



**Small changes made**

Christoffer Holm Rosenkilde authored 2 months ago



Forked from an inaccessible project.

**readme.md** 2.21 KiB

# Tasks

## Recursion

Package `FilesInDirs` contains `FindFilesRecursive.java`. The purpose of the class is to recursively iterate through a given directory, and its subdirectories.

- Implement the method `findFiles(File file)` in a way, so that it increments the variable `numFiles`, for every "atomic" file (atomic meaning it has no sub-files), and simultaneously writes the full file path in the console, for each file incremented. If the method encounters a directory, it should instead increment the variable `numDirs` and make a recursive call (meaning `FilesInDirs` should be called in the method).

### Manipulation of arrays

Within the package `ArrayManipulation` contains the class `ArrayManipulation.java`. Within the `ArrayManipulation.java`-class you will find the methods `evenOdd()` and `sort()`. The method `evenOdd()` should take an array of random numbers (between 0 - 100) as input.

### Task A

The numbers must be arranged so that all odd numbers precede all even numbers. Write a method using the method signature `public int[] evenOdd(int [] array)`, which rearranges all the numbers in the array. For example:

Input: [71, 1, 2, 68, 36, 59, 70, 22, 81, 89]

Output: [71, 1, 59, 81, 89, 22, 70, 36, 68, 2]

Hint: You might want somewhere to temporarily store your numbers, while you are sorting them

### Task B

For this task you are going to be implementing the method signature `private void sort(int [] array, int splitIndex)`. The point of the method is to sort the input array, in a manner so that it arranges the numbers of the within the array in ascending order.

The `int splitIndex` is supposed to denote where in the array, the sorting must start over from lowest to highest, in case you might have more than one type of numbers to be sorted. We now want to call this method from within the `evenOdd()`-method, as we want the two parts of the array to be sorted like they are underneath:

Input: [71, 1, 2, 68, 36, 59, 70, 22, 81, 89]

EvenOdd: [71, 1, 59, 81, 89, 22, 70, 36, 68, 2]

Output: [1, 59, 71, 81, 89, 70, 68, 36, 22, 2]

In the example above the `int splitIndex` is set to 5, in the context of a 5 number array.

Hint: You might want to check out the `Arrays.sort()` function for this



```

1 package FilesInDirs;
2
3 import java.io.File;
4 import java.util.Scanner;
5 public class FindFilesRecursive {
6     private int numDirs;
7     private int numFiles;
8     public static void main(String[] args) {
9         // Prompt the user to enter a directory or a
10        file
11        System.out.print("Enter a directory or a file
12        : ");
13        Scanner input = new Scanner(System.in);
14        String directory = input.nextLine();
15        File startDir = new File(directory);
16
17        FindFilesRecursive ffr = new
18        FindFilesRecursive();
19        ffr.findFiles(startDir);
20        System.out.println("\n*****\n" + ffr
21        );
22    }
23
24    // Exercise: If a file is a directory: Call all
25    files recursively,
26    // else print full path to the file. Count both
27    dirs and atomic files.
28    private void findFiles(File file) {
29        if(file.isDirectory()){
30            File[] files = file.listFiles();
31            numDirs++;
32        }else{
33            System.out.println(file.getPath());
34            numDirs++;
35        }
36    }
37
38    @Override
39    public String toString() {
40        return "FindFilesRecursive{" + "noDirs=" +
41        numDirs + ", noFiles=" + numFiles + '}';

```

```
35     }  
36 }  
37
```

```
1 package ArrayManipulation;
2
3 import java.sql.Array;
4 import java.util.Arrays;
5 import java.util.Random;
6 public class ArrayManipulation {
7     public static void main(String[] arg) {
8         Random generator = new Random(222);
9         int[] array = new int[10];
10        for (int i = 0; i < array.length; i++) {
11            array[i] = generator.nextInt(100);
12        }
13        System.out.println("Input: " + Arrays.
toString(array));
14
15        ArrayManipulation arrMani = new
ArrayManipulation();
16        int[] result = arrMani.evenOdd(array);
17        System.out.println("Output: " + Arrays.
toString(result));
18    }
19
20    public int[] evenOdd(int[] array) {
21        //Her har jeg lavet en Array som et objekt og
defineret dets længde.
22        int[] order = new int[array.length];
23        //Her har jeg lavet en variabel som holder
styr på ulige tal.
24        //Jeg har lavet en variabel også, som holder
styr på lige tal.
25        int odd = 0;
26        int even = order.length-1;
27        //For hvert tal har jeg forsøgt i array at
finde en circular array index.
28        for(int number : array){
29            if(number % 2 == 0){
30                //Her har vi sagt, at hvis vi i array
har et lige tal i index, så er den tal lige.
31                order[even] = number;
32                even--;
33            }else{
```

```
34          //Her har vi sagt, at hvis vi i array
           har et ulige tal i index, så er den ulige.
35          order[odd] = number;
36          odd++;
37      }
38      //Her udprinter vi resultatet og derved
           sortere tallene udefra deres forekommelse i listen.
39      System.out.println("evenOdd: " +Arrays.
           toString(order));
40      sort(order, odd);
41      return order;
42  }
43  }
44
45  private void sort(int[] array, int splitIndex) {
46      Arrays.sort(array,0,splitIndex);
47      //Her splitter vi indekset, så vi har lige og
           ulige tals sorteringer.
48      for(int i = splitIndex; i<array.length; i++){
49          array[i] = -array[i];
50      }
51      Array.sort(array,splitIndex,array.length-1);
52
53      for(int i = splitIndex; i< array.length;i++){
54          array[i] = +array[i];
55      }
56  }
57 }
58
59
60
```