

Instuderingsfrågor, del E

Flerdimensionella fält

Uppgift 1.

Skriv en en deklaration som skapar ett 2-dimensionellt heltalsfält med följande utseende:

```
1 3 6 10
2 5 9 13
4 8 12 15
7 11 14 16
```

Uppgift 2.

Deklarera ett fält för att lagra en avståndstabell mellan 10 orter.

Uppgift 3.

Nedanstående deklarationssatser resulterar i kompileringsfel. Varför?

- a) **double**[][] matris = **new double**(5)(4);
- b) **int**[][] temp= **new double**[3][9];
- c) **int**[][] speed = {{1.0, 3.0, 5.0 }, {7.0 , 9.0}};

Uppgift 4.

Följande deklaration har gjorts

```
char[][] letter = {{ 'a', 'b', 'c', 'd' }, { 'x', 'y', 'z' }};
```

Vilket värde har

- a) letter[0][1]
- b) letter[1][2]
- c) letter[0]

Uppgift 5.

Antag att du i ett program har följande deklarationer

```
int[][] vekt = {{3, 5, 8}, {2, 4, 9, 6, 3}, {9, 5}, {2, 4, 8, 2}};
```

Vad blir utskriften när nedanstående programsekvens exekveras?

```
System.out.println(vekt.length);
System.out.println(vekt[0].length);
System.out.println(vekt[1].length);
System.out.println(vekt[2].length);
```

Uppgift 6.

Betrakta nedanstående kodsegment

```
int[][] mat = new int[3][4];
for (int row = 0; row < mat.length; row++) {
    for (int col; col < mat[row].length; col++) {
        if (row < col)
            mat[row][col] = 1;
        else if (row == col)
            mat[row][col] = 2;
        else
            mat[row][col] = 3;
    }
}
```

Vilket innehåll har fältet mat?

Uppgift 7.

Vad blir utskriften av nedanstående programsatser?

```
import java.util.Arrays;
public class Test {
    public static void main(String[] args) {
        int[] oldArray = {1, 2, 3, 4, 5, 6, 7, 8, 9};
        int[][] newArray = new int[3][3];
        int row = 0;
        int col = 0;
        for (int index = 0; index < oldArray.length; index = index + 1) {
            newArray[row][col] = oldArray[index];
            row = row + 1;
            if ((row % 3) == 0) {
                col = col + 1;
                row = 0;
            }
        }
        for (int i = 0; i < newArray.length; i = i + 1)
            System.out.println(Arrays.toString(newArray[i]));
    } //main
} //Test
```

Uppgift 8.

Vad blir utskriften av nedanstående kodsegment?

```
int a2d[][] = {{10, 20, 30, 40, 50},
               {90, 80, 70, 60, 50},
               {60, 10, 90, 70, 30}};
for (int i = 0; i <= 4; i = i+1) {
    int tmp = a2d[0][i];
    for (int j = 1; j <= 2; j = j+1) {
        if (a2d[j][i] > tmp) {
            System.out.println(j);
            tmp = a2d[j][i];
        }
    }
    System.out.println(tmp);
}
```

Uppgift 9.

Skriv en metod

```
public static boolean exist(int[][] m)
```

som tar ett 2-dimensionellt heltalsfält **m** och returnerar värdet **false** om fältet **m** inte existerar (dvs har värdet **null**) eller innehåller 0 rader, annars returneras värdet **true**.

Uppgift 10.

Skriv en metod

```
public static int longestRow(int[][] m)
```

som tar ett 2-dimensionellt heltalsfält **m** och returnerar längden av den längsta raden i fältet **m**. Du får anta att fältet **m** existerar, dvs inte är **null**.

Exempel:

Antag att följande deklaration har gjorts

```
int[][] matrix = {{2, 2},
                  {4, 4, 4, 4},
                  {3, 3, 3}};
```

Ett anrop av `longestRow(matrix)` returnerar värdet 4.

Uppgift 11.

Skriv en metod

```
public static int max(int[][] m)
```

som tar ett heltalsfält `m` och returnerar värdet av det största element i `m`.

Uppgift 12.

En *enhetsmatrix* eller *identitetsmatrix* är en matrix vars element är 1 i diagonalen och 0 i övrigt. En enhetsmatrix kan ha godtycklig storlek men är alltid kvadratisk, dvs. den har lika många kolumner som rader. Figurerna nedan visar en två enhetsmatriser av storleken 3 respektive 4.

1	0	0
0	1	0
0	0	1

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Du skall skriva en metod

```
public double[][] identityMatrix(int size);
```

som skapar och returnerar en enhetsmatrix av storleken `size`.

Uppgift 13.

Skriv en metod

```
public static int largestSumOfRow(int[][] m)
```

som tar ett 2-dimensionellt heltalsfält `m` och returnerar den största radsumman i fältet `m`. Du får anta att fältet `m` existerar, dvs inte är **null**, samt att `m` innehåller minst en rad och att ingen rad är tom, dvs alla rader har minst ett element.

Exempel:

Antag att följande deklaration har gjorts

```
int[][] matrix = {{1, 2, 3, 4, -9},  
                 {2, 3, 1, 3, 1},  
                 {5, 5, 5, 6, 5}};
```

Ett anrop av `largestSumOfRow(matrix)` returnerar värdet 26.

Uppgift 14.

Skriv en metod

```
public static int indexOfLargestRow(int[][] m)
```

som tar ett 2-dimensionellt heltalsfält `m` och returnerar index för den rad i fältet `m` som har störst radsumma.

Exempel:

Antag att följande deklaration har gjorts

```
int[][] matrix = {{1, 2, 3, 4, -9},  
                 {2, 3, 1, 3, 1},  
                 {5, 5, 5, 6, 5}};
```

Ett anrop av `indexOfLargestRow(matrix)` returnerar värdet 2.

Uppgift 15.

Skriv en metod

```
public static void largestRowFirst(int[][] arr)
```

som tar ett 2-dimensionellt heltalsfält **arr** och förändrar fältet **arr** på så sätt att den rad som har största radsumman byter plats med den första raden.

Exempel:

Antag att följande deklaration har gjorts

```
int[][] matrix = {{1, 2, 3, 4, - 9},  
                 {2, 3, 1, 3, 1},  
                 {5, 5, 5, 6, 5}};
```

Ett anrop av **largestRowFirst(matrix)** innebär att **matrix** får utseendet

```
{{5, 5, 5, 6, 5},  
 {2, 3, 1, 3, 1},  
 {1, 2, 3, 4, - 9}}
```

Uppgift 16.

Skriv en metod

```
public static int largestSumOfColumn(int[][] m)
```

som tar ett 2-dimensionellt heltalsfält **m** och returnerar den största kolumnsumman i fältet **m**. Du får anta att alla rader i fältet är lika långa.

Exempel:

Antag att följande deklaration har gjorts

```
int[][] matrix = {{1, 2, 3, 4, - 9},  
                 {2, 3, 1, 3, 1},  
                 {5, 5, 5, 6, 5}};
```

Ett anrop av **largestSumOfColumn(matrix)** returnerar värdet 13.

Uppgift 17.

Skriv en metod

```
public static int min(int[][] m)
```

som tar ett heltalsfält **m** och returnerar värdet av det största element i **m**.

Uppgift 18.

Skriv en metod

```
public static int indexOfLargestColumn(int[][] m)
```

som tar ett 2-dimensionellt heltalsfält **m** och returnerar index för den kolumn i fältet **m** som har störst kolumnsumma.

Exempel:

Antag att följande deklaration har gjorts

```
int[][] matrix = {{1, 2, 3, 4, - 9},  
                 {2, 3, 1, 3, 1},  
                 {5, 5, 5, 6, 5}};
```

Ett anrop av **indexOfLargestColumn(matrix)** returnerar värdet 3.

Uppgift 19.

Skriv en metod

```
public static void largestColumnFirst(int[][] arr)
```

som tar ett 2-dimensionellt heltalsfält **arr** och förändrar fältet **arr** på så sätt att den kolumn som har största kolumnsumman byter plats med den första kolumnen.

Exempel:

Antag att följande deklaration har gjorts

```
int[][] matrix = {{1, 2, 3, 4, - 9},  
                  {2, 3, 1, 3, 1},  
                  {5, 5, 5, 6, 5}};
```

Ett anrop av **largestColumnFirst(matrix)** innebär att **matrix** får utseendet

```
{{4, 2, 3, 1, - 9},  
 {3, 3, 1, 2, 1},  
 {6, 5, 5, 5, 5}}
```

Uppgift 20.

Vad skrivs ut av nedanstående kodavsnitt?

```
List<String> list = new ArrayList<String>();  
list.add("A");  
list.add("B");  
list.add("C");  
list.add("D");  
for (String str : list) {  
    System.out.println(str);  
}
```

Uppgift 21.

Vad skrivs ut av nedanstående kodavsnitt?

```
ArrayList<Integer> list1 = new ArrayList<Integer>();  
ArrayList<Integer> list2 = new ArrayList<Integer>();  
list1.add(1);  
list1.add(2);  
list1.add(5);  
list2.addAll(list1);  
list1.addAll(list2);  
System.out.println(list1);
```

Uppgift 22.

Vad skrivs ut av nedanstående kodavsnitt?

```
ArrayList<String> list = new ArrayList<String>();  
list.add("red");  
list.add("green");  
list.add("red");  
list.add("black");  
for (int i = 0; i < list.size(); i = i + 1)  
    if (list.get(i).equals("red")) {  
        list.remove(element);  
        i = i - 1;  
    }  
System.out.println(list);
```

Uppgift 23.

Vad skrivs ut av nedanstående kodavsnitt?

```
ArrayList<String> list = new ArrayList<String>();
list.add("red");
list.add("green");
list.add("red");
list.add("black");
for (int i = 0; i < list.size(); i = i + 1)
    if (list.get(i).equals("red")) {
        list.set(i, "yellow");
        i = i - 1;
    }
System.out.println(list);
```

Uppgift 24.

Betrakta nedanstående metod:

```
public static ArrayList<Integer> mystery(int n) {
    ArrayList<Integer> list = new ArrayList<Integer>();
    for (int i = 1; i <= n; i = i + 1) {
        list.add(new Integer(i * i + 3));
    }
    return list;
} //mystery
```

Vad blir utskriften av nedanstående sats? planerats

```
System.out.println(mystery(6));
```

Uppgift 25.

Vad är fel med nedanstående kodsegment?

- a) ArrayList<int> values = new ArrayList<int>();
- b) ArrayList<Integer> values = new ArrayList();
- c) ArrayList<Integer> values = new ArrayList<Integer>;
- d) ArrayList<Integer> values = new ArrayList<Integer>();
for (int i = 1; i <= 10; i = i + 1) {
 values.set(i - 1, i * i);
}
- e) ArrayList<Integer> values;
for (int i = 1; i <= 10; i = i + 1) {
 values.add(i * i);
}

Uppgift 26.

Skriv en statisk metod

```
public static int nrOfTargets(ArrayList<Integer> list, Integer target)
```

som tar en lista **list** samt ett värde **target** och returnerar hur många gånger värdet **target** förekommer i listan **list**.

Uppgift 27.

Skriv en statisk metod

```
public static ArrayList<Integer> reverse(ArrayList<Integer> original)
```

som tar en lista **original** och returnerar en ny lista som innehåller samma element som i lista **original** men där elementen har omvänd ordning, dvs. det första elementet i listan **original** kommer sist list i den nya listan, det andra elementet i listan **original** kommer näst sist list i den nya listan, osv.

Uppgift 28.

Vad blir utskriften av nedanstående kodavsnitt?

```
int[] arr = {4, 2, 9, 7, 3};  
for (int k : arr) {  
    System.out.print(k + " ");  
}  
for (int k : arr) {  
    k = k + 10;  
    System.out.print(k + " ");  
}
```