**Trường Công nghệ thông tin và Truyền thông Đại Học Bách Khoa Hà Nội**

A red and yellow sign with a yellow and white background

Description automatically generated

**Đề tài: Mô tả các thao tác trên ngăn xếp, hàng đợi, danh sách**

**Môn: Lập trình hướng đối tượng**

GV hướng dẫn: TS. Nguyễn Thị Thu Trang

**Nhóm 21**

Nguyễn Mạnh Quân 20225758

Nguyễn Anh Quân 20225907

Nguyễn Hồng Quân 20225908

Phạm Viết Anh Quân 20215288

Dương Đăng Quang 20210716

**Phân chia công việc nhóm**

|  |  |  |
| --- | --- | --- |
| Nguyễn Mạnh Quân | * Xử lý thao tác với class List * Làm báo cáo | Hoàn thành đúng tiến độ |
| Nguyễn Anh Quân | * Thiết kế GUI, Menu và nút Quit * Thiết kế use case và class diagram | Hoàn thành đúng tiến độ |
| Nguyễn Hồng Quân | * Xử lý thao tác với class Data Structure * Làm báo cáo | Hoàn thành đúng tiến độ |
| Phạm Viết Anh Quân | * Xử lý thao tác với class Stack * Thiết kế slide | Hoàn thành đúng tiến độ |
| Dương Đăng Quang | * Xử lý thao tác với class Queue * Thiết kế slide | Hoàn thành đúng tiến độ |
| Công việc cùng thực hiện | * Lên ý tưởng cho project | Hoàn thành đúng tiến độ |

* **Link project:** [**anhquan-ngg/LTHDT.20241-21**](https://github.com/anhquan-ngg/LTHDT.20241-21)

1. **Giới thiệu bài toán**
   1. ***Giới thiệu đề tài***

**Stack**, **queue** và **list** là những cấu trúc dữ liệu cơ bản trong khoa học máy tính, đóng vai trò quan trọng trong việc tổ chức và quản lý dữ liệu hiệu quả. Thông qua chương trình này, sinh viên sẽ thực hiện và hiểu rõ các thao tác cơ bản trên những cấu trúc này, bao gồm tạo, thêm, sắp xếp, tìm kiếm và xóa phần tử.

Nhận thấy rằng các kiến thức từ môn học Lập trình hướng đối tượng có thể được áp dụng để mô tả các thao tác trên ngăn xếp, hàng đợi, danh sách. Vì vậy, nhóm 21 chúng em thực hiện đề tài xây dựng chương trình ”Mô tả các thao tác trên ngăn xếp, hàng đợi, danh sách ” nhằm học hỏi, củng cố và phát triển các kĩ năng về thực hành lập trình hướng đối tượng.

* 1. ***Các tính năng***
* Tính năng chính của chương trình là mô tả các thao tác cơ bản của 3 cấu trúc Stack, Queue, List.
* Người dùng sẽ có thể chọn thực hiện các thao tác tạo, xóa, tìm phần tử, sắp xếp phần tử(đối với List), thêm phần tử vào cấu trúc mà người dùng lựa chọn
* Chương trình sẽ thực hiện và xuất ra kết quả cho người dùng
* Người dùng có thể Back để quay lại Menu, và ấn nút Quit để thoát khỏi chương trình
  1. ***Công cụ sử dụng***
* Ngôn ngữ sử dụng: Java
* Version Control: Git (Github)
* GUI: Swing

1. **Phân Tích Biểu Đồ Use Case**
   1. **Biểu đồ usecase**

**A diagram of a company

Description automatically generated**

* 1. **Đặc tả Use case**

Khi sử dụng chương trình, người dùng có thể thực hiện nhiều thao tác trên đó:

* **Chọn cấu trúc (Choose Stack, Queue, List):** Khi mở chương trình, người dùng được yêu cầu chọn 1 trong 3 cấu trúc: Stack, Queue, List.
* **Chọn thao tác (Choose Operation):** Sau khi chọn cấu trúc, người dùng có thể chọn các thao tác: Thêm phần tử **(Insert value),** xóa phần tử **(Delete value),** tìm phần tử **(Find value),** Xem mọi phần tử **(Show all values),** Xóa mọi phần tử **(Clear all values),** sắp xếp phần tử **(Sort list).**
* **Mở menu trợ giúp (Show help):** Người dùng có thể mở trợ giúp ở trên thanh menu để đọc về thông tin ứng dụng
* **Thoát chương trình (Exit program):** Người dùng có thể nhấn vào nút Quit ở trên thanh menu để thoát ứng dụng.

1. **Biểu đồ Class Diagram** 
   1. ***Biểu đồ Chung Class Diagram***

***A screenshot of a computer

Description automatically generated***

* 1. **Phân tích biểu đồ chung Class Diagram:**

**Mối quan hệ giữa các lớp**

1. DataStructure:

- Là lớp cha (abstract class hoặc base class) cho các lớp con List, Queue, và Stack.

- Các lớp con kế thừa các phương thức và thuộc tính từ lớp DataStructure.

2. List, Queue, và Stack:

- Đây là các lớp con của DataStructure, mỗi lớp mở rộng chức năng của lớp cha để cung cấp các thao tác cụ thể cho danh sách, hàng đợi, và ngăn xếp.

- List có thêm phương thức sort() để sắp xếp danh sách.

- Queue có các thuộc tính như queue, front, rear và phương thức isEmpty() để kiểm tra xem hàng đợi có trống không.

- Stack có các thuộc tính như stack, pointer và phương thức isEmpty() để kiểm tra ngăn xếp có trống không.

3. ListGUI, QueueGUI, và StackGUI:

- Các lớp GUI (giao diện người dùng) tương ứng với các lớp List, Queue, và Stack.

- Mỗi lớp GUI có mối quan hệ tập hợp (aggregation) với lớp dữ liệu tương ứng:

- ListGUI sử dụng List.

- QueueGUI sử dụng Queue.

- StackGUI sử dụng Stack.

4. MainGUI:

- Là lớp chính, có vai trò quản lý và hiển thị giao diện tổng thể.

- Có mối quan hệ tổng hợp (composition) với các lớp GUI (ListGUI, QueueGUI, và StackGUI), vì nó sử dụng chúng để cung cấp chức năng giao diện.

Kết luận:

- DataStructure là lớp cha, các lớp List, Queue, và Stack kế thừa từ nó.

- Các lớp GUI (ListGUI, QueueGUI, StackGUI) tương ứng với các lớp dữ liệu cụ thể.

- MainGUI là lớp chính, điều phối hoạt động giữa các thành phần.

**Cách thức thực hiện các phương thức quan trọng trong các lớp List, Queue, và Stack:**

1. Lớp List

* insert(int value):

Chèn giá trị vào danh sách bằng cách thêm vào ArrayList.

Sử dụng phương thức add() của ArrayList

public void insert(int value) {

    list.add(value);

    System.out.println("Added " + value + " to list.");

}

* delete(int value):

Xóa giá trị nếu tồn tại trong danh sách.

Sử dụng phương thức remove() và kiểm tra xem giá trị có tồn tại hay không.

public void delete(int value) {

    if (list.remove((Integer) value)) {

        System.out.println("Removed " + value + " from list.");

    } else {

        System.out.println("Value not found in list.");

    }

}

* find(int value):

Tìm vị trí của giá trị trong danh sách.

Sử dụng phương thức indexOf()

public int find(int value) {

    return list.indexOf(value);

}

* sort():

Sắp xếp danh sách theo thứ tự tăng dần.

Sử dụng Collections.sort().java

public void sort() {

    Collections.sort(list);

    System.out.println("List sorted");

}

2. Lớp Queue

* insert(int value):

Thêm giá trị vào cuối hàng đợi nếu chưa đầy.

Sử dụng con trỏ rear để quản lý vị trí chèn.

public void insert(int value) {

    if (rear < size) {

        queue[rear++] = value;

        System.out.println("Enqueued " + value + " to queue.");

    } else {

        System.out.println("Queue is full.");

    }

}

* delete():

Xóa giá trị ở đầu hàng đợi nếu không rỗng.

Tăng con trỏ front sau khi xóa.

public void delete() {

    if (front < rear) {

        System.out.println("Dequeued " + queue[front++] + " from queue.");

    } else {

        System.out.println("Queue is empty.");

    }

}

* find(int value):

Tìm vị trí của giá trị trong hàng đợi từ front đến rear.

public int find(int value) {

    for (int i = front; i < rear; i++) {

        if (queue[i] == value) {

            return i;

        }

    }

    return -1;

}

* isEmpty():

Kiểm tra xem hàng đợi có rỗng không.

public boolean isEmpty() {

    return front == 0 && rear == 0;

}

3. Lớp Stack

* insert(int value):

Chèn giá trị vào đầu ngăn xếp nếu chưa đầy.

Tăng con trỏ pointer và thêm giá trị.

public void insert(int value) {

    if (pointer < size - 1) {

        stack[++pointer] = value;

        System.out.println("Pushed " + value + " to stack.");

    } else {

        System.out.println("Stack is full.");

    }

}

* delete():

Xóa giá trị ở đầu ngăn xếp nếu không rỗng.

Giảm con trỏ pointer.java

public void delete() {

    if (pointer >= 0) {

        System.out.println("Popped " + stack[pointer--] + " from stack.");

    } else {

        System.out.println("Stack is empty.");

    }

}

* find(int value):

Tìm vị trí của giá trị trong ngăn xếp từ 0 đến pointer.java

public int find(int value) {

    for (int i = 0; i <= pointer; i++) {

        if (stack[i] == value) {

            return i;

        }

    }

    return -1;

}

* isEmpty():

Kiểm tra xem ngăn xếp có rỗng không.

public boolean isEmpty() {

    return pointer == -1;

}

**Cách thức thực hiện các phương thức quan trọng trong các lớp ListGUI, QueueGUI, StackGUI, và MainGUI:**

1. File ListGUI.java

- updateDisplay()

Cách thực hiện:

Xóa nội dung cũ trong displayArea bằng setText("").

Lấy danh sách hiện tại từ đối tượng list bằng phương thức show() và hiển thị trong displayArea bằng append().

- updateDisplay(int idx)

Cách thực hiện:

Xóa nội dung cũ trong displayArea.

Hiển thị thông báo "Found at [vị trí]" khi tìm thấy phần tử trong danh sách.

- createNorthPanel()

Cách thực hiện:

Tạo một JPanel chứa các nút bấm và JTextField.

Gắn các ActionListener cho các nút:

Insert: Lấy giá trị từ inputField, chèn vào danh sách bằng list.insert(num), sau đó cập nhật giao diện.

Delete: Kiểm tra giá trị nhập vào, nếu tồn tại trong danh sách thì xóa bằng list.delete(num), nếu không hiển thị thông báo lỗi.

Clear: Xóa toàn bộ danh sách bằng list.clear() và cập nhật giao diện.

Find: Tìm vị trí của phần tử trong danh sách bằng list.find(num) và hiển thị kết quả.

Sort: Sắp xếp danh sách bằng list.sort() và cập nhật giao diện.

2. File MainGUI.java

- main(String[] args)

Cách thực hiện:

Tạo một JFrame chính để hiển thị các lựa chọn cho người dùng.

Thêm các nút bấm vào giao diện:

Stack: Mở giao diện quản lý stack bằng cách gọi new StackGUI().

Queue: Mở giao diện quản lý queue bằng cách gọi new QueueGUI().

List: Mở giao diện quản lý list bằng cách gọi new ListGUI().

Help: Hiển thị hướng dẫn sử dụng bằng JOptionPane.

Exit: Xác nhận thoát chương trình với JOptionPane trước khi gọi System.exit(0).

3. File QueueGUI.java

- updateDisplay()

Cách thực hiện:

Xóa nội dung cũ trong displayArea.

Hiển thị trạng thái hiện tại của hàng đợi bằng cách gọi queue.show().

- updateDisplay(int idx)

Cách thực hiện:

Xóa nội dung cũ trong displayArea.

Hiển thị vị trí của phần tử được tìm thấy trong hàng đợi (nếu có).

- createNorthPanel()

Cách thực hiện:

Tạo một JPanel chứa các nút điều khiển và JTextField.

Gắn ActionListener cho các nút:

Push: Chèn phần tử vào hàng đợi bằng queue.insert(num) và cập nhật giao diện.

Pop: Xóa phần tử khỏi hàng đợi bằng queue.delete(), nếu hàng đợi trống thì hiển thị thông báo lỗi.

Clear: Xóa toàn bộ hàng đợi bằng queue.clear() và cập nhật giao diện.

Find: Tìm vị trí phần tử trong hàng đợi bằng queue.find(num) và hiển thị kết quả.

Back: Đóng giao diện hiện tại bằng dispose().

4. File StackGUI.java

- updateDisplay()

Xóa nội dung cũ trong displayArea.

Hiển thị trạng thái hiện tại của stack bằng cách gọi stack.show().

- updateDisplay(int idx)

Cách thực hiện:

Xóa nội dung cũ trong displayArea.

Hiển thị vị trí của phần tử được tìm thấy trong stack (nếu có).

- createNorthPanel()

Cách thực hiện:

Tạo một JPanel chứa các nút điều khiển và JTextField.

Gắn ActionListener cho các nút:

Push: Thêm phần tử vào stack bằng stack.push(num) và cập nhật giao diện.

Pop: Xóa phần tử khỏi stack bằng stack.pop(), nếu stack trống thì hiển thị thông báo lỗi.

Clear: Xóa toàn bộ stack bằng stack.clear() và cập nhật giao diện.

Find: Tìm vị trí phần tử trong stack bằng stack.find(num) và hiển thị kết quả.

Back: Đóng giao diện hiện tại bằng dispose().

1. **Kết luận**

* **Ưu điểm:**
* Đáp ứng được các chức năng cơ bản của chương trình.
* Người dùng có thể thao tác với các kiểu cấu trúc Stack, Queue, List và nhận được kết quả thực hiện thao tác từ chương trình.
* **Nhược điểm:**
* Chương trình chưa có animation minh họa.
* Thiết kế UI còn có thể cải thiện hơn.