Lösningsförslag: Instuderingsfrågor, del E

Uppgift 1.

```
int[][] b=  {{1,3,6,10},
             {2,5,9,13},
             {4,8,12,15},
             {7,1,14,16 }};
```

Uppgift 2.

d) `int[][] distance = new int[10][10];`

Uppgift 3.

a) `double[][] matris = new double(5)(4);` (och) istället för [och] i vänsterledet
b) `int[][] temp= new double[3][9];` olika typ i vänster- och högerled
c) `int[][] speed = {{1.0, 3.0, 5.0 }, {7.0 , 9.0}};` olika typ i vänster- och högerled

Uppgift 4.

a) 'b' b) 'z' c) {'a', 'b', 'c', 'd'}

Uppgift 5.

Utskriften blir:

```
4
3
5
2
```

Uppgift 6.

Utskriften blir:

```
2 1 1 1
3 2 1 1
3 3 2 1
```

Uppgift 7.

```
[1, 4, 7]
[2, 5, 8]
[3, 6, 9]
```

Uppgift 8.

Utskriften blir:

```
1
90
1
80
1
2
90
1
2
70
50
```

Uppgift 9.

```
public static boolean exist(int[][] m) {
    if (m == null || m.length == 0)
        return false;
    else
        return true;
} //exist
```

Uppgift 10.

```
public static int longestRow(int[][] m) {
    int longest = 0;
    for (int row = 0; row < m.length; row = row + 1) {
        if (m[row].length > longest)
            longest = m[row].length;
    }
    return longest;
} //longestRow
```

Uppgift 11.

```
public static int max(int[][] m) {
    int biggestSoFar = m[0][0];
    for (int row = 0; row < m.length; row = row + 1) {
        for (int col = 0; col < m[row].length; col = col + 1) {
            if (m[row][col] > biggestSoFar) {
                biggestSoFar = m[row][col];
            }
        }
    }
    return biggestSoFar;
}//max
```

Uppgift 12.

```
public double[][] identityMatrix(int size) {
    double[][] dArray = new double[size][size]; // allocate the two-dimensional array at once
    for (int row = 0; row < size; row = row + 1) {
        for (int column = 0; column < size; column = column + 1) {
            dArray[row][column] = 0;
        }
        dArray[row][row] = 1.0;
    }
    return dArray;
}// identityMatrix
```

Uppgift 13.

```
public static int largestSumOfRow(int[][] m) {
    int largestSoFar = Integer.MIN_VALUE;
    for (int row = 0; row < m.length; row = row + 1) {
        int sum = 0;
        for (int col = 0; col < m[row].length; col = col + 1) {
            sum = sum + m[row][col];
        }
        if (sum > largestSoFar) {
            largestSoFar = sum;
        }
    }
    return largestSoFar;
}//  largestSumOfRow
```

Uppgift 14.

```
public static int indexOfLargestRow(int[][] m) {
    int largestSoFar = 0;
    int indexOfLargest = 0;
    for (int row = 0; row < m.length; row = row + 1) {
        int sum = 0;
        for (int col = 0; col < m[row].length; col = col + 1) {
            sum = sum + m[row][col];
        }
        if (sum > largestSoFar) {
            largestSoFar = sum;
            indexOfLargest = row;
        }
    }
    return indexOfLargest;
}// indexOfLargestRow
```

Uppgift 15.

```
public static void largestRowFirst(int[][] arr) {
    int largestRow = indexOfLargestRow(arr);
    int[] temp = arr[0];
    arr[0] = arr[largestRow];
    arr[largestRow] = temp;
}//largestRowFirst
```

Uppgift 16.

```
public static int largestSumOfColumn(int[][] m) {
    int largestSoFar =  0;
    for (int row = 0; row < m.length; row = row + 1) {
        largestSoFar = largestSoFar + m[row][0];
    }
    for (int col = 1; col < m[0].length; col = col + 1) {
        int sum = 0;
        for (int row = 0; row < m.length; row = row + 1) {
            sum = sum + m[row][col];
        }
        if (sum > largestSoFar) {
            largestSoFar = sum;
        }
    }
    return largestSoFar;
}//largestSumOfColumn
```

Uppgift 17.

```
public static int min(int[][] m) {
    int smallestSoFar  = Integer.MAX_VALUE;
    for (int row = 0; row < m.length; row = row + 1) {
        for (int col = 0; col < m[row].length; col = col + 1) {
            if (m[row][col] < smallestSoFar) {
                smallestSoFar = m[row][col];
            }
        }
    }
    return smallestSoFar;
}//min
```

Uppgift 18.

```
public static int indexOfLargestColumn(int[][] m) {
    int largestSoFar = 0;
    for (int row = 0; row < m.length; row = row + 1) {
        largestSoFar = largestSoFar + m[row][0];
    }
    int indexOfLargest = 0;
    for (int col = 1; col < m[0].length; col = col + 1) {
        int sum = 0;
        for (int row = 0; row < m.length; row = row + 1) {
            sum = sum + m[row][col];
        }
        if (sum > largestSoFar) {
            largestSoFar = sum;
            indexOfLargest = col;
        }
    }
    return indexOfLargest;
}//  indexOfLargestColumn
```

Uppgift 19.

```
public static void largestColumnFirst(int[][] arr) {
    int largestColumn = indexOfLargestColumn(arr);
    for (int row = 0; row < arr.length; row = row + 1) {
        int temp = arr[row][0];
        arr[row][0] = arr[row][largestColumn];
        arr[row][largestColumn] = temp;
    }
}// largestColumnFirst
```

Uppgift 20.

Utskriften blir:

A
B
C
D

Uppgift 21.

Utskriften blir:

[1, 2, 5, 1, 2, 5]

Uppgift 22.

Utskriften blir:

[green, black]

Uppgift 23.

Utskriften blir:

[yellow, green, yellow, black]

Uppgift 24.

Utskriften blir:

[4, 7, 12, 19, 28, 39]

Uppgift 25.

a) Det går inte att lagra primitiva datatyper i en ArrayList. Objekt av tillhörande omslagsklass måste lagras. Deklarationen skall alltså vara:

```
ArrayList<Integer> values = new ArrayList<Integer>();
```

b) Vilken typ av värden som skall lagras i listan *måste* anges vid anropet av konstruktorn. Deklarationen skall alltså vara:

```
ArrayList<Double> values = new ArrayList<Double>();
```

c) Parameterlistan *måste* anges vid anrop till konstruktorn. Deklarationen skall alltså vara:

```
ArrayList<String> values = new ArrayList<String>();
```

d) Ingen insatns av ArrayList har skapats, varför referensvariabeln values har värdet **null**. Innan forsatsen måste listan skapas med satsen

```
values = new ArrayList<Integer>();
```

e) values refererar till en ArrayList som innehåller 0 element varför det inte går att ändra värdet på de 10 första elementen i listan, eftersom dessa inte finns. Troligen skall elementen inte ändras utan läggas in i listan, vilket innebär att det är metoden add som skall anropas och inte metoden set. Detta går bra eftersom elementen position 0, 1, 2, .., 9 läggs in i tur och ordning.

Uppgift 26.

```
public static int nrOfTargets(ArrayList<Integer> list, Integer target) {
    int nrOf = 0;
    for (int i = 0 ; i < list.size(); i = i + 1) {
        if (target.equals(list.get(i)))
            nrOf = nrOf + 1;
    }
    return nrOf;
}//nrOfTargets
```

Uppgift 27.

```
public static ArrayList<Integer> reverse(ArrayList<Integer> original) {
    ArrayList<Integer> reversed = new ArrayList<Integer>();
    for (int i = original.size() - 1 ; i >= 0; i = i - 1) {
        reversed.add(original.get(i));
    }
     return reversed;
}// reverse
```

Uppgift 28.

4 2 9 7 3 14 12 19 17 13