Universidad de las Fuerzas Armadas-ESPE, DCCO- SW - OOP 9652 - Final Exam

Puntos totales 33/51 ?

This is it! The semester is over
I hope you enjoyed the ride!
Good Luck, future Software Engineers!

Se ha registrado el correo del encuestado (**asmiranda@espe.edu.ec**) al enviar este formulario.

8 de 11 puntos

Full Name (LastName FirstName) for example. Lascano Jorge * 0/1

Miranda Alison

Select the right answer

1. Mongo DB is? * 1/1

● NO-SQL Database that uses JSON by default

○ SQL Database

○ Cloud Database

2. Modularity for OOP is expressed by? * 1/1

○ Classes and methods

○ Packages and methods

● Packages, classes and methods

3. The Essence of Encapsulation is? * 1/1

● Ensure that the private implementation details of a component are insulated so they cannot be accessed or modified by other components

○ Ensure that the public implementation details of a component are not insulated so they can be accessed or modified by other components

○ Ensure that the private implementation details of a component are public so they can be accessed or modified by other components

4. What is an object oriented programming Language only? 0/1

○ Smalltalk

○ Python

● Java

5. Diference between an abstract class and an interface? * 0/1

● An interface has method implemetations, an abstraction doesn't

○ An abstract class only has method protoypes

○ An abstract class has method implemetations, an interface doesn't

6. S in SOLID principles tells that? * 1/1

○ A class must not depend on other class

○ An application must be based in interfaces

● A class only does one thing related to its target

7. O in SOLID principles tells that? * 1/1

○ Software entities must be open for extension and open for modification

○ Software entities must be closed for extension but open for modification

● Software entities must be open for extension but closed for modification

8. L in SOLID principles tells that? * 1/1

○ We must program towards interfaces and layers

○ Concrete classes don’t depend on abstract classes

● Specialization Objects can replace its generalization objects

9. Localization of design decisions is related to? * 1/1

● modularization

○ polymorphism

○ inheritance

10. What Programming Paradigm did you learn the last semester? * 1/1

○ Object Oriented Programming Paradigm

○ Functional Programming

● Procedural Programming

Which of the following sentences are NOT true? (mark with an X) **6 de 10 puntos**

	NOT TRUE	Puntuación
We should not use humor when naming variables	☐	.../1
We should use problem domain names, instead of using generic identifier names	☐	.../1
The first rule of functions is that they should be as long as need, as far as they solve the problem	☑	1/1
It is OK that one function performs several and different not related actions	☑	1/1
It is a good practice of programming to have more than three arguments for a method	☑	1/1
We should use verbs for naming classes	☑	1/1
We should use nouns to name methods	☑	1/1
We should always comment every method, every class, and every attribute	☑	1/1
Vertical Formatting means indent every line of code	☐	0/1
Team rules must be adjusted to the rules agreed by every programmer's rule	☐	0/1

Answer with the name of the design pattern that matches its definition, please use FULL Capital letters for all answers: SINGLETON, ABSTRACT FACTORY, COMPOSITE, OBSERVER, STRATEGY, TEMPLATE METHOD,

8 de 10 puntos

21. Defines a family of algorithms, encapsulates each one, and make them interchangeable. This pattern lets the algorithm vary independently from the clients that use it *1/1

STRATEGY

22. Defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically *1/1

OBSERVER

23. Defines the skeleton of an algorithm in an operation, deferring some steps to subclasses. This pattern lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure *1/1

TEMPLATE METHOD

24. Composes objects into tree structures to represent part-whole hierarchies. This pattern lets clients treat individual objects and compositions of objects uniformly *1/1

COMPOSITE

25. Provides an interface for creating families of related or dependent objects without specifying their concrete classes *1/1

ABSTRACT FACTORY

26. Ensure a class only has one instance, and provide a global point of access to it *1/1

SINGLETON

Answer the code snippet questions according to the class of JavaScript: (FUNCTION, VARIABLE, CALL TO A FUNCTION)

27. let class=7996; * 1/1

VARIABLE

28. let app = Express() * 0/1

FUNCTION

29. * 0/1

```
app.get('/', function (req, res) {
  res.send('Hello World from nodeJS! without refreshing')
})
```

CALL TO A FUNCTION

30. UML stands for * 1/1

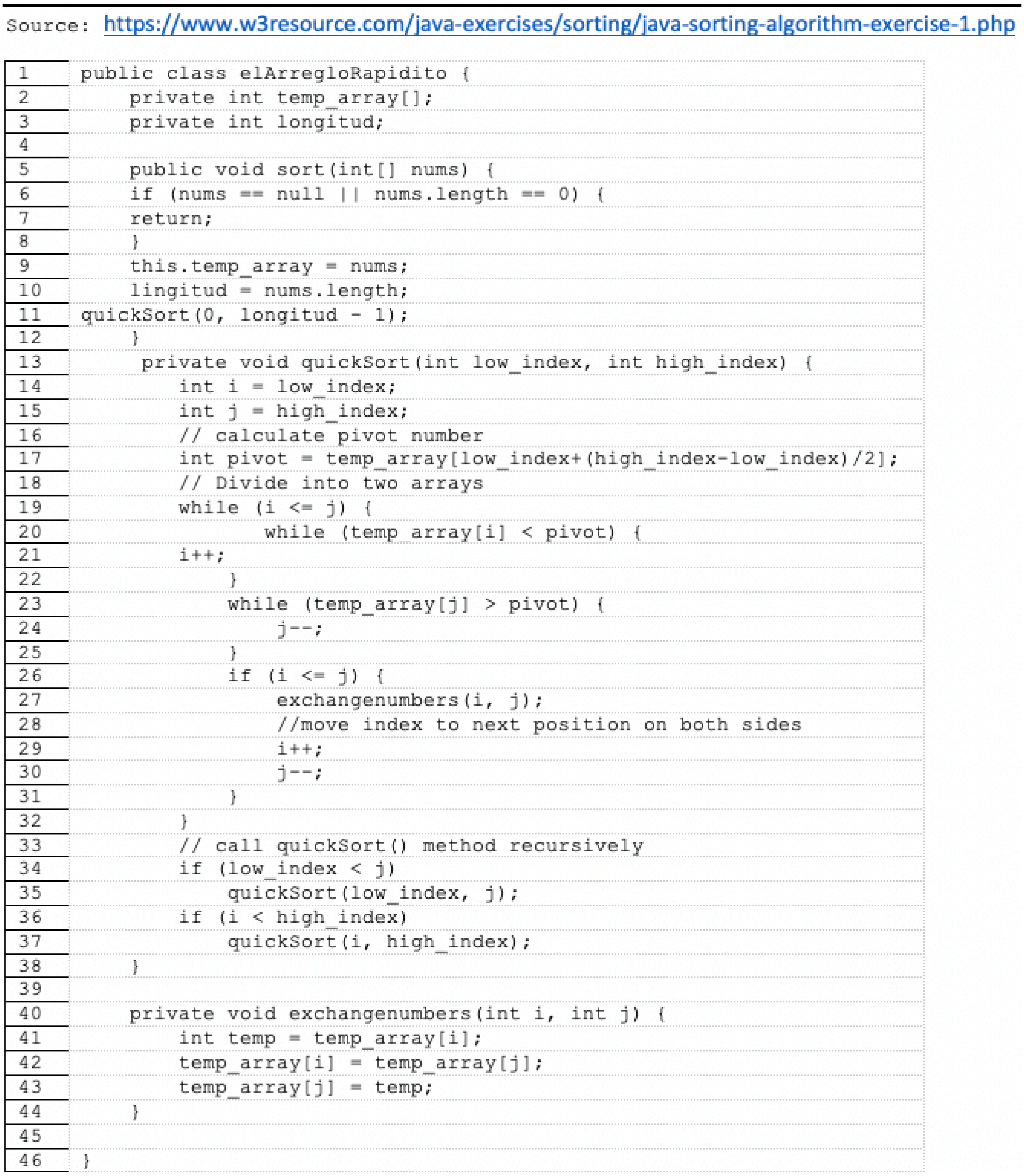
UNIFIED MODELING LANGUAGE

Clean Code

11 de 20 puntos

Find the 20 lines that make the following snippet less understandable. Only select the 20 ugly lines of code. If you select more than 20 lines of code. Your grade will be deducted based on those extra selected lines

Ugly QuickSort implementation in Java



Source: https://www.w3resource.com/java-exercises/sorting/java-sorting-algorithm-exercise-1.php

```
public class elArregloRapidito {
    private int temp_array[];
    private int longitud;

    public void sort(int[] nums) {
    if (nums == null || nums.length == 0) {
    return;
    }
    this.temp_array = nums;
    lingitud = nums.length;
quickSort(0, longitud - 1);
    }
     private void quickSort(int low_index, int high_index) {
        int i = low_index;
        int j = high_index;
        // calculate pivot number
        int pivot = temp_array[low_index+(high_index-low_index)/2];
        // Divide into two arrays
        while (i <= j) {
               while (temp_array[i] < pivot) {
        i++;
            }
            while (temp_array[j] > pivot) {
                j--;
            }
            if (i <= j) {
                exchangenumbers(i, j);
                //move index to next position on both sides
                i++;
                j--;
            }
        }
        // call quickSort() method recursively
        if (low_index < j)
            quickSort(low_index, j);
        if (i < high_index)
            quickSort(i, high_index);
    }

    private void exchangenumbers(int i, int j) {
        int temp = temp_array[i];
        temp_array[i] = temp_array[j];
        temp_array[j] = temp;
    }

}
```

Please, click the 20 ugly lines of code ONLY

	Column 1	Puntuación
1	◉	1/1
2	◉	1/1
3	◉	1/1
4	○	.../0
5	○	0/1
6	◉	1/1
7	◉	1/1
8	◉	.../0
9	○	.../0
10	◉	1/1
11	◉	1/1
12	◉	.../0
13	○	0/1
14	○	.../0
15	○	.../0
16	◉	.../0
17		

17	○	0/1
18	●	.../0
19	○	.../0
20	●	1/1
21	●	.../0
22	●	.../0
23	○	0/1
24	○	.../0
25	○	.../0
26	○	.../0
27	●	1/1
28	●	.../0
29	○	.../0
30	○	.../0
31	○	.../0
32	●	.../0
33	●	.../0
34	○	0/1
35		

36	○	0/1
37	○	0/1
38	●	1/1
39	○	.../0
40	○	.../0
41	●	1/1
42	○	0/1
43	○	0/1
44	○	.../0
45	○	.../0
46	○	.../0
46	○	.../0