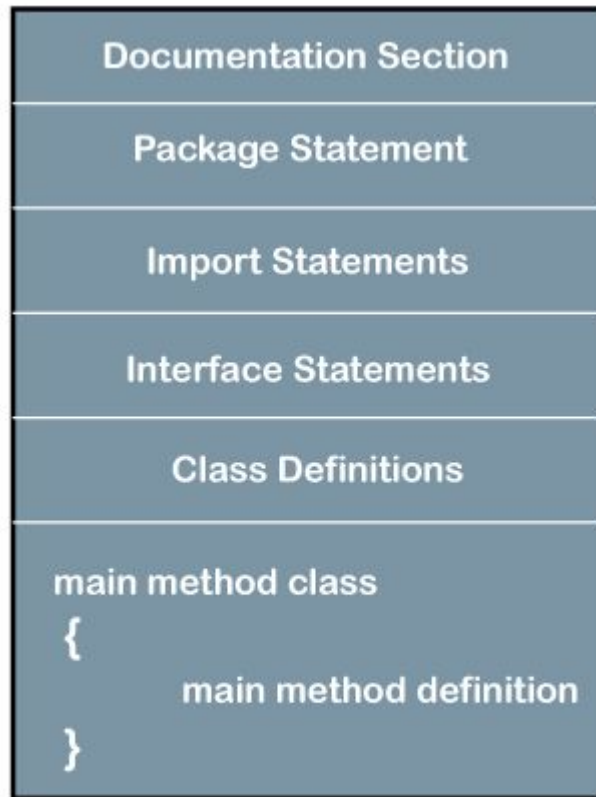


Lesson 2

01.05.2023



Структура класу Java



Structure of Java Program

```
package org.allyourcode.myfirstproject;
```

```
public class MyFirstJavaClass {  
    /**  
     * @param args  
     */  
    public static void main (String[] args) {  
        javax.swing.JOptionPane.showMessageDialog  
            (null, "Hello");  
    }  
}
```



Types of Java Comments

01

Single Line

The single line comment is used to comment only one line.

02

Multi Line

The multi line comment is used to comment multiple lines of code.

03

Documentation

The documentation comment is used to create documentation API. To create documentation API, you need to use javadoc tool.

Класс vs. Файл

```
// Compiling this Java program would  
// result in multiple class files.
```

```
class Sample  
{  
  
}
```

```
// Class Declaration
```

```
class Student  
{  
  
}
```

```
// Class Declaration
```

```
class Test  
{  
    public static void main(String[] args)  
    {  
        System.out.println("Class File Structure");  
    }  
}
```



main()

```
1 | public static void main(String[] args) { }
```

public – модифікатор доступу

static – ключове слово, яке показує що не треба створювати екземпляр класу для виклику методу

void – значення, що повертається (нічого не повертати)

main – ім'я методу, яке дозволяє JVM ідентифікувати початкову точку запуску програми

String[] args – масив параметрів, з якими може запускатися програма

Імена змінних та методів

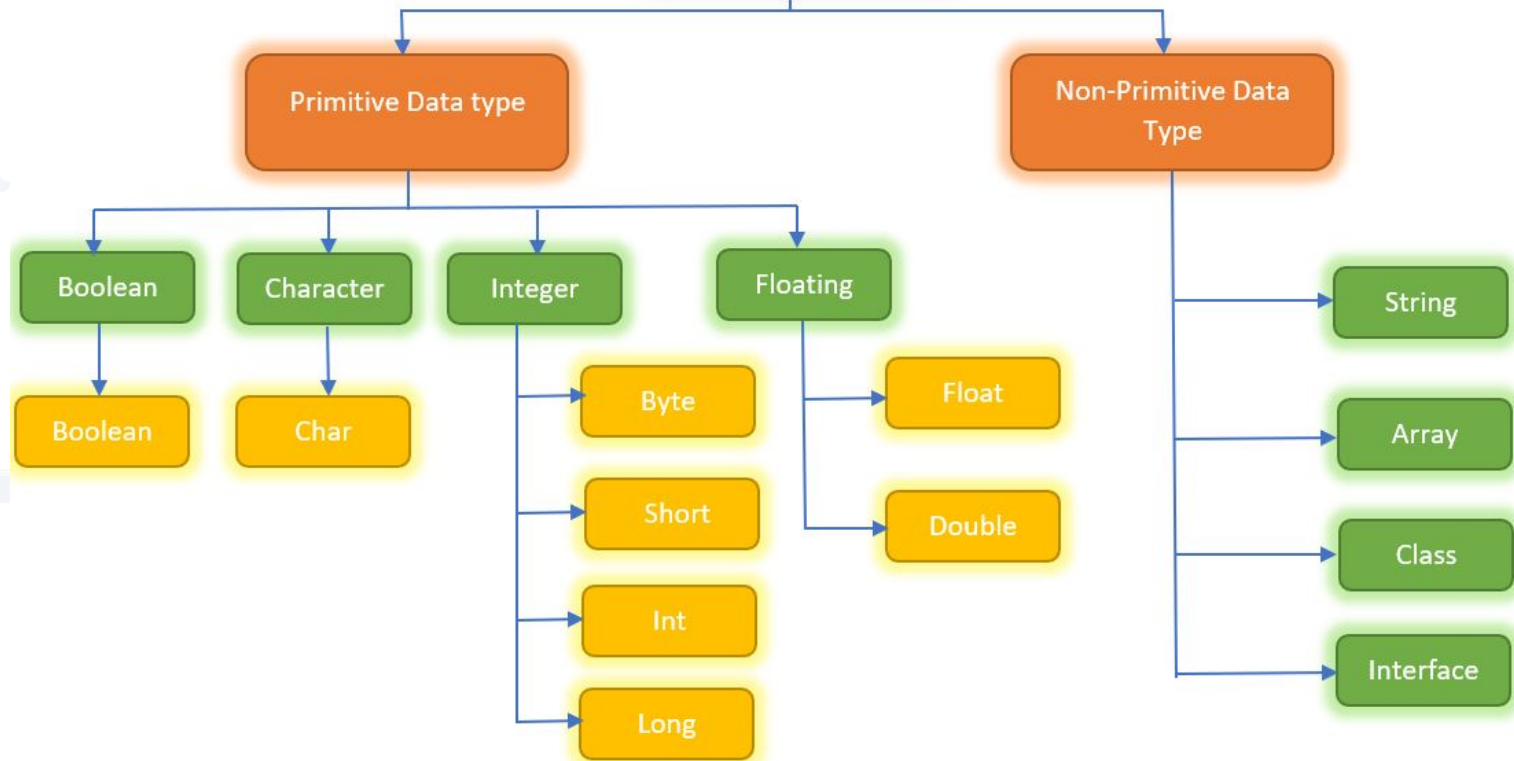
1. Ім'я має складатися із латинських букв. Хоч джава і підтримує кирилицю — все ж таки краще називати змінні латиницею. Від себе додаю давайте їм англійські назви.
2. Назва має нести певний сенс. Це означає, що змінні з ім'ям **a** або **newVariable** не буде нести ніякого сенсу. Постарайтеся, щоб ім'я було «розмовляючим». Це нелегка праця — вигадувати імена для змінних, класів та методів. Згодом та з досвідом робити це буде легше.
3. Не можна використовувати цифри на початку імені. Це не тільки негласне правило, а й правило мови java. Якщо Ви спробуєте використовувати цифри на початку імені – програма не скомпілюється. Якщо ж цифри в середині або в кінці імені - то їх використання припустимо якщо ж цифра нестиме якийсь сенс змінної, а чи не просто **a1**, **a2**.
4. Хоча використання символу \$ і _ допустимо на початку назви, я б порадив Вам по можливості уникати ці символи в іменах

5. Програмісти, які пишуть програми мовою java використовують **camelCase** (верблюжий стиль). Якщо ім'я змінної складається більше, ніж із одного слова Наступне слово пишеться з великої літери: `userName`, `jackpotFunctionalView`.

6. Не можна використовувати зарезервовані слова як ім'я. На малюнку Нижче представлені всі службові 54 слова в мові java. Вчити та запам'ятовувати їх не потрібно. Згодом вони самі «засядуть» у голові.

<code>abstract</code>	<code>continue</code>	<code>for</code>	<code>new</code>	<code>switch</code>
<code>assert</code>	<code>default</code>	<code>goto*</code>	<code>package</code>	<code>synchronized</code>
<code>boolean</code>	<code>do</code>	<code>if</code>	<code>private</code>	<code>this</code>
<code>break</code>	<code>double</code>	<code>implements</code>	<code>protected</code>	<code>throw</code>
<code>byte</code>	<code>else</code>	<code>import</code>	<code>public</code>	<code>throws</code>
<code>case</code>	<code>enum</code>	<code>instanceof</code>	<code>return</code>	<code>transient</code>
<code>catch</code>	<code>extends</code>	<code>int</code>	<code>short</code>	<code>try</code>
<code>char</code>	<code>final</code>	<code>interface</code>	<code>static</code>	<code>void</code>
<code>class</code>	<code>finally</code>	<code>long</code>	<code>strictfp</code>	<code>volatile</code>
<code>const*</code>	<code>float</code>	<code>native</code>	<code>super</code>	<code>while</code>

Data Types in Java



Java's primitive types


Ordered by descending upper range limit

CATEGORIZATION	NAME	UPPER RANGE	LOWER RANGE	BITS
Floating point	double	Really huge positive	Really huge negative	64
	float	Huge positive	Huge negative	32
Integral	long	9,223,372,036,854,775,807	-9,223,372,036,854,775,808	64
	int	2,147,483,647	-2,147,483,648	32
	char	65,535	0	16
	short	32,767	-32,768	16
	byte	127	-128	8
Boolean	boolean	No numeric equivalent	true or false	1

Створення та ініціалізація змінної

1. За допомогою операції надання знака = ми присвоюємо значення змінної. Причому вона завжди відпрацьовує праворуч-наліво:

k = 10




Значення 10 присвоюється змінною k

2. Надавати значення змінної ми можемо двома способами: в 2 рядки або в 1 рядок

```
1 class Test {  
2     public static void main(String args[]){  
3         int k;  
4         k = 10;  
5         System.out.println (k);  
6     }  
7 }
```

```
1 class Test {  
2     public static void main(String args[]){  
3         int k = 10;  
4         System.out.println (k);  
5     }  
6 }
```



3. Також необхідно запам'ятати:

Якщо привласнюємо значення змінної типу String, укладаємо його у подвійні лапки

```
String title = "Java";
```

Якщо надаємо значення змінної типу char, укладаємо його в одинарні лапки

```
char letter = 'M';
```

Якщо надаємо значення змінної типу float, обов'язково після нього додаємо букву f

```
float pi = 3.14f;
```

4. Слово "ініціалізація" - це те саме, що і "привласнити початкове" значення змінної"



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

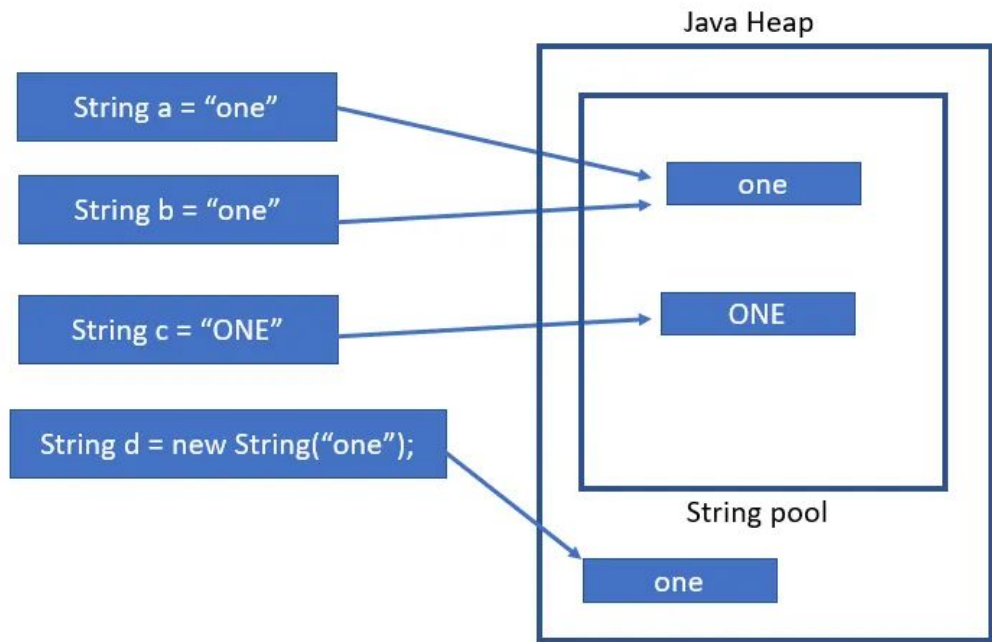
substring(): повертає підрядок, починаючи з певного індексу до кінця або до певного індексу

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
```

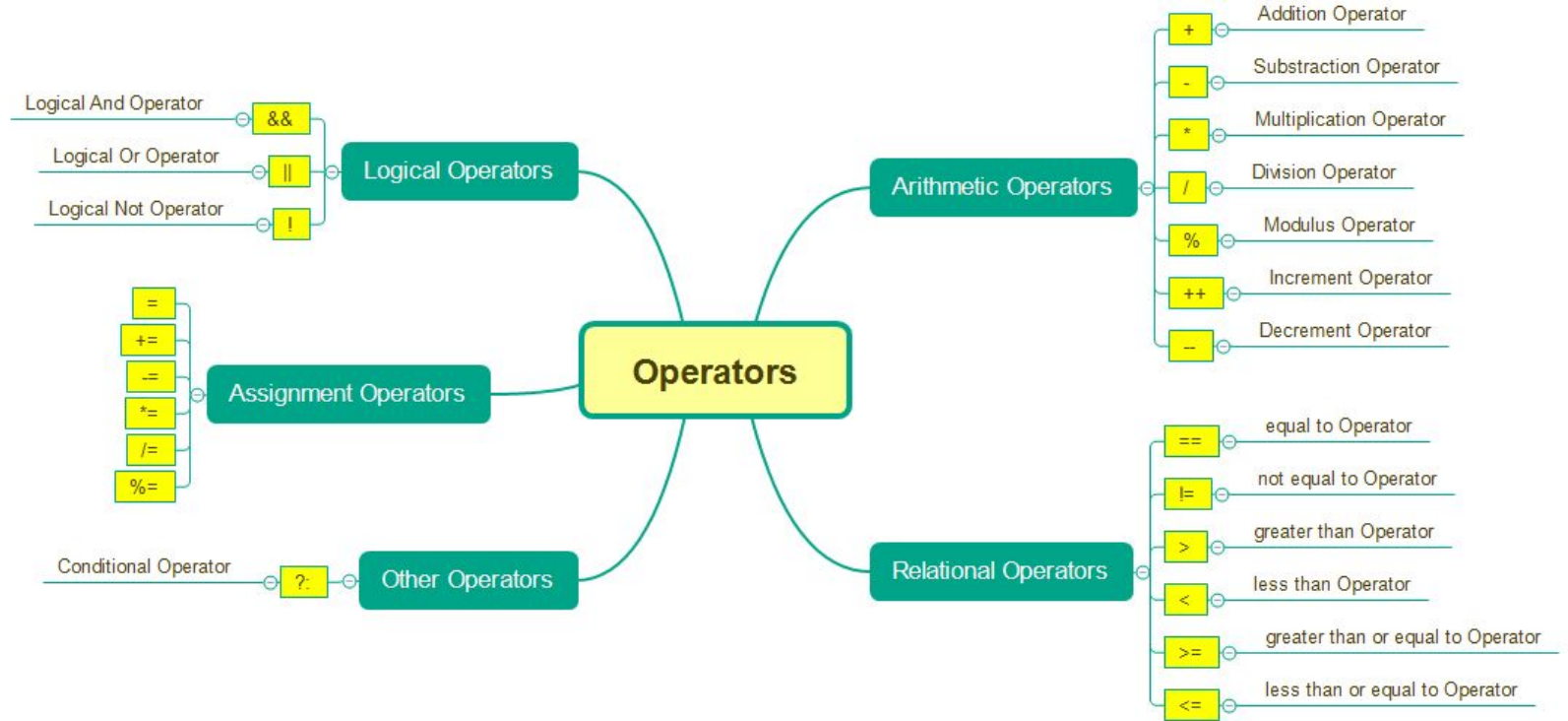
```
a == d; // false
```

```
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





Інкремент та декремент

У програмуванні дуже часто доводиться виконувати операції, коли:

змінна має збільшитися на одиницю $\rightarrow +1$

змінна має зменшитися на одиницю $\rightarrow -1$

Тому вигадали окремі операції зі змінними, які називаються інкремент та декремент.

Інкремент – відповідає за збільшення змінної на одиницю. Позначається як $++$. Наприклад, якщо у нас є змінна i і ми до неї застосуємо інкремент, тоді це буде записано як $i++$. А це означає, що значення змінної i має бути збільшено на 1

Декремент – відповідає за зменшення змінної на одиницю. Позначається як $--$. Наприклад, якщо у нас є змінна n і ми до неї застосуємо декремент, тоді це буде записано як $n--$. А це означає, що значення змінної n має бути зменшено на 1.

Тож у чому ж відмінність між постфіксною та префіксною формами?

У постфікській формі: спочатку використовується старе значення у обчисленнях далі у наступних обчисленнях використовується вже нове значення

У префікській формі: відразу використовується нове значення у обчисленнях

Operator	Result
&	Logical AND
	Logical OR
^	Logical XOR (exclusive OR)
	Short-circuit OR
&&	Short-circuit AND
!	Logical unary NOT
&=	AND assignment
=	OR assignment
^=	XOR assignment
==	Equal to
!=	Not equal to

Parenthesis

Postfix

Prefix

Multiplicative

Additive

Relational

Logical



Operators	Notation	Precedence/Priority
Postfix	expr++, expr--	1
Unary	++expr, --expr, +expr -expr, ~, !	2
Multiplicative	*, /, %	3
Additive	+, -	4
Shift	<<, >>, >>>	5
Relational	<, >, <=, >=, instanceof	6
Equality	==, !=	7
Bitwise AND	&	8
Bitwise Exclusive OR	^	9
Bitwise Inclusive OR		10
Logical AND	&&	11
Logical OR		12
Ternary	?:	13
Assignment	=, +=, -=, *=, /=, %=, &=, ^=, =, <<=, >>=, >>>=	14