Оператори організації робочої програми.

Масиви в Java

Масивом називається іменований набір змінних однакового типу. Кожен елемент масиву є змінною, яка однозначно визначається шляхом вказівки **імені масиву** і цілочисельнної позиції елементу в масиві - **індексу елемента в масиві**.

Масиви в Java є гібридом об'єктів і примітивних типів.

Опис масиву проводиться в три етапи

На **першому етапі** виконується **оголошення масиву** (***array declaration***).

ім'я-типу [ ] ідентифікатор-масиву

або

ім'я-типу ідентифікатор-масиву[ ]

ім'я-типу - ім'я примітивного або посилального типу
ідентифікатор-масиву - ім'я, що привласнюється масиву.

```
int [ ] myArray;
```

```
int myArray[ ];
```

Якщо кілька масивів мають однаковий тип, їх також можна оголосити в списку, використовуючи перший формат запису

```
short [ ] a, b;
```

На **другому етапі**, етапі **визначення** або **створення масиву** (***array instantiation***) вказується кількість елементів масиву, зване його **довжиною**, виділяється місце для масиву в оперативній пам'яті, змінна-посилання отримує адресу масиву.

```
new ім'я-типу [розмір-масиву]
```

розмір-масиву - змінна або вираз будь-якого цілочисельного типу, за винятком типу **long**

```
myArray = new int[5];
byte dim1 = 10, dim2 = 2;
a = new short[dim1];
b = new short[dim1 + dim2];
```

На **третьому етапі** проводиться **ініціалізація масиву** (***array initialization***). Елементи масивів числових типів отримують нульове значення відповідного типу, елементи булевских масивів - значення false, а елементи масивів посилальних типів - значення null.

Масиви можуть бути ініційовані під час створення. При цьому ***розмір-масиву*** в квадратних дужках не вказується, а після закриваючої квадратної дужки в фігурних дужках задається список значень елементів масиву, розділених комами.

```
myArray = new int[ ]{0, 1, 2, 3, 4};
```

Можна об'єднати перший і другий етапи в одному операторі

```
int [ ] myArray = new int[5];
```

Оскільки при ініціалізації масиву розмір масиву вказувати не треба, можна об'єднати перший і третій етап в одному операторі

```
int [ ] myArray = {0, 1, 2, 3, 4};
```

Доступ до елементів масиву

Елемент масиву, крім значення, характеризується своєю позицією в масиві - **індексом**. Індексом елемента є значення типу **byte**, **short**, **int** або **char**, укладену в квадратні дужки, наприклад **myArray[4]**.

Індекс елемента масиву змінюється від нуля до величини, на одиницю меншою розміру масиву

```
int myArrayLength = myArray.length;
```

Багатовимірні масиви

У масивів в Java повинна бути вказана принаймні одна розмірність, проте в програмах можуть використовуватися і багатовимірні масиви, причому інші розмірності можна визначати під час виконання програми.

Елементами масивів в Java можуть бути інші масиви.

```
int [ ][ ] matrix = new int[3][4];
```

Перший (лівий) розмір масиву повинен задаватися при його створенні. Інші розміри можуть зазначатися пізніше.

Таким чином, можна створювати непрямокутної масиви

Приклад

```
int [ ] [ ] triangleMatrix = new int [3] [ ] ;
triangleMatrix[0] = new int [1];
triangleMatrix[0][0] = 1;
triangleMatrix[1] = new int [2];
triangleMatrix[1][0] = 2;
triangleMatrix[1][1] = 3;
triangleMatrix[2] = new int [3];
triangleMatrix[2][0] = 4;
triangleMatrix[2][1] = 5;
triangleMatrix[2][2] = 6;
```

Матриця **triangleMatrix** буде трикутною, і буде мати наступний вигляд:

1
2 3
4 5 6

Змінні лічильного типу

Визначаються за допомогою ключового слова **enum.**

Оголошення лічильної змінної

```
модифікатори-класу enum ідентифікатор-класу {
        ім'я-константи-1, …, ім'я-константи-n
}
```

список імен констант задає значення, які може приймати змінна ***ідентифікатор-класу***

Пример

```
public enum Month {
JANUARY, FEBRUARY, MARCH, APRIL,MAY, JUNE, JULY, AUGUST,
SEPTEMBER, OCTOBER, NOVEMBER, DECEMBER
}
```

```
Month currentMonth;
currentMonth = Month. SEPTEMBER;
```

Оголошення лічильної змінної є оголошенням класу і може мати більш складний вид

```
модифікатори-класу enum ідентифікатор-класу {
ім'я-константи-1 (список-аргументів), ...,
ім'я-константи-n (список-аргументів);
тіло-класу
}
```

Список-аргументів задає відокремлювані один від одного комами аргументи, які передаються конструктору класу при створенні об'єкта даного класу. У тілі класу, крім конструктора, можуть бути визначені й інші методи для роботи зі змінними об'єкта.

Приклад

```
public enum Month {
JANUARY(1,31), FEBRUARY(2,28), MARCH(3,31), APRIL(4,30), MAY(5,31), JUNE(6,30),
JULY(7,31), AUGUST(8,31), SEPTEMBER(9,30), OCTOBER(10,31), NOVEMBER(11,30),
DECEMBER(12,31);
        private int monthIndex;
        private int numberOfDaysInMonth;
        Month(int monthIndex, int numberOfDaysInMonth){
                this.monthIndex = monthIndex;
                this.numberOfDaysInMonth = numberOfDaysInMonth;
        }
        int getMonthIndex()
        {
                return monthIndex;
        }
int getNumberOfDaysInMonth() {
                return numberOfDaysInMonth;
        }
}
```

Використання змінної в іншому класі:

```
Month currentMonth;
currentMonth = Month. SEPTEMBER;
int currentMonthIndex = currentMonth.getMonthIndex();
int currentNumberOfDaysInMonth =
currentMonth.getNumberOfDaysInMonth();
```

Внаслідок виконання цього фрагмента програми змінній **currentMonthIndex** буде присвоєно значення **9**, а змінній **currentNumberOfDaysInMonth** - значення **30**.

Оператори управління Java

Умовний вираз ...?... : ...

```
умова?значення1:значення2
```

Коли ***умова*** має значення **true**, функція повертає ***значення1***, в іншому випадку - ***значення2***.

У виразах для умов і значень можна використовувати пробіли, в тому числі відокремлювати ними символи "**?** " і "**:**" .

```
j=i<5?i+1:i+2
```

або

```
j=(i<5)?(i+1):(i+2)
```

Складений оператор

Згідно синтаксису мови Java в багатьох конструкціях можливий тільки один оператор, але часто зустрічається ситуація, коли необхідна послідовність з декількох операторів.
Тоді використовується складений оператор - блок коду між фігурними дужками { }.

```
оператор
{
        послідовність операторів
}
```

або

```
оператор{
        послідовність операторів
}
```

Другий спосіб встановлено за умовчанням в середовищі NetBeans.

Оператори передачі управління в Java

При роботі програми оператори виконуються послідовно. Однак часто буває необхідним змінити порядок виконання операторів. Для цього використовуються оператори управління програмою.

- **умовний оператор;**
- **оператори циклу;**
- **оператори переходу;**
- **оператор вибору.**

Умовний оператор

```
if (булевий-вираз)
оператори-1;
else
оператори-2;
```

Якщо значення ***булевого-виразу*** дорівнює **true**, то виконуються ***оператори-1***, інакше виконуватися ***оператори-2***. Оператор **else** може бути опущений, в цьому випадку, якщо ***булевий-вираз*** дорівнює **false**, виконується оператор, наступний за оператором **if**.
Якщо ***оператори-1*** або ***оператори-2*** містять кілька операторів, вони повинні бути укладені у фігурні дужки.

```
if (x == 0)
      y = 0;
else {
      y = 1;
      z = x + 1;
}
```

Приклад 1

```
if(a<b)
        a=a+1;
else
        if(a==b)
                a=a+1;
        else {
                a=a+1;
                b=b+1;
        }
```

Приклад 2

```
if(умова)
        оператор1;
        оператор2;
else
        оператор3;
```

```
if(умова) {
        оператор1;
        оператор2;
} else
оператор3;
```

Приклад 3

```
if(умова)
        оператор1;
else
        оператор2;
оператор3;
```

```
if(умова)
        оператор1;
else {
        оператор2;
        оператор3;
}
```

Оператори циклу

- **оператор for;**
- **розширений оператор for;**
- **оператор while;**
- **оператор do…while.**

Оператор циклу for

for (*ініціалізація-циклу; контрольний-вираз; кроковий-вираз*)
тіло-циклу

Ініціалізація-циклу виконується один раз перед першим проходом тіла циклу

Контрольний-вираз обчислюється перед початком кожного виконання тіла циклу

Кроковий-вираз обчислюється в кінці кожного виконання тіла циклу

Тіло циклу може містити один оператор або кілька операторів, укладених у фігурні дужки.

```
int a[ ] = {1, 2, 3, 4, 5};
int sum = 0;
for (int i = 0; i < a.length; i++)
sum+=a[i];
```

У вираз для ініціалізації циклу і в вираз для ітерації циклу можна включити **кілька операторів, розділених комами**. Зазвичай така форма застосовується, коли в циклі необхідно використовувати кілька змінних циклу

```
boolean isSymmetric = true;
…
for(int i=0, j=a.length-1; i < j; i++, j--)
if(a[i] != a[j])
isSymmetric = false;
```

Розширений оператор for

```
for(модифікатори-змінної тип
ідентифікатор-змінної: вираз)
тіло-циклу
```

Вираз в цьому циклі має задавати набір значень змінної. Якщо набір значень є масивом, в якості вираження задається ім'я масиву. Набір елементів змінної лічильного типу і колекцій можна отримати за допомогою статичного методу **values()**. В процесі виконання циклу змінній з ім'ям ***ідентифікатор-змінної*** будуть послідовно присвоюватися значення всіх елементів масиву, змінної перечислимого типу або колекції.

```
int a[ ] = {1, 2, 3, 4, 5};
int sum = 0;
for (int ai : a)
sum+=ai;
```

```
int dayNumber = 0;
for (Month monthN : Month.values())
dayNumber+=
monthN.getNumberOfDaysInMonth();
```

Оператори циклу while і do…while

```
while(булевий-вираз)
тіло-циклу
```

```
do
тіло-циклу
while(булевий-вираз);
```

Обидва циклу виконуються доти, поки ***булевий-вираз*** має значення **true**, проте в циклі **while** обчислення ***булевого-виразу*** проводиться до початку чергового виконання циклу, а в циклі **do ... while** - після його чергового виконання.

```
int a[ ] = {1, 2, 3, 4, 5};
int sum = 0, i = 0;
while (i < 0)
{
        sum+=a[i];
        i ++;
}
```

```
int a[ ] = {1, 2, 3, 4, 5};
int sum = 0, i = 0;
do
{
        sum+=a[i];
        i ++;
}
while (i < 0);
```

Оператори переходу

В Java відсутній оператор **goto** *мітка*

Оператор переривання

break *мітка*;

передає управління за межі циклу або оператора вибору, позначеного зазначеної ***міткою***. ***Мітка*** являє собою звичайний ідентифікатор Java, за яким слідує двокрапка. ***Мітка*** може бути опущена - в цьому випадку управління передається за межі циклу або оператора вибору, що містить даний оператор **break**.

```
int a[ ] = {1, 2, 3, 0, 5};
int i, zeroIndex = -1;
for (i = 0; i < a.length; i++)
if(a[i] == 0)
break;
if (i < a.length)
zeroIndex = i;
```

- використання оператора **break** без мітки

```
int a[ ][ ] = {{1,2,3},{5,6,7},{8,9,0},{11,12,13}};
int i, j;
…
iLoop:
for(i = 0; i < 4; i++)
{
        for(j = 0; j < 3; j++)
        {
                if (a[i][j] == 0)
                        break iLoop;
        }
}
```

- використання оператора **break** з міткою

Оператор продовження

continue *мітка*;

Передає управління всередині циклу.
Якщо ***мітка*** опущена, оператор **continue** передає управління на самий кінець тіла циклу і, якщо контрольний вираз, обчислений після того, як черговий прохід тіла циклу був завершений оператором **continue**, дорівнюватиме **true**, виконання циклу продовжиться.

Це буває корисно для того, щоб пропустити при виконанні тіла циклу деякі з операторів.
Якщо ***мітка*** вказана, оператори пропускаються в циклі, позначеному зазначеної міткою.

```
int a[ ] = {1, 2, 3, 4, 5};
int sum = 0;
for (int i = 0; i < a.length; i++)
{
if(a[i]%2 != 0)
continue;
sum += a[i];
}
```

Оператор вибору

Оператор вибору **switch** зазвичай використовується, коли необхідно організувати розгалуження програми за кількома напрямками, однак умови перевірки в ньому повинні бути виражені за допомогою різних значень цілої змінної.

```
switch (вираз)
{
case значення-1: оператори-1;
case значення-2: оператори-2;
…
default: оператори-n;
}
```

Вираз, який ставиться в круглих дужках після ключового слова **switch**, може належати до одного з типів **char**, **byte**, **short** або **int**, а також одним із значень змінної перечислимого типу.

Перевірка значення ***виразу*** на рівність заданому константному ***значенню*** здійснюється за допомогою операторів **case**, що входять в оператор **switch**.
Всі оператори **case** повинні містити різні значення.
Як тільки один з операторів **case** упізнав значення ***виразу***, управління отримують оператори, що йдуть після символу ":" за цим оператором **case**.

Після цього виконуються оператори, що містяться у всіх наступних операторах **case** і в операторі **default**. Для того, щоб обійти виконання наступних операторів **case** і оператора **default**, слід задати останнім оператором для даного **case** оператор **break**. Якщо ***значення виразу*** не збігається ні з одним зі значень, виконуються оператори, що йдуть за оператором **default** (цей оператор може бути опущений).

Оператори, що йдуть за двокрапкою, можна не укладати в фігурні дужки.

Якщо для декількох ***значень*** виконуються одні й ті ж дії, оператори **case** можна об'єднати.

```
case значення-1:
case значення-2: оператори;
```

```
char s;
int x;
switch(s) {
case 'a':
        x = 0;
        break;
case 'A':
        x = 1;
        break;
case ' ':
case '(':
case ')':
        x = 2;
        break;
default:
        x = 3;
}
```

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.