Java Menu-Driven Program for Multiple Operations

This document contains a Java program that provides a menu-driven approach to perform operations on integers, arrays, recursion, sorting algorithms, and stack implementations. The program demonstrates modular design using static methods and classes.

```
import java.lang.String;

import java.lang.System;

import java.util.Scanner;

import java.util.Stack;


class Test {

  static void m() {

    System.out.println();

    System.out.println("*********Main Menu**********");

    System.out.println("Option1: Two Values programs");

    System.out.println("Option2: Array programs");

    System.out.println("Option3: Recursive programs");

    System.out.println("Option4: sorting programs");

    System.out.println("Option5: stack programs");

    System.out.println("Option0: Exit the program");

  }


  static void valuesList() {

    System.out.println();

    System.out.println("List of value programs:");

    System.out.println("Option0: Exits the value programs");

    System.out.println("Option1: Displays Two values program without return");
```

```
        System.out.println("Option2: Displays sum of Two values program with return");
        System.out.println("Option3: Swaps Two values using temporary variable");
        System.out.println("Option4: Swaps Two values using arithmetic operators using + and
-");
        System.out.println("Option5: Swaps Two values using operators * and /");
        System.out.println("option6: Swaps Two values using bitwise operator ^");
        System.out.println("Option7: Swaps Two values using expression");
        System.out.println("Option8: Finds max using ternary operator and returns");
        System.out.println("Option9: Finds max using predefined static method max() and
returns");
    }

    static void m1(int a, int b) {
        System.out.println("a value is:" + a + " " + "b value is:" + b);
    }

    static int m2(int a, int b) {
        return a + b;
    }

    static void swap1(int a, int b) {
        int temp = a;
        a = b;
        b = temp;
        System.out.println("After swap a and b values:" + a + " " + b);
    }
```

```
static void swap2(int a, int b) {
    a = a + b;
    b = a - b;
    a = a - b;
    System.out.println("After swap a and b values:" + a + " " + b);
}

static void swap3(int a, int b) {
    a = a * b;
    b = a / b;
    a = a / b;
    System.out.println("After swap a and b values:" + a + " " + b);
}

static void swap4(int a, int b) {
    a = a ^ b;
    b = a ^ b;
    a = a ^ b;
    System.out.println("After swap a and b values:" + a + " " + b);
}

static void swap5(int a, int b) {
    a = (a + b) - (b = a);
    System.out.println("After swap a and b values:" + a + " " + b);
}
```

```
  static int maxVersion1(int a, int b) {
     return (a > b) ? a : b;
  }

  static int maxVersion2(int a, int b) {
     return Math.max(a, b);
  }

  // -------- Array Methods --------
  static void arrayMenu() {
     System.out.println();
     System.out.println("List of Array Programs:");
     System.out.println("Option0: Exits the array programs");
     System.out.println("Option1: print the array elements without return");
     System.out.println("Option2: print the array elements with return");
     System.out.println("Option3: return the array by multiplying with 2");
     System.out.println("Option4: return the sum of array elements");
     System.out.println("Option5: return the max of the array");
     System.out.println("Option6: return the min of the array");
     System.out.println("Option7: returns the sum of elements that are greater than 80 and
equal to 40");
     System.out.println("Option8: returns the elements divisible by both 2 and 3");
  }

  static void m3(int[] arr) {
     for (int a : arr) {
```

```
        System.out.print(a + " ");
    }
}

static int[] m4(int[] arr) {
    return arr;
}

static int[] m5(int[] arr) {
    int[] d = new int[arr.length];
    for (int i = 0; i < arr.length; i++) {
        d[i] = arr[i] * 2;
    }
    return d;
}

static int m6(int[] arr) {
    int sum = 0;
    for (int j : arr) {
        sum += j;
    }
    return sum;
}

static int m7(int[] arr) {
    int max = arr[0];
```

```
    for (int j : arr) {
        if (j > max) max = j;
    }
    return max;
}

static int m8(int[] arr) {
    int min = arr[0];
    for (int j : arr) {
        if (j < min) min = j;
    }
    return min;
}

static int sumOfElementsGreater80(int[] arr) {
    int sum = 0;
    for (int j : arr) {
        if (j > 80 || j == 40) sum += j;
    }
    return sum;
}

static int[] elementsDivisibleBy2And3(int[] arr) {
    int count = 0;
    for (int j : arr) {
        if (j % 2 == 0 && j % 3 == 0) count++;
```

```
    }
    int[] result = new int[count];
    int k = 0;
    for (int j : arr) {
        if (j % 2 == 0 && j % 3 == 0) result[k++] = j;
    }
    return result;
}

// -------- Recursion --------
static void recursionMenu() {
    System.out.println();
    System.out.println("List of Recursive programs");
    System.out.println("Option0: Exits the recursive programs");
    System.out.println("Option1: Sum of n elements");
    System.out.println("Option2: Factorial of given number");
    System.out.println("Option3: Numbers from 1 to n");
    System.out.println("Option4: Numbers from n to 1");
    System.out.println("Option5: Displays characters from z to my name first letter");
}

static int sum(int n) {
    if (n == 0) return 0;
    return n + sum(n - 1);
}
```

```
static int fact(int n) {
    if (n == 0) return 1;
    return n * fact(n - 1);
}

static void nNumbers(int n) {
    if (n < 1) return;
    nNumbers(n - 1);
    System.out.print(n + " ");
}

static void nNumbersReverse(int n) {
    if (n < 1) return;
    System.out.print(n + " ");
    nNumbersReverse(n - 1);
}

static void reverseLower(char ch) {
    if (ch == 'g') return;
    System.out.print(ch + " ");
    reverseLower((char) (ch - 1));
}

static void reverseUpper(char ch) {
    if (ch == 'G') return;
    System.out.print(ch + " ");
```

```
    reverseUpper((char) (ch - 1));
}

// -------- Sorting --------
static void sort() {
    System.out.println();
    System.out.println("Sorting Menu");
    System.out.println("Option 0: Exits the sorting programs");
    System.out.println("Option 1: Bubble sort in ascending order");
    System.out.println("Option 2: Bubble sort in descending order");
    System.out.println("Option 3: Selection sort in ascending order");
    System.out.println("Option 4: Selection sort in descending order");
}

static int[] aBubbleSort(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;
            }
        }
    }
    return arr;
}
```

```
static int[] dBubbleSort(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;
         }
      }
   }
   return arr;
}

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

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

// ---------- Stack ----------
class StackArray {
    int size = 6;
    int[] data;
    int Top = -1;

    StackArray() {
        data = new int[size];
    }

    static void stackMenu() {
```

```
        System.out.println();
        System.out.println("Stack Menu");
        System.out.println("Option 0: Exits the stack programs");
        System.out.println("Option 1: To check whether stack is empty or not");
        System.out.println("Option 2: To print the size of stack");
        System.out.println("Option 3: Push the elements in to stack using user-defined
method");
        System.out.println("Option 4: Pop the element from stack using user-defined method");
        System.out.println("Option 5: Return the top element using user-defined method");
        System.out.println("Option 6: search for the element in stack");
        System.out.println("Option 7: Displays the elements in stack");
        System.out.println("Option 8: Implementation of stack operations using Pre-defined
stack class");
        System.out.println("Option 9: Implementation of stack operations using Pre-defined
stack class for similar elements");
    }

    boolean isEmpty() {
        return Top == -1;
    }

    int size() {
        return Top + 1;
    }

    void addStack(int value) {
        if (size() == data.length) {
```

```
        System.out.println("Stack is full");
        return;
    }
    Top++;
    data[Top] = value;
}

void show() {
    if (isEmpty()) {
        System.out.println("stack is empty");
        return;
    }
    for (int i = 0; i <= Top; i++) {
        System.out.print(data[i] + " ");
    }
}

int deleteTop() {
    if (isEmpty()) {
        System.out.println("stack is empty");
        return -1;
    }
    int temp = data[Top];
    Top--;
    return temp;
}
```

```
    int peek() {
        if (isEmpty()) {
            System.out.println("stack is empty");
            return -1;
        }
        return data[Top];
    }

    boolean search(int target) {
        if (isEmpty()) {
            System.out.println("stack is empty");
            return false;
        }
        for (int i = 0; i <= Top; i++) {
            if (data[i] == target) return true;
        }
        return false;
    }
}

// ---------- Main ----------
public class All1 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int choice;
```

```
do {
   Test.m();
   System.out.print("Enter your choice: ");
   choice = sc.nextInt();

   switch (choice) {
      case 1:
         // values programs
         break;
      case 2:
         // arrays
         break;
      case 3:
         // recursion
         break;
      case 4:
         // sorting
         break;
      case 5:
         // stack
         break;
      case 0:
         System.out.println("Thank you! Exiting the program...");
         break;
   }
} while (choice != 0);
```

```
    }
}
```