CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Kết thúc học phần(đề 04)

Name: chìu cắm minh

ID: 50133

Class: IT19A2B

Bài làm

Câu 1.

```
*************** Class Employee ************************

/**
 * a)Khai báo lớp nhân viên Employee.
 */
public class Employee {
    private int code;
    private String fullName;
    private double salary;

    public Employee(int code, String fullName, double salary) {
        this.code = code;
        this.fullName = fullName;
        this.salary = salary;
    }

    public double getSalary() {
        return salary;
    }

    @Override
    public String toString() {
        return "Code: " + code + "; FullName: " + fullName + ";
Salary: " + salary;
    }

}


*************** Class SinglyLinkedList ************************
import java.util.NoSuchElementException;
```

```java
/**
 * lớp danh sách liên kết SinglyLinkedList để lưu các nhân viên
 */
public class SinglyLinkedList {
    private EmployeeNode head;

    // b) Viết hàm thêm nhân viên vào đầu danh sách: addFirst, thêm
vào 5
    // nhân viên.
    public void addFirst(Employee employee) {
        var newNode = new EmployeeNode(employee);
        newNode.setNext(this.head);
        this.head = newNode;
    }

    // c) Viết hàm hiển thị danh sách các nhân viên: printList
    public void printList() {
        var currentNode = this.head;
        while (currentNode != null) {
            System.out.println(currentNode.getEmployee().toString());
            currentNode = currentNode.getNext();
        }
    }

    // d)Viết hàm xóa nhân viên đầu danh sách: removeFromFront
    public void removeFromFront() {
        if (this.isEmpty()) {
            throw new NoSuchElementException();
        }
        this.head = this.head.getNext();
    }

    // e)Viết hàm tìm kiếm linearSearch thông tin các nhân viên theo
mức
    // lương lớn hơn 5000: searchByFee
    public void searchByFee() {
        var currentNode = this.head;
        while (currentNode != null) {
            if (currentNode.getEmployee().getSalary() > 5000) {

System.out.println(currentNode.getEmployee().toString());
            }
            currentNode = currentNode.getNext();
        }
    }
```

```
    // Hàm check rỗng
    private boolean isEmpty() {
        return this.head == null;
    }

}
```

********** Class EmployeeNode **********

```
public class EmployeeNode {
    private Employee employee;
    private EmployeeNode next;

    public EmployeeNode(Employee employee) {
        this.employee = employee;
    }

    public void setNext(EmployeeNode next) {
        this.next = next;
    }

    public EmployeeNode getNext() {
        return next;
    }

    public Employee getEmployee() {
        return employee;
    }

}
```

********** Class Test **********

```
/**
 * Câu 1: Gỉa sử cho một danh sách liên kết đơn mà mỗi phần tử chứa
thông tin về
 * một đối tượng nhân viên Employee bao gồm các thuộc tính mã code
(int), tên
 * fullName (String) và mức lương salary (double)
 */
public class Test {
    public static void main(String[] args) {
        Employee employee1 = new Employee(1111, "minhchiu1", 4323);
```

```
        Employee employee2 = new Employee(2222, "minhchiu2", 5442);
        Employee employee3 = new Employee(3333, "minhchiu3", 6352);
        Employee employee4 = new Employee(4444, "minhchiu4", 233);
        Employee employee5 = new Employee(5555, "minhchiu5", 832);


        SinglyLinkedList employees = new SinglyLinkedList();
        employees.addFirst(employee4);
        employees.addFirst(employee2);
        employees.addFirst(employee3);
        employees.addFirst(employee5);
        employees.addFirst(employee1);
        System.out.println("Danh sách nhân viên: ");
        employees.printList();
        employees.removeFromFront();
        employees.removeFromFront();
        System.out.println("\nDanh sách nhân viên sau khi
removeFromFront 2 lần: ");
        employees.printList();
        System.out.println("\nThông tin các nhân viên theo mức lương
lớn hơn 5000: ");
        employees.searchByFee();
    }
}
```

Câu 2.

```
********** Class Test **********
 /**
 * Câu 2: Giả sử cho cấu trúc dữ liệu hàng đợi Queue với hàm enqueu(a)
là hàm
 * thực hiện nạp a vào hàng đợi và hàm dequeue() là hàm thực hiện lấy
phần tử ra
 * khỏi hàng đợi.
 * */

/**
 * ❖ Giả sử cho dãy các thao tác sau đây, biết rằng hàng đợi ban đầu
được khởi
 * tạo rỗng: enqueu(5), enqueu(3), dequeue(), enqueu(2), enqueu(8),
dequeue(),
 * enqueu(9), enqueu(1), dequeue(), enqueu(7), enqueu(6), dequeue(),
dequeue(),
 * enqueu(4), dequeue(), dequeue(). */
```

```
/**
 * ❖ Hãy viết ra dãy các phần tử của hàng đợi (chỉ rõ vị trí đầu và
cuối của
 * hàng đợi) sau khi thực hiện mỗi thao tác.
 *
 */
public class Test {
     public static void printInfoList(MyQueue myQueue) {
        System.out.println("Danh sách các phần tử: ");
        myQueue.show();

        System.out.println("Phần tử đầu tiên: " + myQueue.getHead());
         System.out.println("Phần tử cuối cùng: " +
myQueue.getTail());
        System.out.println("----------------------------");
    }


    public static void main(String[] args) {
        MyQueue myQueue = new MyQueue();
        myQueue.enqueu(5);
        printInfoList(myQueue);

        myQueue.enqueu(3);
        printInfoList(myQueue);

        myQueue.dequeue();
        printInfoList(myQueue);

        myQueue.enqueu(2);
        printInfoList(myQueue);

        myQueue.enqueu(8);
        printInfoList(myQueue);

        myQueue.dequeue();
        printInfoList(myQueue);

        myQueue.enqueu(9);
        printInfoList(myQueue);

        myQueue.enqueu(1);
        printInfoList(myQueue);
```

```
        myQueue.dequeue();
        printInfoList(myQueue);

        myQueue.enqueu(7);
        printInfoList(myQueue);

        myQueue.enqueu(6);
        printInfoList(myQueue);

        myQueue.dequeue();
        printInfoList(myQueue);


        myQueue.dequeue();
        printInfoList(myQueue);

        myQueue.enqueu(4);
        printInfoList(myQueue);

        myQueue.dequeue();
        printInfoList(myQueue);

        myQueue.dequeue();
        printInfoList(myQueue);
    }
}

************** Class MyQueue ********************

import java.util.EmptyStackException;

public class MyQueue {

    public class Node {
        private int value;
        private Node next;

        public Node(int value) {
            this.value = value;
        }
    }

    private Node head;
    private Node tail;

    public boolean enqueu(int value) {
```

```
    Node newNode = new Node(value);
    if (isEmpty())
        head = tail = newNode;
    else {
        tail.next = newNode;
        tail = newNode;
    }
    return true;
}

public int dequeue() {
    var result = head.value;
    if (isEmpty())
        throw new EmptyStackException();
    else if (head == tail) {
        head = tail = null;
    } else {
        head = head.next;
    }
    return result;
}

private boolean isEmpty() {
    return head == null;
}

public void show() {
    if (isEmpty()) {
        System.err.println("Queue is Empty!");
        return;
    }
    var currentNode = head;
    while (currentNode != null) {
        System.out.print(currentNode.value + " ");
        currentNode = currentNode.next;
    }
    System.out.println();
}

public int getHead() {
    if (isEmpty()) {
        System.err.println("Queue is Empty!");
        return -1;
    }
    return head.value;
}
```

```
    public int getTail() {
        if (isEmpty()) {
            System.err.println("Queue is Empty!");
            return -1;
        }
        return tail.value;
    }
}
```