Termpaper 2

Name	Student ID	Course	University
Andrii Yatsura	19387	Programming Languages and Paradigms	Wrocław Academy of Business

Recursion is a fundamental programming concept where a function calls itself to solve a problem. It's a flexible approach widely used in various paradigms like procedural, object-oriented, and functional programming. In this paper I'll explore the topic using Java programming language and a QuickSort algorithm.

Procedural

In this procedural programming example, the QuickSort algorithm is implemented using static methods in a class. Procedural programming focuses on writing procedures or functions that perform operations on the data. The `quickSort` and partition methods are defined as static, emphasizing a straightforward, step-by-step procedural approach to sorting the array. This style is characterized by a sequence of imperative steps and direct manipulation of data.

```java
class QuickSort {
    static void quickSort(int[] arr, int start, int end) {
        if (start < end) {
            int pIndex = partition(arr, start, end);
            quickSort(arr, start, pIndex - 1); // Recursive call for left partition
            quickSort(arr, pIndex + 1, end);   // Recursive call for right partition
        }
    }

    static int partition(int[] arr, int start, int end) {
        int pivot = arr[end];
        int i = start - 1;

        for (int j = start; j < end; j++) {
            if (arr[j] <= pivot) {
                i++;
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }

        int temp = arr[i + 1];
        arr[i + 1] = arr[end];
        arr[end] = temp;
        return i + 1;
    }
}
```

Object-Oriented

In the object-oriented paradigm, the QuickSort algorithm is encapsulated within a class. The key concept here is encapsulation and the use of class methods to define behaviors. The `QuickSortOOP` class contains non-static methods to sort an array, and it maintains the state related to the sorting process. The recursive logic is encapsulated within the class, demonstrating the object-oriented principles of bundling data and methods that operate on the data within one unit.

```java
class QuickSortOOP {
    void sort(int[] arr) {
        quickSort(arr, 0, arr.length - 1);
    }

    private void quickSort(int[] arr, int start, int end) {
        if (start < end) {
            int pIndex = partition(arr, start, end);
            quickSort(arr, start, pIndex - 1);
            quickSort(arr, pIndex + 1, end);
        }
    }

    private int partition(int[] arr, int start, int end) {
        int pivot = arr[end];
        int i = start - 1;

        for (int j = start; j < end; j++) {
            if (arr[j] <= pivot) {
                i++;
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }

        int temp = arr[i + 1];
        arr[i + 1] = arr[end];
        arr[end] = temp;
        return i + 1;
    }
}
```

Functional

For the functional programming example, I'll use Python as it is slightly more suitable for this paradigm.

In this implementation, the `quick_sort` function takes an array and divides it into three parts: left, middle, and right, based on the pivot element. It then recursively sorts the left and

right parts. This approach ensures that the original array is not modified, complying with one of the main principles of the functional paradigm - immutability.

```python
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    else:
        pivot = arr[len(arr) // 2]
        left = [x for x in arr if x < pivot]
        middle = [x for x in arr if x == pivot]
        right = [x for x in arr if x > pivot]
        return quick_sort(left) + middle + quick_sort(right)
```