Змінні

- Оголошення змінних
- Типи змінних
- Правила іменування
- Примітивні типи даних і рядки
- Масиви

Оголошення змінних

- Змінні використовуються в програмі для зберігання даних. Будь-яка змінна має три базових характеристики:
  - ім'я;
  - тип;
  - значення.

Характеристики змінних

- Ім'я унікально ідентифікує змінну і дозволяє до неї звертатися в програмі.
- Тип описує, які величини може зберігати змінна.
- Значення - поточна величина, що зберігається в змінної на даний момент.
- Робота зі змінною завжди починається з її оголошення (declaration). Звичайно, воно має включати в себе ім'я оголошуваної змінної.
  - У Java будь-яка змінна має строгий тип, який задається при оголошенні і ніколи не змінюється.

Типи змінних (1 з 2)

- **Змінні примірника** (Instance Variables) - значення змінної екземпляра класу, унікальне для кожного екземпляра класу
- **Змінні класу** (Class Variables) - всі змінні які оголошені як статичні (за допомогою модифікатора static).
  - Існує всього одна копія статичної змінної в незалежності від кількості екземплярів класу

Типи змінних (2 з 2)

- Змінні методу (Local Variables) - подібно до того, як об'єкт зберігає свій стан в полях, методи часто зберігають їх тимчасовий стан в локальних змінних.
- Параметри методів

Правила іменування

- Ім'я **повинно** бути допустимим ідентифікатором.
- Ім'я **не повинно** бути ключовим словом, логічним літералом (true або false), або зарезервованим словом null.
- Ім'я **повинно бути унікальним** в своїй області видимості.

Оголошення змінних

type *identifier* [= value][, *identifier* [= value]]

- Значення може бути зазначено **одразу** (ініціалізація):
  - **статична** ініціалізація: int d = 3, e, f = 5;
  - **динамічна** ініціалізація: double c = Math.sqrt (4.);
- У більшості випадків завдання початкової величини можна і відкласти:
  - int d, e, f;

Примітивні типи даних

- У мові Java існує вісім примітивних типів.
- Всі вони визначені специфікацією мови та є ключовими словами:
-
  - byte
  - short
  - int
  - long
  - float
  - double
  - boolean
  - char

Примітивні типи даних **byte**

- byte являє собою **8-бітове** число.
  - має мінімальне значення **-128** і максимальне значення **127** (включно).
  - тип даних може бути корисним для **економії пам'яті у великих масивах**, де така економія дійсно має значення.
  - може бути використаний **замість int**, де його обмеження допомагають **уточнити код**: той факт, що діапазон змінної обмежений може служити формою документації.

Примітивні типи даних short

- short являє собою **16-бітове** число.
  - має мінімальне значення **-32768** і максимальне значення **32767** (включно).
  - область застосування типу **аналогічна** з областю застосування типу **byte**.

Примітивні типи даних int і long

.int являє собою **32-бітове** число.

- має мінімальне значення **-2,147,483,648** і максимальне значення **2,147,483,647** (включно).
- застосуємо в більшості випадків як тип за замовчуванням для **цілих чисел**.
- у випадку якщо довжини типу **не достатньо** необхідно використовувати **long**.

.long являє собою **64-бітове** число.

- мінімальне значення **-9,223,372,036,854,775,808** і максимальне значення **9,223,372,036,854,775,807** (включно).

Примітивні типи даних float і double

- **float** — тип даних **з плаваючою комою** одинарної точності розміром **32 біта**.
  - визначений стандартом IEEE 754
  - тип даних може бути корисним для економії пам'яті у великих масивах замість double.
- **double** - тип даних з плаваючою комою подвійної точності розміром 64 бита.
- **заборонено** використовувати float та double для зберігання точних величин (наприклад, кількості грошей)

Примітивні типи даних boolean

- **boolean** має лише два значення: **true** або **false**
- використовуйте цей тип даних в якості простого **прапора істина/хиба** в умовах.
- цей тип даних являє **один біт** інформації, але його "розмір" чітко не визначений.

Примітивні типи даних **char**

- Тип даних **символ** являє собою **один символ** в кодуванні **Unicode**.
- Розмір типу **16 біт**.
- мінімальне значення '\ **u0000**' і максимальне значення '\ **uffff** '.

Тип даних String

- введений для підтримки **рядків**.
- **будь-який текст** укладений у **подвійні лапки** "" являє собою екземпляр класу java.lang.String.
    - String s = "це рядок";

Значення за замовчуванням

- Змінні **примірника** і **класу** автоматично ініціюються значеннями за замовчуванням.

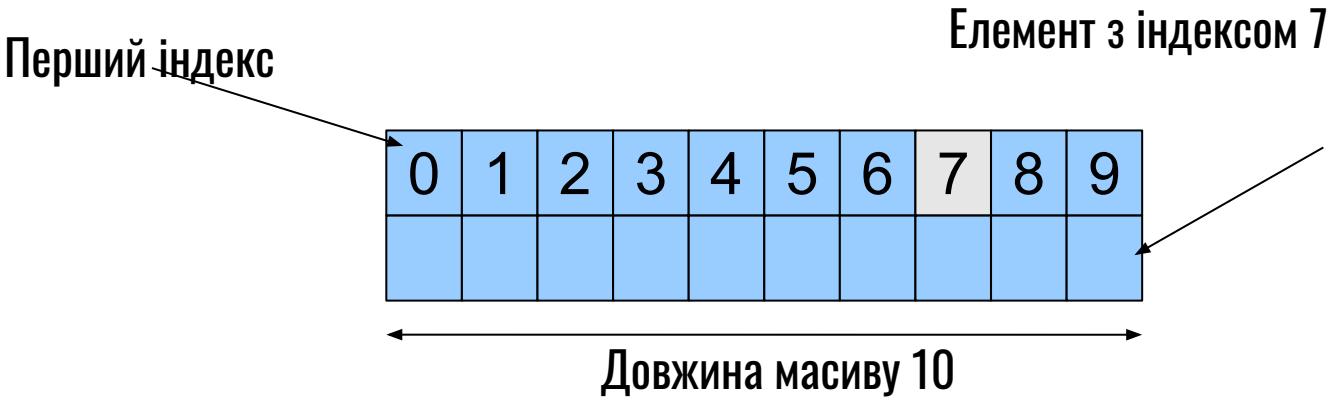
Тип	Значення за замовчуванням
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (або об'єкт)	null
boolean	false

Закінчення типів літералів

- За замовчуванням тип **цілочисельного** літерала **int**
- Якщо літерал **закінчується на l або L** то його тип **long**
- За замовчуванням тип литерала **з плаваючою точкою double**
- Якщо літерал **з плаваючою точкою закінчується на f або F** то його тип **float**
- Якщо літерал **закінчується на d або D** то його тип **double**

Масиви

- Масив — це кінцева послідовність впорядкованих елементів одного типу, доступ до кожного елементу в якій здійснюється за його індексом.



Оголошення масивів

- Варіанти оголошення масиву:
  - тип [] ім'я;
  - тип ім'я [];
  - тип [] ім'я = new тип [розмір];
  - тип [] ім'я = {ел0, ел1, ..., елN};
- Наприклад:
- a = new int [10]; // Масив з 10 елементів типу int
- int n = 5;
- ar1 = new double [n]; // Масив з 5 елементів double

class ArrayDemo

```
class ArrayDemo {
    public static void main(String[] args) {
        int[] anArray;  // declares an array of integers
        anArray = new int[10]; // allocates memory for 10 integers
        anArray[0] = 100; // initialize first element
        anArray[1] = 200;  // initialize second element
        anArray[2] = 300;// etc.
        System.out.println("Element at index 0: " + anArray[0]);
        System.out.println("Element at index 1: " + anArray[1]);
        System.out.println("Element at index 2: " + anArray[2]);
        System.out.println("Element at index 3: " + anArray[3]);
        System.out.println("Element at index 4: " + anArray[4]);
    }
}
```

Розмір масиву

Для визначення розміру масиву використовується властивість **length**

```
System.out.println (anArray.length);
```

Багатовимірні масиви

У Java багатовимірні масиви це одномірні масиви з елементами у вигляді масивів
тип [] [] ім'я; або тип ім'я [] [];
тип [] [] ім'я = new тип [розмір] [розмір];
тип [] [] ім'я = {{...}, {...}, {...}, {...}}

int[][] twoD = new int[3][5]

«Нерівні» масиви

```
int twoD[][] = new int[4][];
   twoD[0] = new int[1];
   twoD[1] = new int[2];
   twoD[2] = new int[3];
   twoD[3] = new int[4];
```