Object Oriented Programming - Lab 2

Date : 25-08-2021

1. Write a Java program to implement Stack data structure. The input to your program should be hardcoded into two arrays viz. operation[] and input[].
    a. Array operation [ ] - should contain values between 1-4. The action to be done for operation [i]:
        1 - push input [i] to the stack 1
        2 - pop from stack 1
        3 - copy values of stack 1 to stack 2 in correct order
        4 - print contents of stack 1 and 2
    b. Array input [ ] - should contain float values to be pushed to stack.

    For example:
    operation [] = { 1, 1, 1, 2, 1, 2, 3, 4}
    input [] = { 2.1, 2.2, 2.3, 0, 2.4, 0, 0, 0}
    Operations to be done are in the order:
    - push(stack1, 2.1)
    - push(stack1, 2.2)
    - push(stack1, 2.3)
    - pop(stack1)
    - push(stack1, 2.4)
    - pop(stack1)
    - copy(stack1, stack2)
    - display(stack1), display(stack2)

    Note: *your program should not read any input from the user. Use the above example as the hard-coded input when you are submitting the assignment.*

2. Create a simple java program to store/find employee details in a corporation.
    - Employee has employee-id, name, role (example: manager, supervisor, ceo, etc)
    - Create a composite data type (***class***) for the employee. No need for methods. Use 'class' the same way you would use 'structs' in c-lang.
    - Create the following procedural (***static***) functions
        - **createEmployee** - creates an employee instance and stores/adds it to whatever data structure of your choice (**Note:** the simplest solution is to use an array and assume that the max-size is 5).
        - **findEmployeeById** - takes as parameter an id and finds the employee
        - **findEmployeeByRole** - takes as parameter the 'role' and returns an array of employees matching the role
        - **findEmployeeByName** - takes as parameter the 'name' and returns an array of employees having that name
          (**Note:** there could be two employees with the same name)

    - **printEmployeeDetail** - takes as parameter an employee-instance and pretty-prints the details.
      (**Warning:** Except for this function, there should be no other print statements related to the employee object, anywhere else in the program. There may be print statements in other places such as 'No employee found' etc. But whenever/wherever you want to print an employee object, You MUST use this function only)
- From the **main** function, call the above createEmployee function to input the details of the following employees. You don't have to receive the details from the user. You may hard-code the following directly in main.

ID	Name	Role
1	Jack	Manager
2	Jill	Engineer
3	Tom	Manager
4	Jack	Supervisor

- You need to show examples of all 'find' functions. From main, Call the find functions to find employee(s) and use the printEmployeeDetail function to print. Example calls shown below:
    - findEmployeeById(3)
    - findEmployeeByRole("Manager")
    - findEmployeeByName("Jack")

  (**Note:** Some find functions may return multiple employee objects (array) as a result. so, you might need to iterate over the array to print)

What to turn in

1. Two java files
    a. StackMain.java
    b. EmployeeRecordsMain.Java
2. Two images (Screenshots) of running your program and the output (JPG/PNG)
    a. Stack-output.jpg/png
    b. Employee-output.jpg/png
3. **DO NOT ZIP**. Submit the files directly.