Problem 1

Write a static function which takes an array of **int**s and returns the difference between the index of the last even element and the index of the first even element (or -1 if there are no at least two even elements). Create a program which prints the array (in one line) and tests your function.

Problem 2

Write a static function which takes an array of **int**s and swaps (exchanges the values) of its greatest and smallest elements. Create a program which prints the array (in one line) before and after this operation.

Problem 3

Define the following static functions operating on arrays of **int**s:

- **printArr**: takes an array and prints, in square brackets, its elements in one line, separated with spaces;
- **numOdd**: takes an array and returns the number of its odd elements;
- **numEven**: takes an array and returns the number of its even elements;
- **getOddEven**: takes an array and creates two (possibly empty) arrays — one containing only the odd elements of the input array and one containing only its even elements — and returns a two-element array containing the references to these two arrays;
- **getMinMaxInd**: takes an array and returns a two-element array containing the indices of the minimum and maximum elements.

For example, the following **main** function

download *FuncArrs.java*

```
public static void main (String[] args) {
    int[] a = {-1, 9, 3, 0, -3, 2, 4};
    System.out.print("array: ");
    printArr(a);
    System.out.println(" odd: " + numOdd(a));
    System.out.println("even: " + numEven(a));

    int[] indMinMax = getMinMaxInd(a);
    System.out.println("Index of min: " + indMinMax[0]);
    System.out.println("Index of max: " + indMinMax[1]);

    int[][] res = getOddEven(a);
    System.out.print(" odd elements: ");
    printArr(res[0]);
    System.out.print("even elements: ");
```

```
        printArr(res[1]);
    }
```

should print something like

```
    array: [ -1 9 3 0 -3 2 4 ]
     odd: 4
    even: 3
    Index of min: 4
    Index of max: 1
     odd elements: [ -1 9 3 -3 ]
    even elements: [ 0 2 4 ]
```

Problem 4

Define a two-dimensional array of **String**s with names of countries and their capitals, for example as below

download *Arr2DCapitals.java*

```
        String[][] arr =
              { {"Kenya",  "Nairobi"}, {"Rwanda", "Kigali"},
                {"Gambia", "Banjul"},  {"Ghana",  "Accra"},
                {"Niger",  "Niamey"},  {"Zambia", "Lusaka"} };
```

and then write a program which reads the name of a country and prints the name of its capital (don't forget that **String**s should be compared using **equals** or **equalsIgnoreCase**, *not* == operator!). The program should display an appropriate message if the country name entered cannot be found.

Problem 5

Define a three-dimensional array of **int**s representing deposits and withdrawals of customers of a bank in and out of their accounts (each customer may have several accounts). For example

download *Arr3DBank.java*

```
        int[][][] opers = {
            { {100, -50, 25}, {150,-300}, {300,-90,100} },
            { {90, -60, 250}, {300,20,-100} },
            { {20, 50}, {300}, {20,-20,40}, {100,-200} }
        };
```

where

- the first index indicates a customer;
- the second index indicates, for a given customer, his/her account;
- the third index indicates, for a given customer and his/her account, subsequent deposits (positive values) and withdrawals (negative values).

The program should create an array of **int**s of dimension equal to the number of customers, the elements of which are sums of all deposits and withdrawals for subsequent customers, in and out of all his/her accounts (for the data as in the example above, these should be the numbers 235, 500 and 310).

After modifications of the sizes and/or values of elements of the array oper, the program should work correctly without any other changes.

Problem 6

Write a static function **inner** which takes a two-dimensional *rectangular* (by assumption) array of **int**s (we assume that both number of rows and number of columns are not smaller than three). The function creates and returns a new two-dimensional array which contains the "inner part" of the original array, i.e., without the first and the last row and without the first and the last column. It doesn't print anything!

For example, the following program

download *Arr2DInn.java*

```
import java.util.Arrays;
public class Arr2DInn {
    // ...
    public static void main (String[] args) {
        int[][] a = { {1,2,3,4,5,6},
                      {2,3,4,5,6,7},
                      {3,4,5,6,7,8},
                      {4,5,6,7,8,9} };
        for (int[] r : a)
            System.out.println(Arrays.toString(r));
        System.out.println();
        for (int[] r : inner(a))
            System.out.println(Arrays.toString(r));
    }
}
```

should print

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

[3, 4, 5, 6]
[4, 5, 6, 7]
```