CS 162FZ: Introduction to Computer Science II

Lecture 10

Tutorial on Lab4

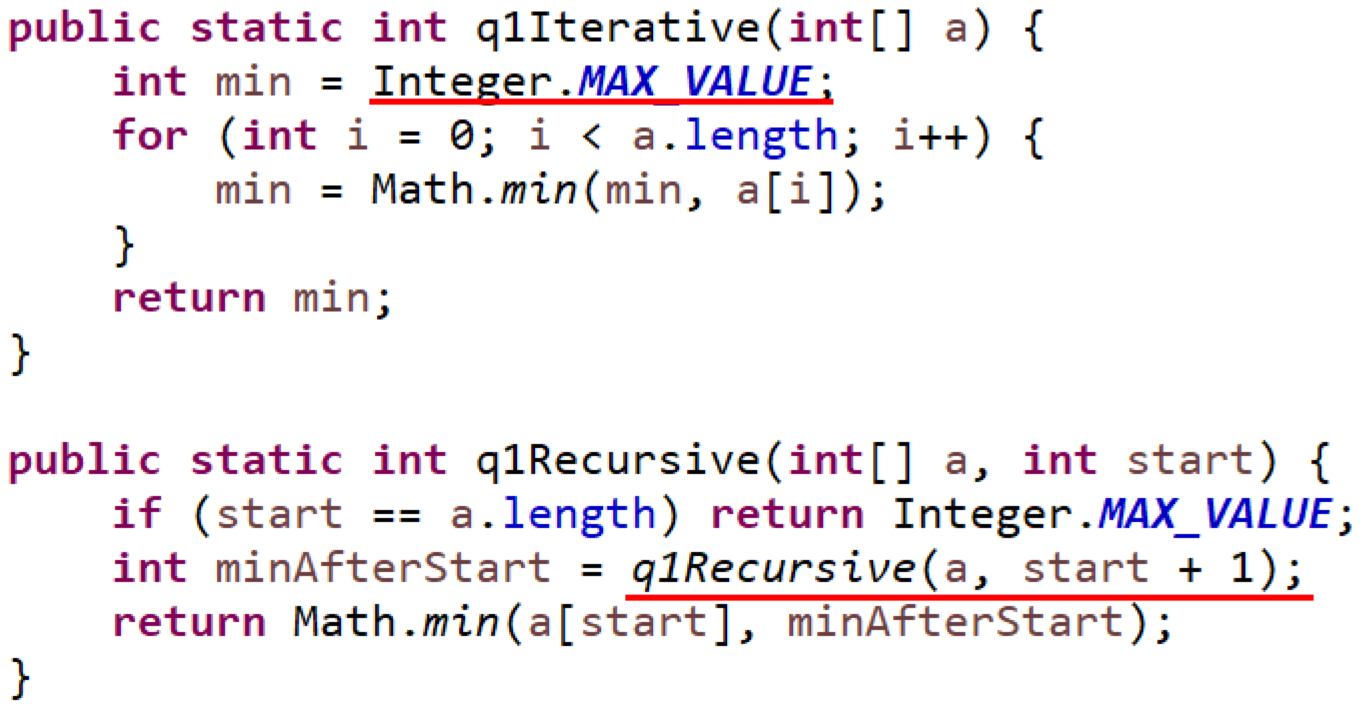
Dr. Chun-Yang Zhang

Lab1, Task1: Minimum Value

Write two Java methods which return the minimum value in an integer array. You should write:

- An iterative method which returns the minimum value in an integer array. In this method you must use either a for or a while loop. You should call this method q1Iterative and it should have a definition as follows: public static int q1Iterative(int[] a) where a is the input array.
- A recursive method which returns the minimum value in an integer array. You should call this method q1Recursive. The definition of the method should be declared as follows: public static int q1Recursive(int[] a, int start) where a is the input array.

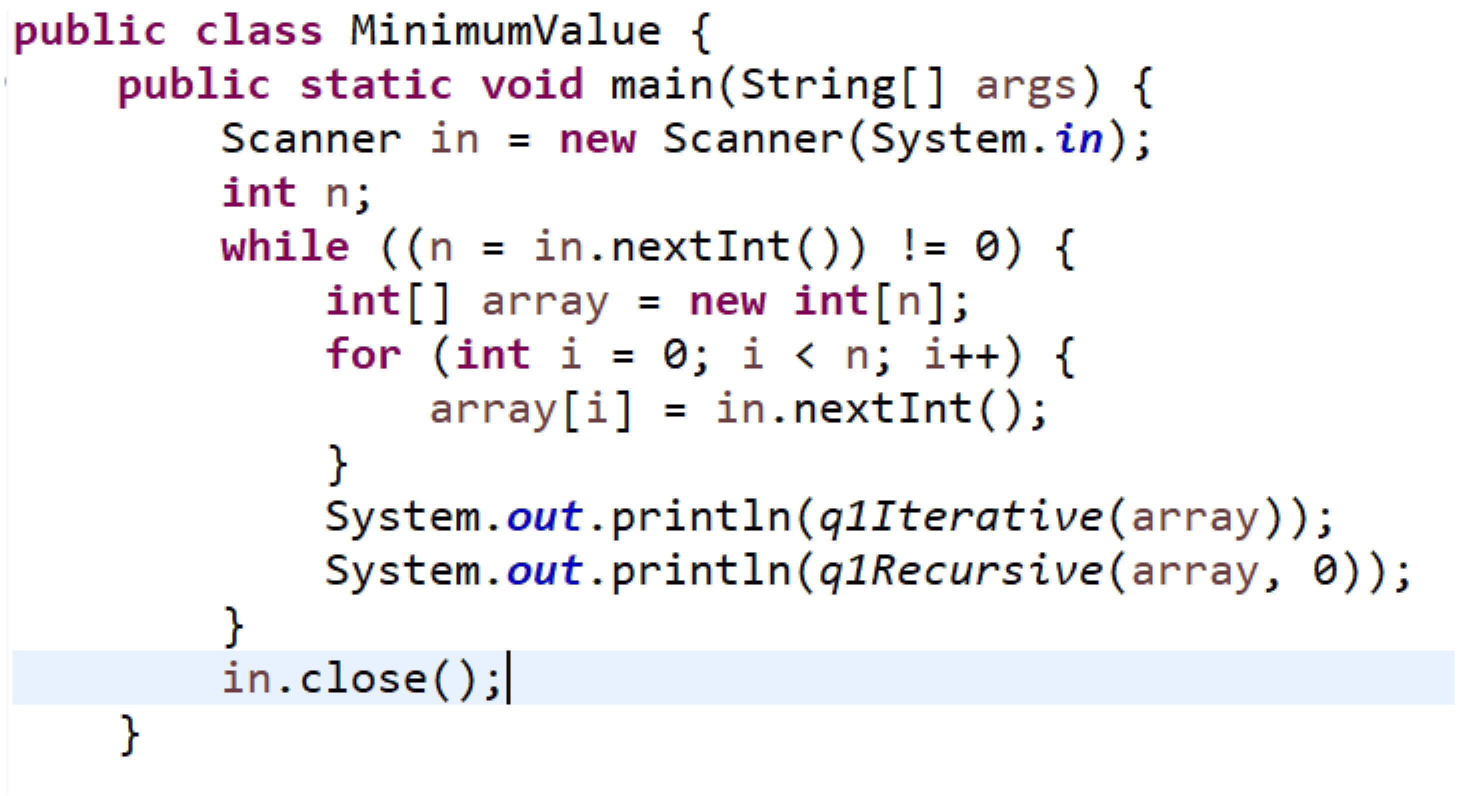
Solution



```java
public static int q1Iterative(int[] a) {
    int min = Integer.MAX_VALUE;
    for (int i = 0; i < a.length; i++) {
        min = Math.min(min, a[i]);
    }
    return min;
}

public static int q1Recursive(int[] a, int start) {
    if (start == a.length) return Integer.MAX_VALUE;
    int minAfterStart = q1Recursive(a, start + 1);
    return Math.min(a[start], minAfterStart);
}
```

Solution



```
public class MinimumValue {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n;
        while ((n = in.nextInt()) != 0) {
            int[] array = new int[n];
            for (int i = 0; i < n; i++) {
                array[i] = in.nextInt();
            }
            System.out.println(q1Iterative(array));
            System.out.println(q1Recursive(array, 0));
        }
        in.close();
    }
```

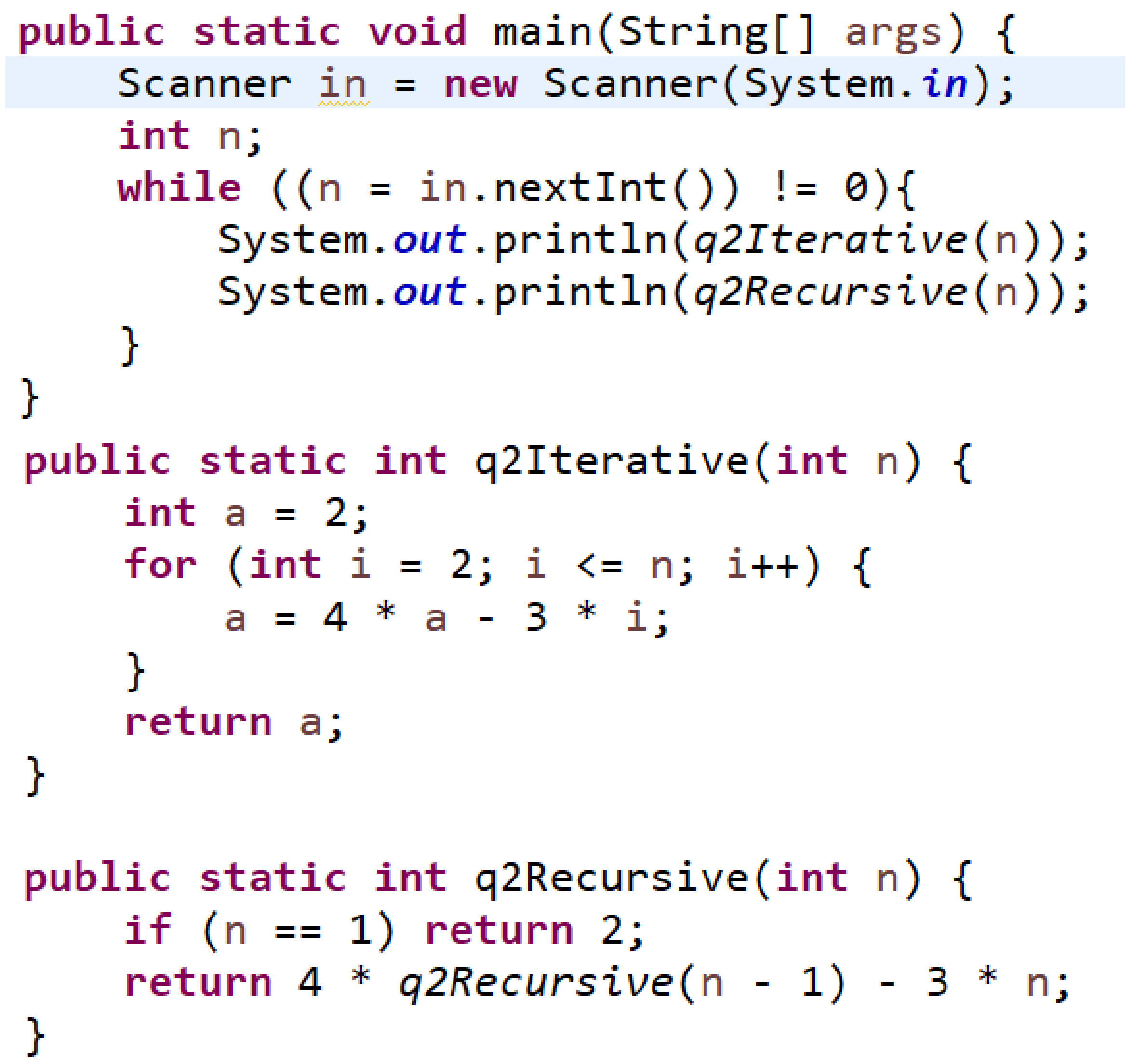
Lab1, Task2: Maths Function

$$a(n) = \begin{cases} a_1 = 2, & n = 1 \\ a_n = 4a_{n-1} - 3n, & n \geq 2 \end{cases}$$

Given the functional definition below for the mathematical function *a(n)*, shown above, you are required to write two Java methods – both methods will be in the same class. You will have seen an example like this in your lecture notes from this week's lectures:

- An iterative method which calculates *a(n)* for any positive value of *n*. In this method you **must** use either a for or a while loop. You should call this method `q2Iterative` and it should have a definition as follows: `public static int q2Iterative(int n)`
- A recursive method which calculates *a(n)* for any positive value of *n*. You should call this method `q2Recursive`. The definition of the method should be declared as follows: `public static int q2Recursive(int n)`

Solution



```java
public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    int n;
    while ((n = in.nextInt()) != 0){
        System.out.println(q2Iterative(n));
        System.out.println(q2Recursive(n));
    }
}
public static int q2Iterative(int n) {
    int a = 2;
    for (int i = 2; i <= n; i++) {
        a = 4 * a - 3 * i;
    }
    return a;
}

public static int q2Recursive(int n) {
    if (n == 1) return 2;
    return 4 * q2Recursive(n - 1) - 3 * n;
}
```

Lab2, Task3: String Reverse

Write a recursive method (no loops) which takes a string and reverses it. You should call this method `reverseRecursive`. The method should take a string and return the string reversed.

Test cases

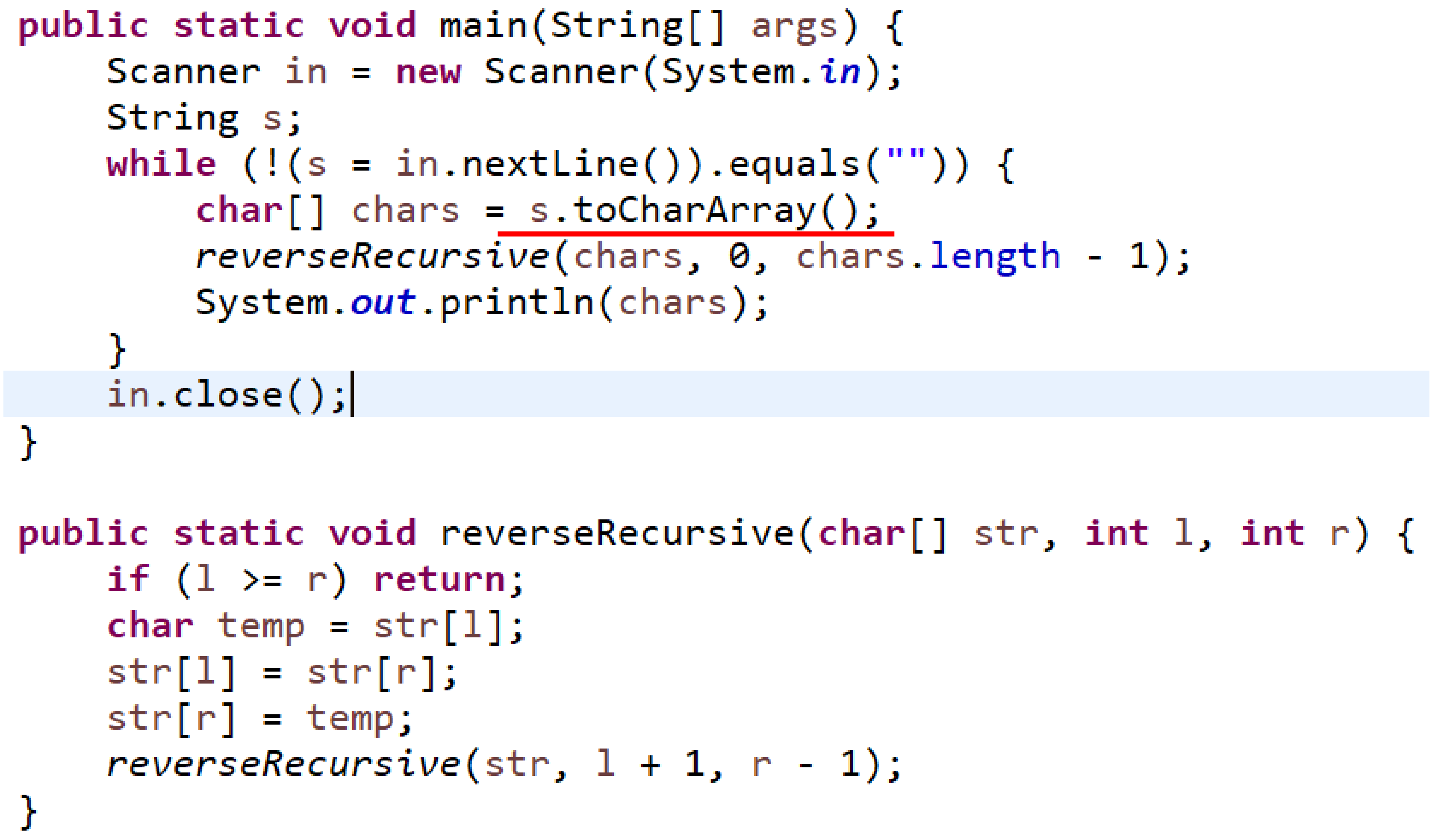
Input: n, where n is a String

Sample Input: Hello

Output: The String reversed

Sample Output: olleH

Solution



```java
public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    String s;
    while (!(s = in.nextLine()).equals("")) {
        char[] chars = s.toCharArray();
        reverseRecursive(chars, 0, chars.length - 1);
        System.out.println(chars);
    }
    in.close();
}

public static void reverseRecursive(char[] str, int l, int r) {
    if (l >= r) return;
    char temp = str[l];
    str[l] = str[r];
    str[r] = temp;
    reverseRecursive(str, l + 1, r - 1);
}
```

Lab1, Task 4: Sum Digits

Write a recursive method (no loops) which takes in a non-negative integer and returns the sum of all the digits in a number. You should call this method `recursiveSum`.

For Example

recursiveSum(126) would return 9 as 1 + 2 + 6 = 9

Test cases

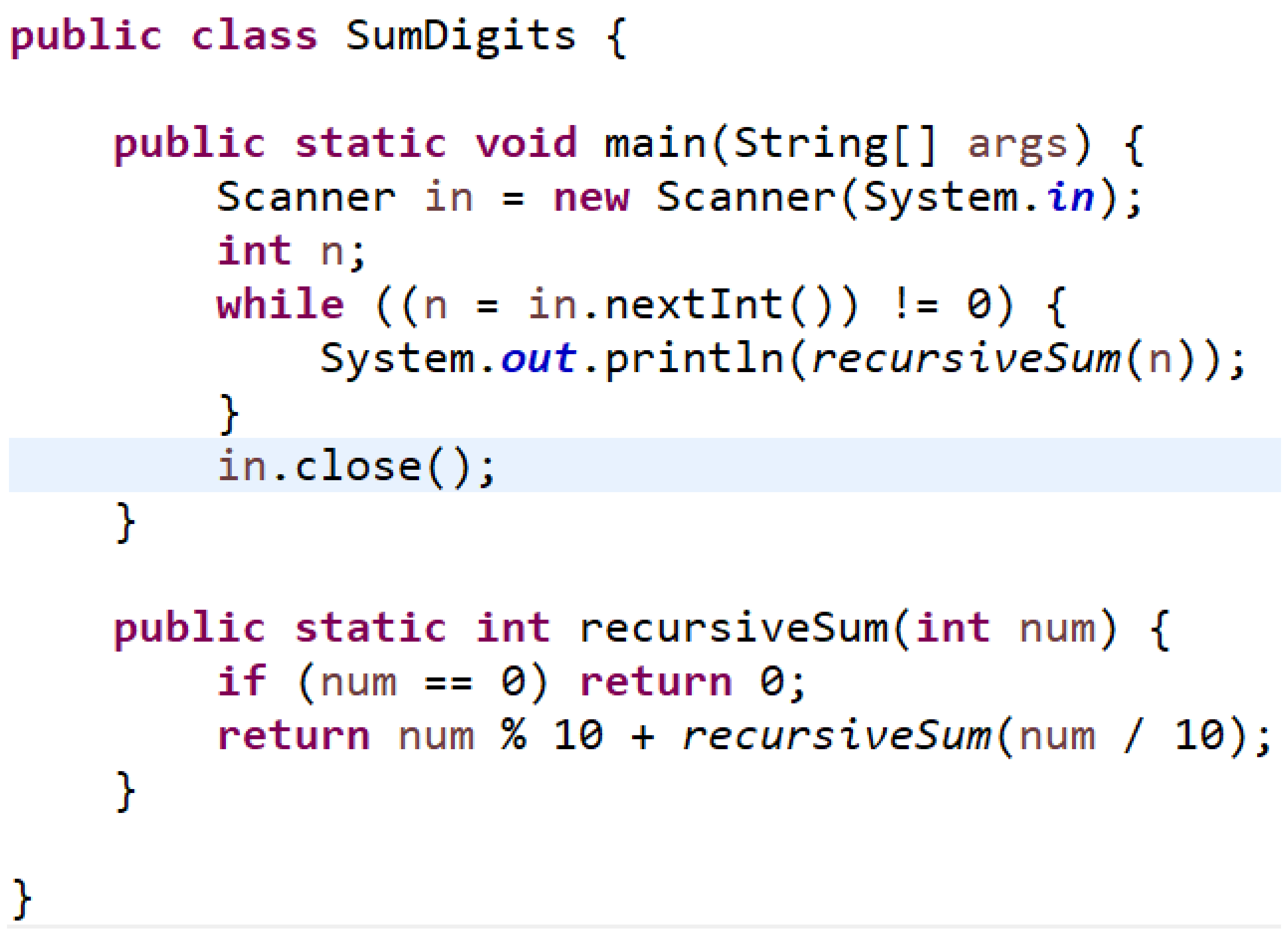
Input: n, where n is a non-negative integer

Sample Input: 126

Output: The sum of all single digits in a number

Sample Output: Res: 9

Solution



```
public class SumDigits {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n;
        while ((n = in.nextInt()) != 0) {
            System.out.println(recursiveSum(n));
        }
        in.close();
    }

    public static int recursiveSum(int num) {
        if (num == 0) return 0;
        return num % 10 + recursiveSum(num / 10);
    }

}
```

Lab1, Task 5: Array value times 10

Given an integer array of size n write a recursive method (no loops) that checks if the array contains somewhere a value followed in the array by that value times 10. You should call this method `checkArray`. The method should take an array and a starting index and should return a Boolean value. The array should be created using the fillArray() method provided.

For Example

checkArray([1, 2, 20], 0) → true
checkArray([3, 30], 0) → true
checkArray([3], 0) → false
checkArray([1, 10, 40, 6, 60, 60, 54] , 4) → false

Test cases

Input:

n m, where n is the length of the array, followed by m numbers

Sample Input:

3 1 2 20

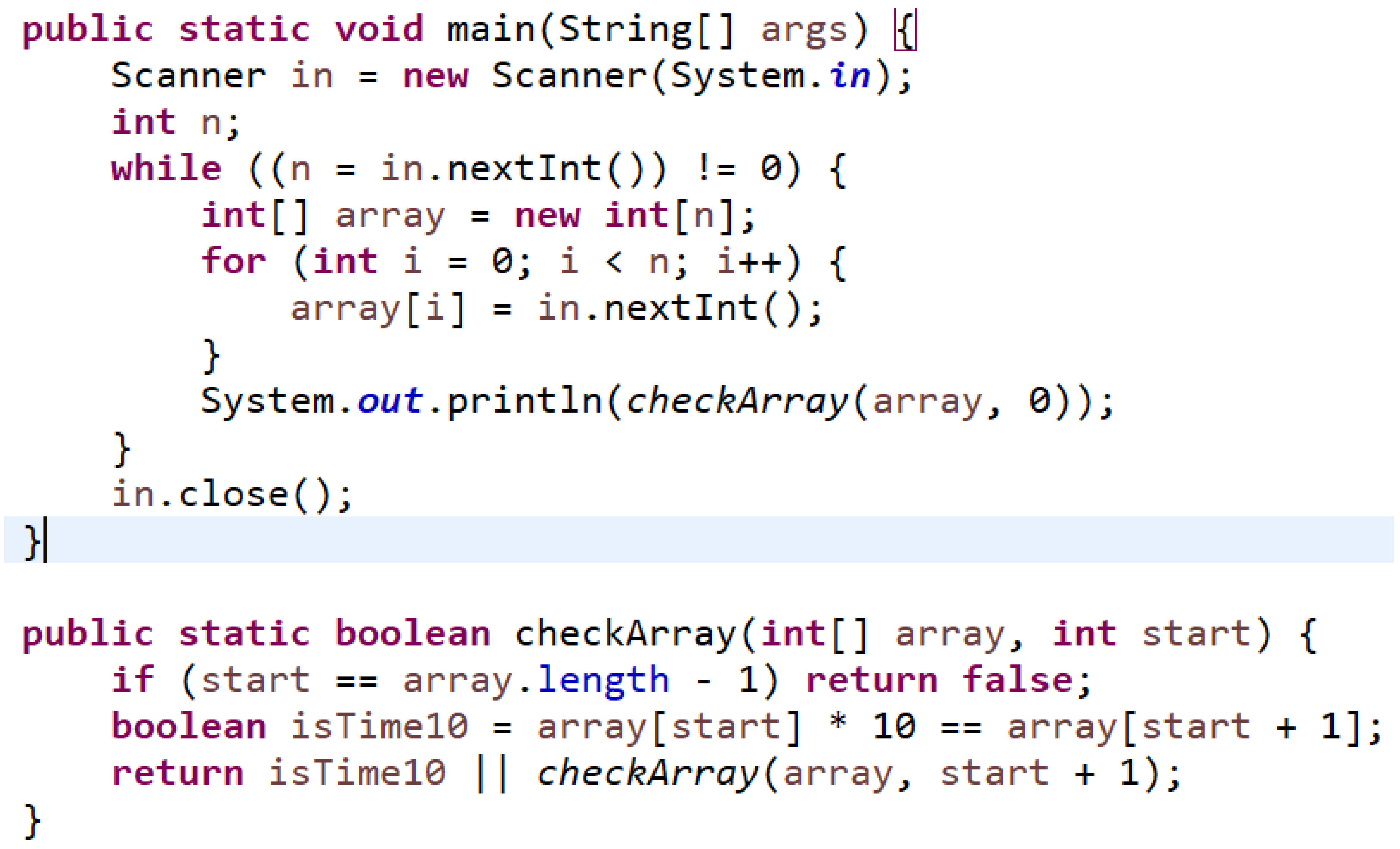
Output:

A boolean value indicating if the value is followed by the value time 10

Sample Output:

true

Solution



```java
public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    int n;
    while ((n = in.nextInt()) != 0) {
        int[] array = new int[n];
        for (int i = 0; i < n; i++) {
            array[i] = in.nextInt();
        }
        System.out.println(checkArray(array, 0));
    }
    in.close();
}

public static boolean checkArray(int[] array, int start) {
    if (start == array.length - 1) return false;
    boolean isTime10 = array[start] * 10 == array[start + 1];
    return isTime10 || checkArray(array, start + 1);
}
```

Lab2, Task6: Factorial

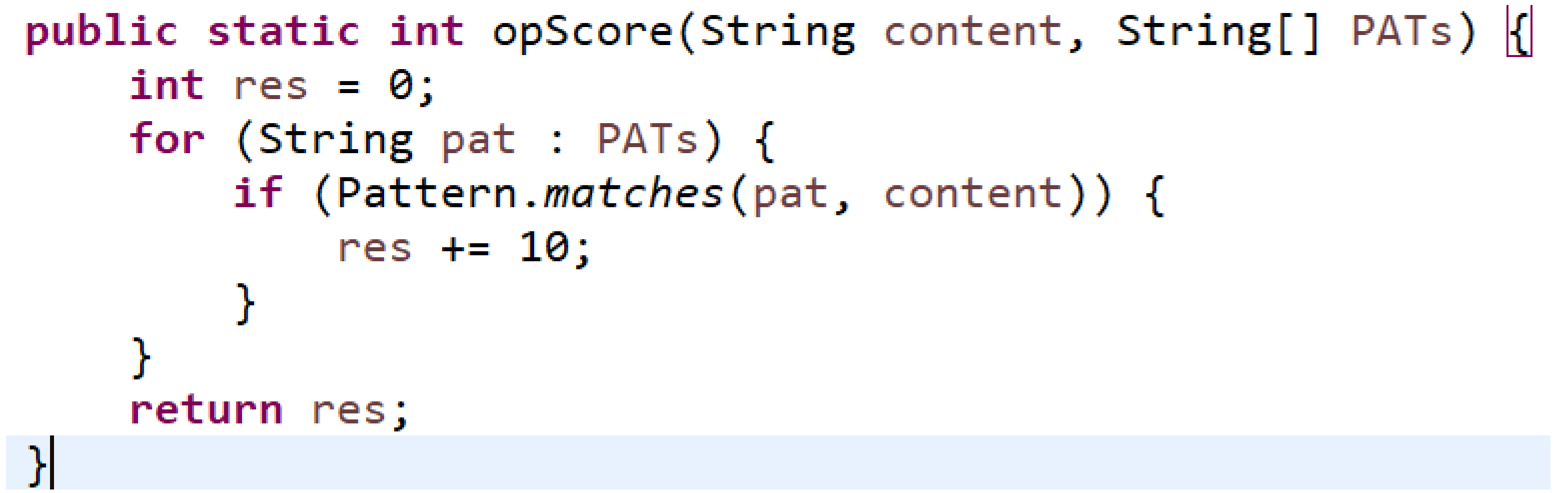
Using the code template given in Programming4.java and the 4 pieces of code in the folder “Student Code” your task is to use your knowledge of regular expressions to write a piece of code that will correct 4 pieces of Student code. The students were asked to write java code to print all the even numbers between 0 and 50 (inclusive). 4 Students have submitted code and the results are stored in the folder “Student code”. You can award the following marks for the code having each of the following: (possible answers given, they can have other variations)

- Class declaration (10 Marks)
  - ➢ .*public class.*
- A main method (10 Marks)
  - ➢ .*void main\(String \[\] args\).*
- A syntactically correct for loop (10 Marks)
  - ➢ .*for\(.*;.*;.*\).*
- The modulo operator (10 Marks)
  - ➢ .*%.*
- A valid print statement (10 Marks)
  - ➢ .*System\.out\.print.*

Solution

```java
public static void main(String[] args) {
    String[] PATs = {
            ".*public class.*",
            ".*void main\\(String \\[\\] args\\).*",
            ".*for\\(.*;.*;.*\\).*",
            ".*%.*",
            ".*System\\.out\\.print.*"};
    String prefix = "C:\\Users\\zhang\\eclipse-workspace"
            + "\\Lab4\\src\\lab4\\Student Code\\Student";
    String suffix = ".java";
    for (int i = 1; i <= 4; i++) {
        String filePath = prefix + i + suffix;
        String[] contents = codeReader(filePath);
        StringBuilder sb = new StringBuilder();
        for (String content : contents) {
            sb.append(content);
        }
        System.out.printf("Student %d: %d Marks\n", i, opScore(sb.toString(), PATs));
    }
}
```

Solution



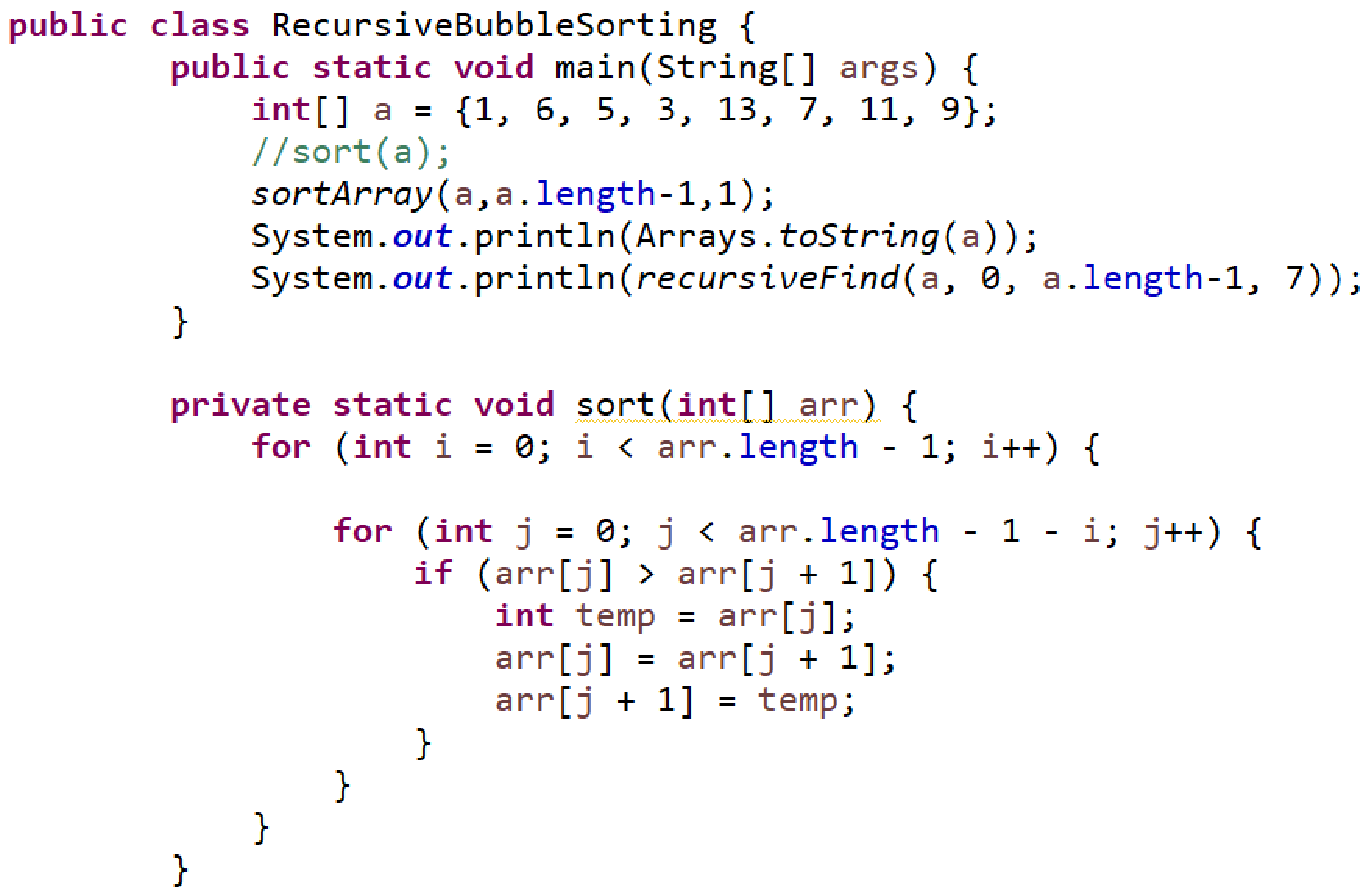
```java
public static int opScore(String content, String[] PATs) {
    int res = 0;
    for (String pat : PATs) {
        if (Pattern.matches(pat, content)) {
            res += 10;
        }
    }
    return res;
}
```

Solution cont'd

```java
public static String[] codeReader(String fileName) {
    //Path to file to read in
    String myfile = fileName;
    //Create an ArrayList (a dynamic type of Array)
    ArrayList<String> lines = new ArrayList<String>();
    try {
        // try to read in the File
        BufferedReader abc = new BufferedReader(new FileReader(myfile));

        String line;
        while ((line = abc.readLine()) != null) {
            lines.add(line);
        }
        abc.close();
    } catch (FileNotFoundException ex) {
        //If the File is not found print to screen
        System.out.println(
                "Unable to open file '" +
                        myfile + "'");
    } catch (IOException ex) {
        //If file is corrupted print to screen
        System.out.println(
                "Error reading file '"
                        + myfile + "'");
    }
    //Convert from ArrayList to Array
    return lines.toArray(new String[]{});
}
```

Recursive Bubble Sorting and Binary Searching



```java
public class RecursiveBubbleSorting {
        public static void main(String[] args) {
            int[] a = {1, 6, 5, 3, 13, 7, 11, 9};
            //sort(a);
            sortArray(a,a.length-1,1);
            System.out.println(Arrays.toString(a));
            System.out.println(recursiveFind(a, 0, a.length-1, 7));
        }

        private static void sort(int[] arr) {
            for (int i = 0; i < arr.length - 1; i++) {

                for (int j = 0; j < arr.length - 1 - i; j++) {
                    if (arr[j] > arr[j + 1]) {
                        int temp = arr[j];
                        arr[j] = arr[j + 1];
                        arr[j + 1] = temp;
                    }
                }
            }
        }
```

Solution cont'd

```java
public static void sortArray(int[] array, int m, int n) {
    if (m > 0) {
        if (array[n] < array[n - 1]) {
            int temp = array[n];
            array[n] = array[n - 1];
            array[n - 1] = temp;
        }
        if (n >= m) {
            sortArray(array, m - 1, 1);
        } else {
            sortArray(array, m, n + 1);
        }
    }
}
```

Solution cont’d

```
private static int recursiveFind(int[] arr,int start,int end,int searchKey){
    if (start <= end) {
        int middle =  (start+end)/2;
        if (searchKey == arr[middle]) {
            return middle;
        } else if (searchKey < arr[middle]) {
            return recursiveFind(arr, start, middle - 1, searchKey);
        } else {
            return recursiveFind(arr, middle + 1, end, searchKey);
        }
    } else {
        return -1;
    }
}
```