Типи даних і оголошення змінних.

Мова програмування Java є **мовою зі строгою типізацією** (strongly typed language) - кожна змінна і кожен вираз мають тип, який повинен бути відомий під час компіляції програми.

Тип обмежує набір значень, які можуть бути присвоєні змінної, або отримані в вираженні, обмежує операції над значеннями і визначає реалізацію конкретної операції.

В Java визначено дві категорії даних:

- примітивні типи (primitive types);
- посилальні типи (reference types).

Оголошення змінної:

```
ім'я-типу ідентифікатор-змінної;
```

```
int x;
String string1;
```

Оголошення декількох змінних одного типу:

```
ім'я-типу ідентифікатор-змінної-1, ідентифікатор-змінної-2,…;
```

```
int x1, x2, x3;
```

Змінна є зазначенням місця зберігання значення змінної в пам'яті. Змінна примітивного типу завжди містить значення змінної зазначеного типу, а змінна посилального типу зберігає посилання (адресу) об'єкта зазначеного типу.

Створювати нові змінні можна в будь-якому місці програми.
Будь-яке оголошення змінної має свою **область дії**, межі якої залежать від того, де саме розташоване це оголошення.

При розміщенні фрагмента тексту програми в пару фігурних дужок {} створюється новий **блок**. Блоки можуть бути вкладеними. Змінна доступна в блоці тільки в тому випадку, якщо вона визначена в цьому блоці або в одному з вищих блоків, в який входить поточний блок.

```
{
int x;
…
          {
        int y;
        …
        }
}
```

початок блоку 1

- початок блоку 2

- кінець блоку 2
- кінець блоку 1

Змінна **x** визначена і в блоці **1** і в блоці **2**, а змінна **y** - тільки в блоці**2**.

У внутрішньому блоці не можна оголошувати змінну з таким же ім'ям, як у зовнішній області дії

Кожна змінна має **час життя**. Коли поточний блок, в якому змінна була оголошена, закінчується, вона стає доступною для знищення за допомогою так званого **збирача сміття**.

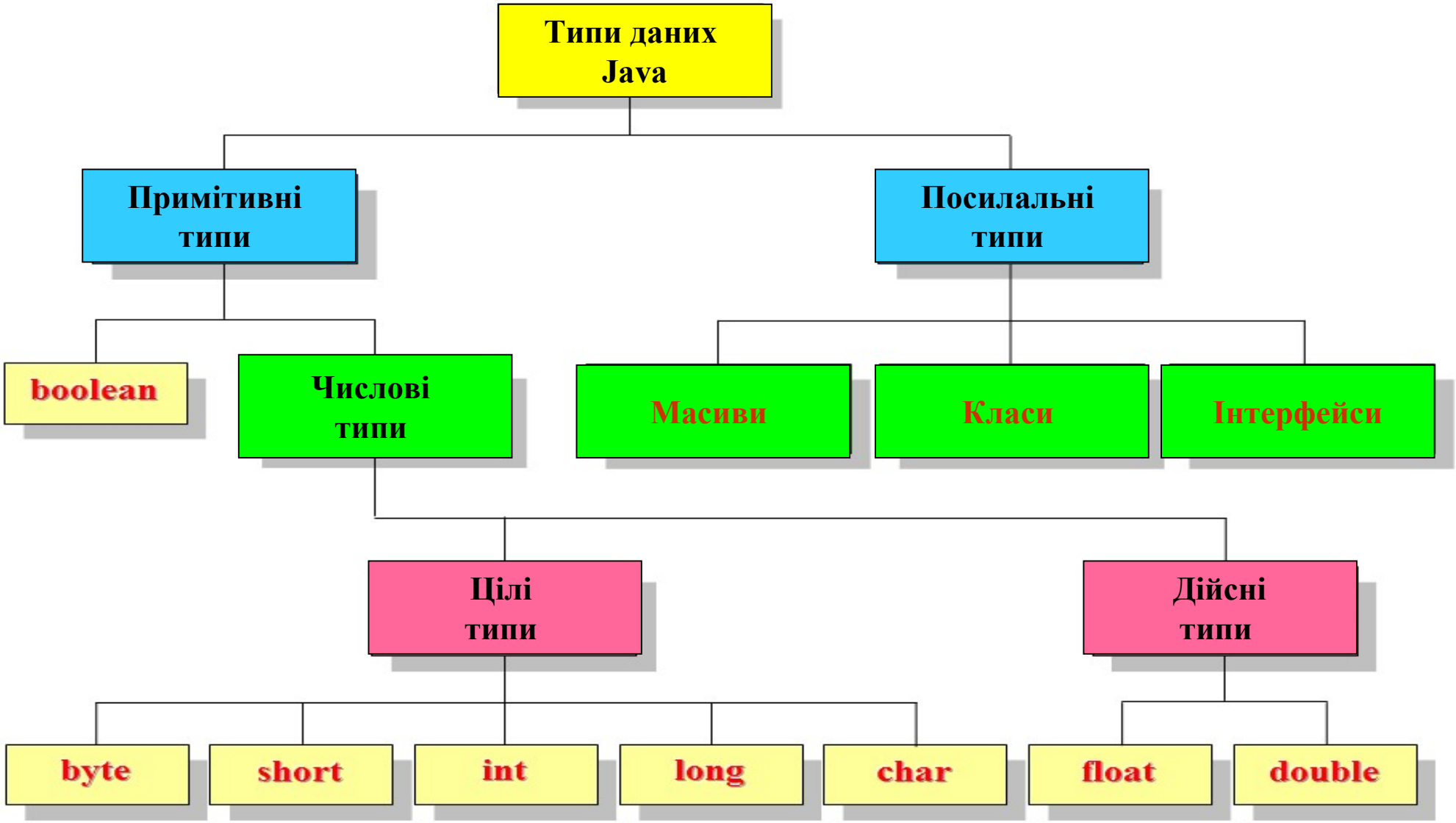
Змінити область дії і час життя змінної можна за допомогою **модифікаторів доступу**. Модифікатори вставляються в оголошення змінної перед *ім'ям-типу*.

За замовчуванням (без модифікатора) змінна доступна тільки класам в тому ж пакеті, що і клас, в якому вона міститься.

Модифікатор **public** визначає, що змінна доступна як всередині, так і поза класом, тобто змінна є глобальною і доступна будь-якому іншому об'єкту.

Модифікатор **private** означає, що змінна доступна тільки в тому класі, в якому вона була оголошена.

Модифікатор **final** визначає, що змінна має постійне (незмінне) значення і не може бути перевизначена.


Типи даних Java
Примітивні типи
Посилальні типи
boolean
Числові типи
Масиви
Класи
Інтерфейси
Цілі типи
Дійсні типи
byte
short
int
long
char
float
double

Примітивні типи даних

Булевий (логічний) тип

Величини типу **boolean** приймають значення **true** або **false**.

Цілочисельні типи

Тип	кількість байтів	Діапазон значень	Опис
byte	1	-128..127	Однобайтове ціле число (8-бітове ціле зі знаком)
short	2	$-2^{15}..2^{15}-1$ = -32768.. 32767	Коротке ціле число (16-бітове ціле зі знаком)
char	2	\u0000..\uFFFF=0.. 65535	Символьний тип (16-бітове ціле без знаку)
int	4	$-2^{31}..2^{31}-1$ = $-2.147483648 \cdot 10^{9}..$ $2.147483647 \cdot 10^{9}$	Ціле число (32-бітове ціле зі знаком)
long	8	$-2^{63}..2^{63}-1$ = $-9.22337203685478 \cdot 10^{18}..$ $9.22337203685478 \cdot 10^{18}$	Довге ціле число (64-бітове ціле зі знаком)

Приклади оголошення цілих змінних:

```
int i,j,k;
int j1;
byte i1, i2=-5;
short i3=-15600;
long m1=1, m2, m3=-100;
```

```
byte a1 = 0xF1, a2 = 0x07;
short r1 = 017;
```

При оголошенні в класі значення цілочисельний змінної за замовчуванням дорівнює нулю

У методі всі змінні перед використанням обов'язково потрібно ініціалізувати

Константами називаються іменовані комірки пам'яті з незмінним вмістом. Оголошення констант здійснюється в класі, при цьому перед ім'ям типу константи ставиться комбінація зарезервованих слів **public** і **final**:

```
public final int MAX1=255;
public final int MILLENIUM=1000;
```

Робота з числами в мові Java

Для зберігання від'ємних чисел використовується ***додатковий код***. Машинне подання позитивного числа n перекладається в машинне подання числа $n_2 = -n$ за наступним алгоритмом:

1. Число $\boldsymbol{n}$ перетворюється в число $\boldsymbol{n_1}$ шляхом заміни всіх одиниць числа $\boldsymbol{n}$ нулями, а нулів одиницями.

2. Переклад $\boldsymbol{n_1}$ в $\boldsymbol{n_2 = n_1 + 1}$ (тобто до одержаного числа $\boldsymbol{n_1}$ додається одиниця молодшого розряду).

Основні оператори для роботи з цілочисельними величинами

+	Оператор складання
–	Оператор вирахування
*****	Оператор множення
/	Оператор ділення
%	Оператор залишку від цілочисельного ділення
=	Оператор присвоювання
++	Оператор інкременту (збільшення на 1)
+=	**v+=i** еквівалентно **v=v+i**
-=	**v-=i** еквівалентно **v=v-i**
***=**	**v*=i** еквівалентно **v=v*i**
/=	**v/=i** еквівалентно **v=v/i**
%=	**v%=i** еквівалентно **v=v%i**

Integer.parseInt(рядок)
Long.parseLong(рядок)

- перетворення строкового подання числа в ціле значення

Побітові операції

Побітові операції розглядають числові значення як поля бітів

Зміщення вліво з урахуванням знака <<

```
int x =  31, z;     // x = 00000000 00000000 00000000 00011111
z = x << 2;         // z = 124: 00000000 00000000 00000000 01111100
```

Зміщення вправо з урахуванням знака >>

```
int x =  -17, z;    // x: = 11111111 11111111 11111111 11101111
z = x >> 2;         // z = -5: 11111111 11111111 11111111 11111011
```

Зміщення вправо без урахуванням знака >>>

```
int x =  -17, z;    // x = 11111111 11111111 11111111 11101111
z = x >>> 2;        // z = 1073741819
                    // z:   00111111 11111111 11111111 11111011
```

Побітове ТА &

```
int x =  112;      // x = 00000000 00000000 00000000 01110000
int y = 94;        // y = 00000000 00000000 00000000 01011110
int z;
z = x & y;         // z=80: 00000000 00000000 00000000 01010000
```

Побітове АБО |

```
int x =  112;      // x = 00000000 00000000 00000000 01110000
int y = 94;        // y = 00000000 00000000 00000000 01011110
int z;
z =x | y;          // z = 126: 00000000 00000000 00000000 01111110
```

Побітове виключаюче АБО ^

```
int x =  112;      // x = 00000000 00000000 00000000 01110000
int y = 94;        // y = 00000000 00000000 00000000 01011110
int z;
z =x ^ y;          // z = 46: 00000000 00000000 00000000 00101110
```

Операції порівняння

Ці операції мають два операнда і повертають логічне значення, відповідне результату порівняння (**false** або **true).**

"==" (дорівнює), **"!=" (не дорівнює),**
">" (більше), **">=" (більше або дорівнює),**
"<" (менше) **"<=" (менше або дорівнює)**

Приклад:

```
boolean isEqual, isNonEqual, isGreater,
isGreaterOrEqual, isLess, isLessOrEqual;
int x1 = 5, x2 = 5, x3 = 3, x4 = 7;
  isEqual = x1 == x2;                    // isEqual = true
  isNonEqual = x1 != x2;                 // isNonEqual = false
  isGreater = x1 > x3;                   // isGreater = true
  isGreaterOrEqual = x2 >= x3;           // isGreaterOrEqual = true
isLess = x3 < x1;                        // isLess = true
isLessOrEqual = x1 <= x3;
```

Булеві операції

Виконуються над **boolean** змінними і їх результатом також є значення типу **boolean**.

Заперечення "!" **виключаюче АБО "^"**

ТА "&" **АБО "|"**

Крім того, до **boolean** операндів можуть бути застосовні операції **"==" (дорівнює)** и **"!=" (не дорівнює).**

Скорочені (short-circuit) логічні операції

скорочене ТА "&&" **скорочене АБО "||"**

При використанні цих операцій другої операнд взагалі не буде обчислюватися, що корисно в тих випадках, коли правильне функціонування правого операнда залежить від того, чи має лівий операнд значення **true** або **false**.

Приклади булевих операцій:

```
boolean isInRange, isValid, isNotValid,
isEqual, isNotEqual;
int x = 8;
isInRange = x > 0 && x < 5;          // isInRange = false
isValid = x > 0 || x > 5;            // isValid = true
isNotValid = !isValid;               // isNotValid = false
isEqual = isInRange == isValid;      // isEqual = false
isNotEqual = isInRange != isValid    // isNotEqual = true
```

Пріоритет операторів

Пріоритет	Група операторів	Оператори				
1 *вищий*	Постфіксні	( )	[ ]	.		
2	Унарні	++ операнд операнд ++	--операнд операнд--	~	!	+ операнд – операнд
3	Створення об'єкта і перетворення типу	new	(тип) операнд			
4	Мультиплікативні	*	/	%		
5	Адитивні	+	–			
6	Зсув бітів	>>	>>>	<<		
7	Відношення	>	>=	<	<=	instanceof
8	Еквівалентності	==	!=			
9	Побітове І	&				
10	Побітове виключаюче АБО	^				
11	Побітове АБО	|				
12	Логічне І	&&				
13	Логічне АБО	||				
14	Умовне	? :				
15 *нижчий*	Присвоювання	=	Оператор = (+=, –=, *=, /= и т. п.)			

Символьний тип char

Символи в Java визначаються за допомогою ключового слова **char** і реалізовані з використанням стандарту Unicode. Можна задати константу-символ в програмі як звичайний символ. Символьне значення повинні бути укладено в пару одиночних апострофів.

```
char symbol1='f';
```

Інший спосіб запису символів: пара символів "**\u**", за якою слідує чотиризначне шістнадцяткове число (в діапазоні від **0000** до **FFFF**), що представляє собою **код символу в Unicode**.

```
char symbol1 = '\u0042';
```

Також символьній змінній можна привласнювати числовий код символу Unicode (номер символу в кодової таблиці)

```
char c1='a';
char c2='\u0061';
char c3=97;
```

Дійсні типи

Тип	Розмір	Діапазон	Опис
`float`	4 байта (32 біта)	$1.5 \cdot 10^{-45}..3.4 \cdot 10^{38}$	Одинарна точність, 7-8 значущих цифр мантиси Тип `real*4` стандарта IEEE754
`double`	8 байт (64 біта)	$5 \cdot 10^{-324}..1.7 \cdot 10^{308}$	Подвійна точність, 15-16 значущих цифр мантиси Тип `real*8` стандарта IEEE754

Для чисел з плаваючою точкою потрібно вказувати цілу і дробову частину, розділені крапкою (4.6, 7.0 і т.п.).

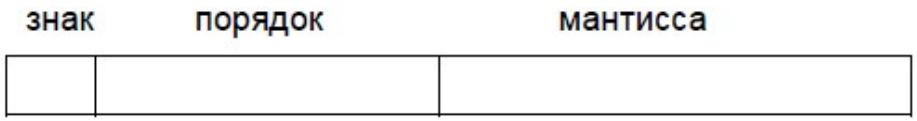
Для великих чисел можна використовувати **експонентну форму запису** (для відділення мантиси від порядку використовується символ "e" або символ "E"), наприклад, число $-3{,}58 \cdot 10^{7}$ записується як -3.58E7, а число $73{,}675 \cdot 10^{-15}$ - як 73.675e-15.

Двійкове подання дійсних чисел

Число x з плаваючою точкою можна представити у вигляді:

$$x = s \cdot m \cdot 2^p.$$

s — знак числа,
m — мантиса,
p — порядок числа.



Якщо знаковий біт дорівнює 0, то число позитивне, якщо 1 - негативне.

Набір бітів, що зберігається в мантисі, задає позитивне число ***m***, що лежить в межах **1≤ *m* < 2.**

При перемножуванні чисел їх мантиси перемножуються, а порядки складаються. При діленні мантиси діляться, а порядки віднімаються.

І множення, і ділення мантисс відбувається за тими ж алгоритмами, що і для цілих чисел. Але при виході за розміри комірки відкидаються не старші, а молодші байти.

Змінні з плаваючою точкою можуть зберігати не тільки чисельні значення, але і будь-який з особливо визначених прапорів (станів):

•негативна нескінченність;

•негативний нуль;

•позитивна нескінченність;

•позитивний нуль;

•«відсутність числа» (not-a-number, **NaN).**

Оператори для роботи з речовими величинами аналогічні операторам для роботи з цілочисельними величинами

У класі java.lang.Math задані константи π = 3,14... (Math.PI) і e = 2,71... (Math.E), а також математичні функції.

Функції, задані в класі Math

Функція	Примітка
Тригонометричні та зворотні тригонометричні функції	
sin(x)	sin(x) - синус
cos(x)	cos(x) - косинус
tan(x)	tg(x) - тангенс
asin(x)	arcsin(x) - арксинус
acos(x)	arccos(x) - арккосинус
atan(x)	arctg(x) - арктангенс
доRadians(angdeg)	angdeg / 180.0·PI; - Переведення кутів з градусів в радіани
toDegrees(angrad)	angrad·180.0 / PI; - Переведення кутів з радіан в градуси

Випадкове число, залишок	
random()	Псевдовипадкове число в діапазоні від 0,0 до 1,0. При цьому: 0 <= Math.random() < 1
IEEEremainder(x,y)	Похибка цілого чисельного поділу x на y, тобто різниця: x - y * n, де n - результат цілісного поділу

Функції, задані в класі Math

Ступені, експоненти, логарифми	
exp(x)	e^x - експонента
expm1(x)	$e^x - 1$. При x, близькому до 0, дає набагато точніші значення, ніж exp(x) –1
log(x)	ln(x) - натуральний логарифм
log10(x)	$\log_{10}(x)$ - десятковий логарифм
log1p(x)	ln(1+x). При x, близькому до 0, дає набагато точніші значення, ніж log(1+x)
sqrt(x)	квадратний корінь
cbrt(x)	кубічний корінь
hypot(x,y)	обчислення довжини гіпотенузи за двома катетами
pow(x, y)	x^y - зведення x вступінь y
sinh(x)	гіперболічний синус
cosh (x)	гіперболічний косинус
tanh(x)	гіперболічний тангенс

Функції, задані в класі Math

Модуль, знак, мінімальна, максимальна кількість	
abs(m), abs(x)	Абсолютне значення числа. Аргумент типу int, long, float або double. Результат того ж типу, що аргумент
signum(a), signum(x)	Знак числа. Аргумент типу float або double. Результат того ж типу, що аргумент
min(m,n), min(x,y)	Мінімальне із двох чисел. Аргументи одного типу. Можливі типи: int, long, float, double. Результат того ж типу, що аргумент
max(m,n), max(x,y)	Максимальне із двох чисел. Аргументи одного типу. Можливі типи: int, long, float, double. Результат того ж типу, що аргумент

Округлення	
ceil(x)	Найближче до x ціле, більше або рівне x
floor(x)	Найближче до x ціле, менше або рівне x
round(a), **round(x)**	Найближче до x ціле. Аргумент типу float або double. Результат типу long, якщо аргумент double і типу int — якщо float. Те саме, що (int)floor(x + 0.5)
rint(x)	Найближче до x ціле
ulp(a), ulp(x)	Відстань до найближчого більшого, ніж аргумент значення того ж типу (дискретність зміни чисел у форматі з плаваючою точкою поблизу даного значення). Аргумент типу float або double. Результат того ж типу, що аргумент

Приведення типів при виконанні операцій

```
double y;
byte x;
y = x + 5;
```

При виконанні операції присвоювання перетворення типів відбувається автоматично, якщо відбувається **розширююче перетворення** (***widening conversion***) і **два типи сумісні**.

byte→short→int→long→float→double

Числові типи не сумісні з типами char и boolean.

Якщо необхідно виконати **звужуюче перетворення**, використовується операція **приведення** (cast) типу, яка має такий вигляд:

```
(тип-перетворення) значення
```

```
byte x = 71;
char symbol = (char) x;
```

Якщо значення з плаваючою точкою присвоюється цілого типу, то при явному перетворенні типу відбувається також усічення (truncation) числа.

При виконанні арифметичних і побітових перетворень всі значення **byte** і **short**, а також **char** розширюються до **int**, потім, якщо хоча б один операнд має тип **long**, тип цілого виразу розширюється до **long**. Якщо один з операндів має тип **float**, то тип повного вираження розширюється до **float**, а якщо один з операндів має тип **double**, то тип результату буде **double**.

Автоматичні розширення типів (особливо розширення short і byte до int) можуть викликати помилки, що погано розпізнаються під час компіляції.

```
byte x = 30, y =5;
x = x + y;
```

Щоб цього уникнути треба використовувати в другому операторі явне перетворення типів

```
byte x = 30, y =5;
x = (byte) (x + y);
```

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.