Προγραμματισμός ΙΙ

1/4/2020

Ορισμός μονοδιάστατου (1Δ) πίνακα

Για να κατασκευάσουμε έναν 1Δ πίνακα 3 στοιχείων τύπου int:
int[] a=new int[3];

Ισοδύναμα:
int[] a; (δήλωση της μεταβλητής)
a=new int[3]; (δέσμευση μνήμης για 3 ακεραίους)

Για να δώσουμε τιμές:
α[0]=5;
α[1]=8;
α[2]=12;

Ορισμός μονοδιάστατου (1Δ) πίνακα

Εναλλακτικά, όλα τα παραπάνω πιο γρήγορα:
int[] a={5, 8, 12};

Οι πίνακες είναι αντικείμενα μιας υποθετικής κλάσης πινάκων.

Έχουν μια χρήσιμη μεταβλητή υπόστασης, τη **length**. Πχ η:
int x=a.length;
Αναθέτει στη μεταβλητή x την τιμή 3.

Πως διατρέχουμε στοιχεία 1Δ πίνακα

Για να διατρέξουμε όλα ή μέρος στοιχείων πίνακα χρησιμοποιούμε βρόχους, όπως είναι ο for

Πως διατρέχουμε στοιχεία 1Δ πίνακα

```
class Array1D
{
	public static void main (String args[])
	{
		//Create the array
		int[] a =new int[10];

		//Fill in the elements of the array
		for (int i=0;i<a.length;i++)
		{
			a[i]=i+1;
		}

		//Print all the elements of the array
		for (int i=0;i<a.length;i++)
		{
			System.out.println(a[i]);
		}
	}
}
```

Ορισμός δισδιάστατου (2Δ) ή πολυδιάστατου πίνακα

Για να κατασκευάσουμε τον 2Δ πίνακα:

$$\boldsymbol{\alpha} = \begin{bmatrix} \mathbf{5} & \mathbf{2} \\ \mathbf{3} & \mathbf{4} \\ \mathbf{1} & \mathbf{8} \end{bmatrix}$$

int[][] a; (δήλωση)
a=new int[3][2]; (δέσμευση μνήμης)

Για να δώσουμε τιμές:
α[0][0]=5;
α[0][1]=2;
…
α[2][1]=8;

Ορισμός δισδιάστατου (2Δ) ή πολυδιάστατου πίνακα

Εναλλακτικά:
int[][] a={ {5,2},{3,4},{1,8} };

Σαν ένας 1Δ πίνακας 3 στοιχείων, όπου κάθε στοιχείο είναι 1Δ πίνακας 2 στοιχείων

Η έκφραση **a[0]** αναπαριστά την πρώτη γραμμή του a,η **a[1]** τη δεύτερη γραμμή κλπ

Η έκφραση **a.length** δίνει το πλήθος γραμμών (εδώ 3) ενώ
οι **a[0].length; a[1].length; a[2].length;**
δίνουν το πλήθος στηλών

Ορισμός δισδιάστατου (2Δ) ή πολυδιάστατου πίνακα

Παράδειγμα κατασκευής 3Δ πίνακα:
int[][][]a=new int[4][3][2];

Πως διατρέχουμε στοιχεία 2Δ πίνακα

```
class Array2D
{
        public static void main (String args[])
        {
                //Create the array
                int[][] a =new int[3][2];

                //Fill in the elements of the array
                int c=1;
                for (int i=0;i<a.length;i++)
                {
                        for (int j=0;j<a[i].length;j++)
                        {
                                a[i][j]=c;
                                c++;
                        }
                }
                //Print all the elements of the array
                for (int i=0;i<a.length;i++)
                {
                        for (int j=0;j<a[i].length;j++)
                        {
                                System.out.print(a[i][j]+"\t");
                        }
                        System.out.println();
                }
        }
}
```

Ακανόνιστοι πίνακες

Είναι οι πίνακες στους οποίους κάθε γραμμή έχει διαφορετικό μήκος. Πχ ο πίνακας: $\alpha = \begin{bmatrix} 1 & & \\ 2 & 3 & \\ 4 & 5 & 6 \end{bmatrix}$

Ακανόνιστοι πίνακες

```
class Array2D
{
        public static void main (String args[])
        {
                //Create the array
                int[][] a =new int[3][];
                a[0]=new int[1];
                a[1]=new int[2];
                a[2]=new int[3];

                //Fill in the elements of the array
                int c=1;
                for (int i=0;i<a.length;i++)
                {
                        for (int j=0;j<a[i].length;j++)
                        {
                                a[i][j]=c;
                                c++;
                        }
                }

                //Print all the elements of the array
                for (int i=0;i<a.length;i++)
                {
                        for (int j=0;j<a[i].length;j++)
                        {
                                System.out.print(a[i][j]+"\t");
                        }
                        System.out.println();
                }
        }
}
```

Πίνακες αντικειμένων

Είναι πίνακες με στοιχεία αντικείμενα κλάσεων (αντί για πρωταρχικούς τύπους)

```
class Human
{
        String name;
        int height;

        Human (String n, int h)
        {
                name=n;
                height=h;
        }

        public String toString()
        {
                return name+" "+height;
        }
}
```

Πίνακες αντικειμένων

```
class HumanArray
{
        public static void main (String args[])
        {
                //Create the array
                Human[] a=new Human[3];

                //Create each object of the array
                a[0]=new Human("Nikos",20);
                a[1]=new Human("Anna",18);
                a[2]=new Human("John",25);

                //Print all the elements of the array
                for (int i=0;i<a.length;i++)
                {
                        System.out.println(a[i]);
                }
        }
}
```

Πίνακες ως ορίσματα μεθόδων

```
class ArrayInput
{
	static void fill(int[] a, int value)
	{
		//Fill in the elements of the array
		for (int i=0;i<a.length;i++)
		{
			a[i]=value;
		}
	}

	public static void main(String args[])
	{
		//Create the array
		int[] x=new int[10];

		//Call method fill to fill array with value 1
		fill(x,1);

		//Print all the elements of the array
		for (int i=0;i<x.length;i++)
		{
			System.out.println(x[i]);
		}
	}
}
```

Πίνακες ως ορίσματα μεθόδων

Οι τιμές του πίνακα τροποποιούνται, όχι μόνο όσο διαρκεί η εκτέλεση της μεθόδου αλλά και στο υπόλοιπο πρόγραμμα.

Δηλ. οι πίνακες περνούν ως ορίσματα **με αναφορά (by reference)**.

Το ίδιο συμβαίνει και σε οποιοδήποτε αντικείμενο όταν περνά ως όρισμα σε μια μέθοδο.

Δε συμβαίνει το ίδιο σε μεταβλητές πρωταρχικού τύπου (πχ int).

Δηλ. οι μεταβλητές πρωταρχικού τύπου περνούν ως ορίσματα **με τιμή (by value)**.

Πίνακες ως τύποι επιστροφής μεθόδων

Οι πίνακες επιστρέφονται από μεθόδους με τον ίδιο τρόπο που επιστρέφεται και οποιαδήποτε άλλη μεταβλητή πρωταρχικού τύπου.

Η διαφορά είναι ότι πριν επιστραφεί ο πίνακας πρέπει πρώτα να έχει δημιουργηθεί μέσα στη μέθοδο (με χρήση της new).

Πίνακες ως τύποι επιστροφής μεθόδων

```
class ArrayOutput
{
        static int[] fill(int numberOfElements, int value)
        {
                //Create the array
                int[] a=new int[numberOfElements];

                //Fill in the elements of the array
                for (int i=0;i<a.length;i++)
                {
                        a[i]=value;
                }
                //Return the array
                return a;
        }                                                       }
```

```
public static void main(String args[])
{
        //Declare the array
        int[] x;

        //Call method fill to fill array with value 1
        fill(x,1);

        //Print all the elements of the array
        for (int i=0;i<x.length;i++)
        {
                System.out.println(x[i]);
        }
}
```

Ορίσματα από τη γραμμή εντολών

Έστω ότι θέλουμε να μεταγλωττίσουμε το πρόγραμμα TestArguments.java από τη γραμμή εντολών:

>javac TestArguments.java

Για την εκτέλεσή του, αν δέχεται τρία ορίσματα:

>java TestArguments one two three

Τα one, two, three είναι ορίσματα που εισάγονται ως αλφαριθμητικά στο πρόγραμμα

Ορίσματα από τη γραμμή εντολών

```
class TestArguments
{
        public static void main(String[] args)
        {
                for (int i=0;i<args.length;i++)
                {
                        System.out.println(args[i]);
                }
        }
}
```

Σύνθετες κλάσεις και μέθοδοι

Κλάσεις με μεταβλητές υπόστασης τύπου κλάσεις

Στην κλάση Human, η μεταβλητή name είναι τύπου κλάσης (String). Δεν απαιτούσε κάποιο ιδιαίτερο χειρισμό, γιατί η δημιουργία ενός αντικειμένου τύπου String δεν απαιτεί τη new:

```
class Human
{
        String name;
        int height;

        Human (String n, int h)
        {
                name=n;
                height=h;
        }

        public String toString()
        {
                return name+" "+height;
        }
}
```

Κλάση ευθείας με μεταβλητές υπόστασης τύπου σημείου

```
class Point
{
        int x,y;

        Point(int x, int y)
        {
                this.x=x;
                this.y=y;
        }

        public String toString()
        {
                return “(” + x + “,” + y + “)”;
        }
}
```

Κλάση ευθείας με μεταβλητές υπόστασης τύπου σημείου

```
class StraightLine
{
        //Instance variables of class type Point
        private Point a,b;

        //Constructor based on coordinates
        public StraightLine(int x1,int y1,int x2,int y2)
        {
                this.a=new Point(x1,y1);
                this.b=new Point(x2,y2);
        }

        //Constructor based on point objects
        public Straight Line (Point a, Point b)
        {
                this.a=new Point(a.x,a.y)
                this.b=new Point(b.x,b.y)
        }
```

Κλάση ευθείας με μεταβλητές υπόστασης τύπου σημείου

```
//Calculate the length of the line
public double length()
{
        //Returns the Euclidean distance of the points
        double d1=a.x-b.x;
        double d2=a.y-b.y;
        return Math.sqrt(d1*d1+d2*d2);
}

//Accessor method – returns an independent point
public Point getPointA()
{
        Point first=new Point(a.x,a.y);
        return first;

        //if you use return a; the point will be dependent
}
```

Κλάση ευθείας με μεταβλητές υπόστασης τύπου σημείου

```
//Accessor method-returns an independent point
public Point getPointB()
{
        Point second=new Point(b.x,b.y);
        return second;
}

// Constructs a new Straightline with inverted points
public StraightLine inverted()
{
        StraightLine r=new StraightLine(b,a);
        return r;
}
```

Κλάση ευθείας με μεταβλητές υπόστασης τύπου σημείου

```
public String toString()
{
        return “Line from ” + a.toString() + “ to “ + b.toString();
}

public static void main(String[] args)
{
        //Construct a line from coordinates
        StraightLine e1=new StraightLine(1,5,8,2);

        //Construct a line from two point objects
        Point p1=new Point(2,3);
        Point p2=new Point(5,6);
        StraightLine e2=new StraightLine(p1,p2);
```

Κλάση ευθείας με μεταβλητές υπόστασης τύπου σημείου

```
        //Get the length of a line
        System.out.println("The first line has a length of " +e1.length()
        + " points");

        //Get the first point
        Point c=e1.getPointA();

        //Get an inverted straight line
        StraightLine e3=e2.inverted();
    }
}
```