Тема 9 - Масиви

16 юни 2018 г. 22:20

Масивите arrays представляват група от подобни на едни на други променливи за чийто достъп се използва една променлива. Масивите могат да бъда едномерни или многомерни. За достъпа до определен елемент от масива се използва неговият индекс.

```
arr[5]; // достъп до 6тият елемент на масива (индексирането на масивите в Java започва от нула)
```

Едномерни масиви

Едномерен масив е просто списък, който групира подобни променливи.
За да обявите масив първо трябва да обявите псевдоним или препратка (reference variable) за него:

```
//Обща форма
type [] var-name
// Код
int [] monthDays;
// Това обявява препратка(reference) от тип "масив от цели числа(array of int)"
```

Това обявяване създава само променлива, която може да сочи към масив от цели числа, но все още не съществува такъв масив. За да създадете такъв масив трябва да използвате оператора **new** и да обявите размера на масива**.**

```
//Създаване на обект от тип масив от цели числа и присвояването му на променливата monthDays
monthDays = new int[12];
```

След това вече monthDays сочи към масив с размер 12 , а елементите първоначално се установяват в 0 (за тип int).

```
// Файл: MonthDays.java
// Демонстрация на едномерен масив
class MonthDays {
    public static void main(String [] args) {
        int [] monthDays; // променлива която може да сочи към масив
        monthDays = new int[12]; // масив с размер 12 с тип int
        monthDays[0] = 31;
        monthDays[1] = 28;
        monthDays[2] = 31;
        monthDays[3] = 30;
        monthDays[4] = 31;
        monthDays[5] = 30;
        monthDays[6] = 31;
        monthDays[7] = 31;
        monthDays[8] = 30;
        monthDays[9] = 31;
        monthDays[10] = 30;
        monthDays[11] = 31; // 11 е последният индекс понеже започваме от 0
        System.out.println("April ima " +monthDays[3] + " dena.");
    }
}
```

Обикновено обявяването на променлива на масив и създаването му се пишат на един и същи ред:

```
int [] monthDays = new int[12];
```

Елементите на масива могат да се зададат и при обявяването му:

```
int [] monthDays = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

Когато се опитвате да присвоите стойност на, елемент от масива Java изрично следи дали тази стойност съвпада с типа обявен от масива или тип, който наследява обявения.

```
// Файл: Average.java
// Изчисляване на средното аритметично на масив
class Average {
    public static void main(String[] args) {
        double numbers[] = {10.1, 11.2, 12.3, 13.4, 14.5};
        double average = 0;
        for(int i = 0; i < 5; i++) {
            average = average + numbers[i];
        }
        System.out.println("Srednoto aritmetichno e " + average/5);
    }
}
```

Подобрена версия на програмата може да се направи като вместо 5 се използва дължината на масива (неговия размер) като се използва променлива, която се казва **length** и тя е обявена в класа масив. И като се използва съкращението за присвояване на число/низ със себе си плюс някаква друга променлива от същият тип **+=** (има съкращение и за другите операции **-=, /=, %=, *=**).

```
// Файл: AverageRefactored.java
class AverageRefactored {
    public static void main(String[] args) {
        double numbers[] = {10.1, 11.2, 12.3, 13.4, 14.5};
        double average = 0;
        for(int i = 0; i < numbers.length; i++) {
            average += numbers[i];
        }
        System.out.println("Srednoto aritmetichno e " +
        average/numbers.length);
    }
}
```

Многомерни масиви

Многомерните масиви в същност представляват масиви от масиви. В Java има някои неща за които трябва да се внимава при обявяването им. За да обявите многомерен масив се използват допълнителни квадратни скоби за всяко измерение, искате да добавите.

```
// Обявява променлива от тип двумерен масив от целочислени числа и го
инициализира с 4 масива, които ще съдържат по 5 елемента.
int [] [] twoDim = new int [4] [5];
```

```
// Файл: TwoDArray.java
// Програма, която създава двумерен масив и номерира елементите му от ляво надясно от горе надолу и след това извежда масива с отделен метод.
class TwoDArray {
    public static void main(String [] args) {
        int [] [] twoDim = new int[4][5];
        int currentNumber = 0;

        for(int i = 0; i < 4; i++) {
            for(int j = 0; j < 5; j++ ) {
                twoDim[i][j] = currentNumber++;
            }
        }
        printTwoDimArr(twoDim);
    }

    private static void printTwoDimArr(int [] [] arr){
        for(int i = 0; i < 4; i++) {
            for(int j = 0; j < 5; j++ ) {
                System.out.print(arr[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

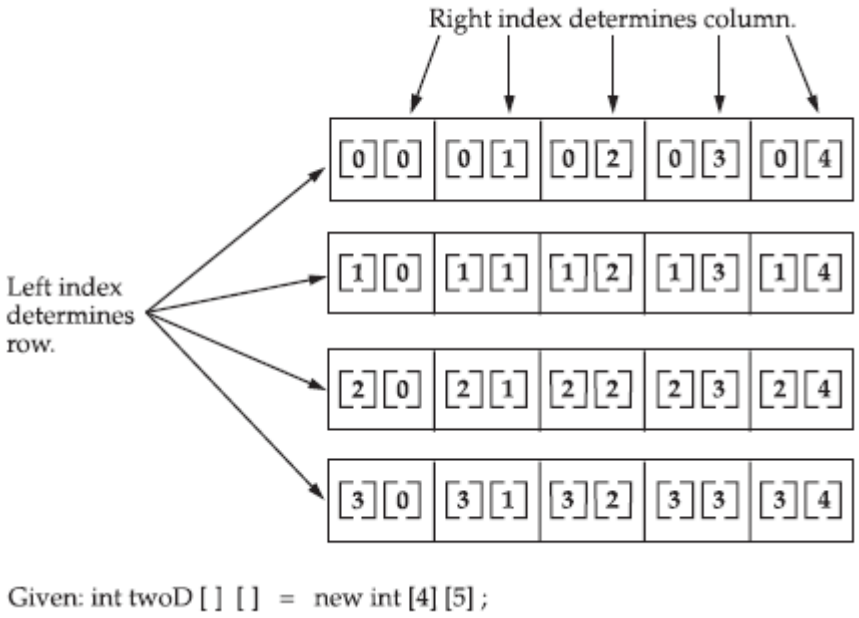
Програмата може да се промени, като не се разчита, че размерът винаги е един и същ и се използват отново променливата в типа масив length:

```
// Файл: TwoDArrayRefactored.java
class TwoDArrayRefactored {
    public static void main(String [] args) {
        int [] [] twoDim = new int[4][5];
        int currentNumber = 0;

        for(int i = 0; i < twoDim.length; i++) {
            for(int j = 0; j < twoDim[i].length; j++ ) {
                twoDim[i][j] = currentNumber++;
            }
        }
        printTwoDimArr(twoDim);
    }

    private static void printTwoDimArr(int [] [] arr){
        for(int i = 0; i < arr.length; i++) {
            for(int j = 0; j < arr[i].length; j++ ) {
                System.out.print(arr[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Структурата на този масив (с индексите за всяка една клетка) изглежда по следният начин:



Given: int twoD [ ] [ ] = new int [4] [5] ;

Понеже можем да искаме да имаме масив от масиви с различни дължини това може да се постигне по следният начин:

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

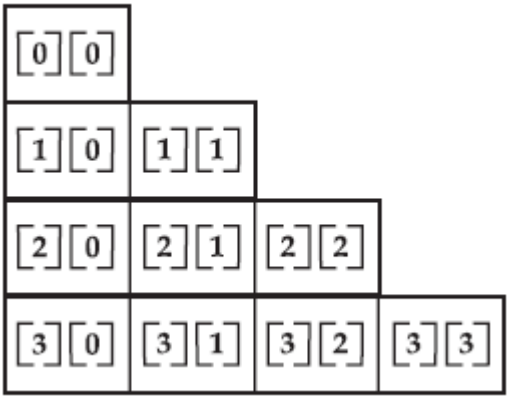
Сега можем да тестваме с помощта на предишната програма, като само променим обявяването на масива.

```
// Файл: TwoDArrayDiffSize.java
class TwoDArrayDiffSize {
    public static void main(String [] args) {
        int [] [] twoDim = new int[4][];
        twoDim[0] = new int[1];
        twoDim[1] = new int[2];
        twoDim[2] = new int[3];
        twoDim[3] = new int[4];
        int currentNumber = 0;

        for(int i = 0; i < twoDim.length; i++) {
            for(int j = 0; j < twoDim[i].length; j++ ) {
                twoDim[i][j] = currentNumber++;
            }
        }
        printTwoDimArr(twoDim);
    }

    private static void printTwoDimArr(int [] [] arr){
        for(int i = 0; i < arr.length; i++) {
            for(int j = 0; j < arr[i].length; j++ ) {
                System.out.print(arr[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Структурата на създаденият от програмата масив изглежда така:



Двумерен масив може да се зададе и по следният начин:

```
int [] [] arr = { {0,1,2,3}, {4,5,6}, {7,8}, {9} };
```

В Java както и в други езици има възможност за масиви с повече от две измерения (с три, четири), но рядко се използват понеже част от измеренията могат да заменят с някакъв клас и да се намали броя на измерения.

Задача 1:

Напишете програма, която създава три едномерни масива, два от които са еднакви по елементи а третия е различен.
Напишете метод:
`static boolean arrEqual(int [] firstArr, int [] secondArr)`
, който да разглежда всеки един от членовете на двата масива едновременно и ако някой от тях не съвпада(един с друг не са еднакви) връща false, ако цикъла завърши накрая да връща true.
Тествайте с трите масива и извеждайте отговора на метода в конзолата.(метода е статичен защото се използва от статичният контекст на main метода.)

Тази задача може да се подобри с помощта на методите от класа Arrays от стандартната библиотека. Разгледайте класа и вижте кои методи могат да ви помогнат за опростяването на програмата. **(Решението на задачата Файл:ArrayEqual.java и Файл:ArrayEqualRefactored.java)**

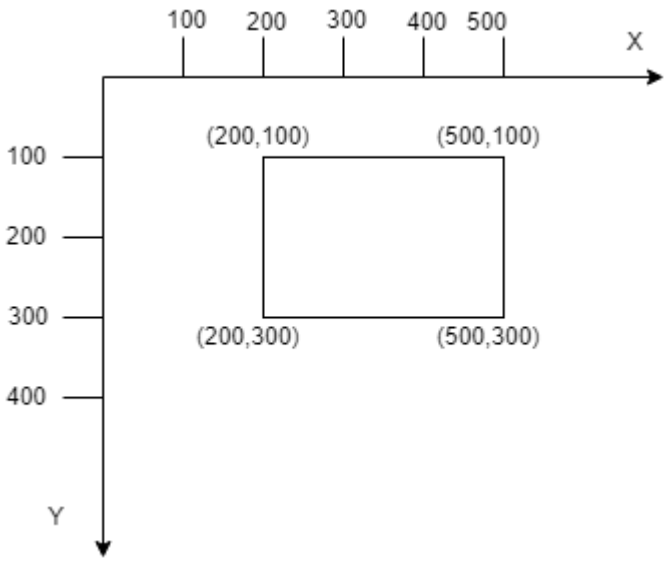
Задача 2:

Създайте клас Point, който съдържа следните полета и методи:
private int x;
private int y;
public Point(int x, int y) - конструктор
public int getX() - връща x полето
public int getY() - връща y полето
Създайте клас Rectangle, който използва масив за съхранение на четирите му точки от тип Point и има следните методи:
public Rectangle(Point leftTop, Point rightTop, Point leftBot, Point rightBot)
public double getArea()
public double getPerimeter()
//Помощни методи
public double getWidth() - използва calcDistance() на двете точки за изчисляване на широчина
public double getHeight() - използва calcDistance() на двете точки за изчисляване на височина
private double calcDistance(Point a, Point b) - пресмята разстоянието по формулата

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

(Нека приемем, че ще създавате класа с точки в една и съща последователност:
Лява горе, дясна горе, лява долу, дясна долу)
Представете си, че координатната система е обърната(както е на мониторите).

Примерен правоъгълник:



Тествайте програмата в клас съдържащ main метод и извежда на конзолата широчината, височината, периметъра и лицето на правоъгълника. **(Решението на задачата Файл:Point.java, Файл:Rectangle.java, Файл:RectangleTest.java)**

Допълнителни линкове към темата:
https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html
https://www.tutorialspoint.com/java/java_arrays.htm

Автор: Димитър Томов - xdtomov@gmail.com