Technical Test

Java

June 2022

Presented by
Tomás Rodeghiero

Río Cuarto, Córdoba

JAVA EXERCISES

1. Describe what Hibernate is and what it is used for.

ANWER

In Java, **Hibernate** is an object-relational mapping tool that facilitates the mapping of attributes between a traditional relational database and the object model of an application, which is used to generate SQL statements, allowing the developer to manually manage the data resulting from the execution of these statements, maintaining portability between all databases with a slight increase in execution time.

2. Write a Java program to generate a magic square of order n (all row, column, and diagonal sums are equal).
From Wikipedia, In recreational mathematics and combinatorial design, a magic square is a n x n square grid (where n is the number of cells on each side) filled with distinct positive integers in the range 1, 2, ..., n2 such that each cell contains a different integer and the sum of the integers in each row, column and diagonal is equal. The sum is called the magic constant or magic sum of the magic square. A square grid with n cells on each side is said to have order n.

ANWER

```
public class MagicSquare {

    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        if (n % 2 == 0) throw new RuntimeException("n must be odd");

        int[][] magic = new int[n][n];

        int row = n-1;
        int col = n/2;
        magic[row][col] = 1;

        for (int i = 2; i <= n*n; i++) {
            if (magic[(row + 1) % n][(col + 1) % n] == 0) {
                row = (row + 1) % n;
                col = (col + 1) % n;
            }
            else {
                row = (row - 1 + n) % n;
                // don't change col
            }
            magic[row][col] = i;
        }
        // print results
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (magic[i][j] < 10)  System.out.print(" ");  // for alignment
                if (magic[i][j] < 100) System.out.print(" ");  // for alignment
                System.out.print(magic[i][j] + " ");
            }
            System.out.println();
        }

    }
}
```

3) Given an array of integers.
Write an algorithm that brings all nonzero elements to the left of the array,
and returns the number of nonzero elements.
The algorithm should operate in place, i.e. shouldn't create a new array.
The order of the nonzero elements does not matter.
The numbers that remain in the right portion of the array can be anything.

Example:
Given the array [1, 0, 2, 0, 0, 3, 4], A possible answer is [4, 1, 3, 2, ?, ?, ?],
4 non-zero elements, where "?" can be any number.

Code should have good complexity and minimize the number of writes to the array.

```
          [ 1, 0, -2, 0, 0, 3, 4, 0, 0]
          [ 4, 1, 3, -2, ?, ?, ? ] return 4
```

- What is the complexity of your algorithm?

ANWER

```
import java.io.*;

class PushZero
{
    // Function which pushes all zeros to end of an array.
    static void pushZerosToEnd(int arr[], int n)
    {
        int count = 0;  // Count of non-zero elements

        // Traverse the array. If element encountered is
        // non-zero, then replace the element at index 'count'
        // with this element
        for (int i = 0; i < n; i++)
            if (arr[i] != 0)
                arr[count++] = arr[i]; // here count is
                                       // incremented

        // Now all non-zero elements have been shifted to
        // front and 'count' is set as index of first 0.
        // Make all elements 0 from count to end.
        while (count < n)
            arr[count++] = 0;
    }

    /*Driver function to check for above functions*/
    public static void main (String[] args)
    {
        int arr[] = {1, 9, 8, 4, 0, 0, 2, 7, 0, 6, 0, 9};
        int n = arr.length;
        pushZerosToEnd(arr, n);
        System.out.println("Array after pushing zeros to the back: ");
        for (int i=0; i<n; i++)
            System.out.print(arr[i]+" ");
    }
}
```

Note: I admit that the codes in this document are not mine, since they have been researched on the internet, because I have never worked with the Java language, but I have been understanding the logic of its codes. However, I promise, in case you guys consider it, I will learn this language more thoroughly in two weeks by studying it hard.

4) Create a package called documents...

Note: Exercise not solved due to my lack of understanding in the Java language.

A. Include in it two classes, Invoice and Order, to represent invoices and orders, respectively.

B. Next, out of the package, create a small program that creates objects of both types and displays them on the screen.

C. Add a third document type, UrgentOrder, that inherits directly from Order. Checkthat the above program is still working properly if we replace an Order with an UrgentOrder.

D. Create a new document type, called Contract, and include it in the HumanResources documents subpackage. In this last package, include also a type of CV document to represent the curriculum vitae of a person.

E. If you haven't already done so, create a generic Document class from which inherit (directly or indirectly) all the other classes we have defined to represent different types of documents.

F. Implement a small program that creates a document of a type selected by the user. Display the document regardless of the specific type of document that has been created in the previous step