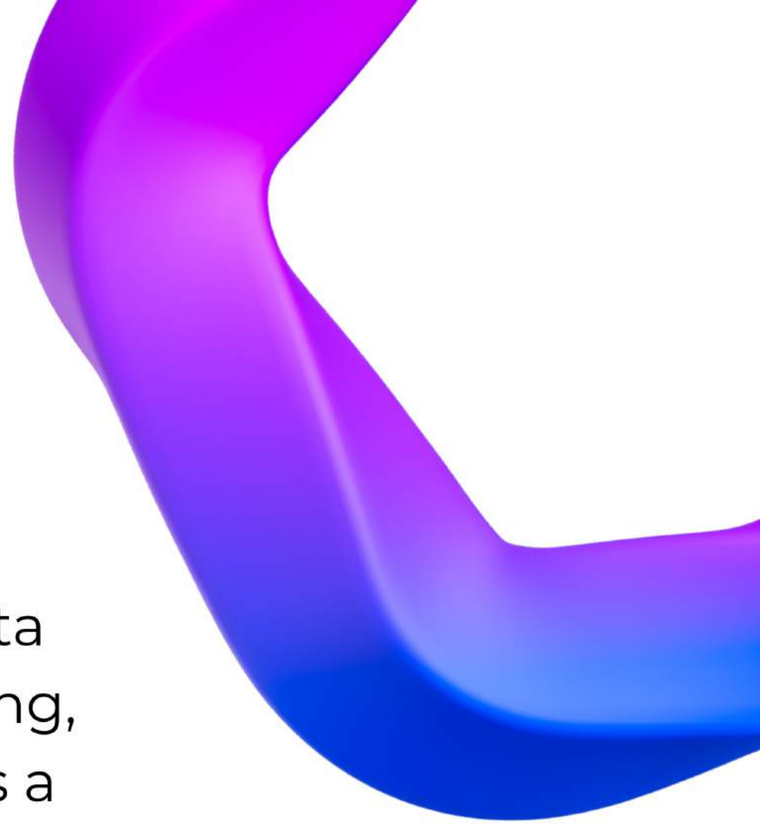
introduce



Student Management System Overview
This system allows for managing students using a stack data structure, implementing basic operations like adding, editing, deleting, searching, displaying, and sorting students. Here's a summary of how it's structured:.

1. Student Class

- Attributes: id, name, marks (marks as a double for precision).
- Methods:
  - getId(), getName(), getMarks() to get student details.
  - getRanking() to return the rank based on marks using predefined ranges (Fail, Medium, Good, Very Good, Excellent).
  - toString() for a readable string representation of a student.

1. Student Class

ADT (Abstract Data Type) is an abstract data type that defines the behavior of a data type without specifying how to implement it.

For example, the Stack ADT defines operations such as push, pop, without knowing how it is performed.

2. ArrayStack Class

2. ArrayStack Class

- Attributes:
  - maxSize: the maximum capacity of the stack.
  - stackArray: an array to store students.
  - top: index to keep track of the top of the stack.

```
class ArrayStack {
    private int maxSize;  3 usages
    private Student[] stackArray;  4 usages
    private int top;  8 usages

    public ArrayStack(int size) {  2 usages
        this.maxSize = size;
        this.stackArray = new Student[maxSize];
        this.top = -1;
    }
```

2. ArrayStack Class

- Methods:
- push(Student student): Adds a student to the stack if space is available.
- pop(): Removes and returns the top student from the stack.
- peek(): Returns the top student without removing it.
- isEmpty(): Checks if the stack is empty.
-

3. Main Class

Operations:

- Add Student: Prompt user for student ID, name, and marks, and add to the stack.
- Edit Student: Allows editing a student's information by ID, updates marks and name.
- Delete Student: Removes the top student from the stack.
- Display All Students: Displays all students in the stack, including their rank based on marks.
- Search Student: Finds a student by ID and shows their details and rank.
- Sort Students: Sorts students by marks using Bubble Sort.

3. Main Class

1Flow:

- The user interacts with the system through a simple menu and can manage students as needed.

- The program uses a stack to store students, utilizing methods like push() and pop() to manage student records.

```
public static void main(String[] args) {

    Scanner scanner = new Scanner(System.in);

    // Nhập số lượng sinh viên cần quản lý
    System.out.println("Enter the number of students in the class: ");
    int numberOfStudents = scanner.nextInt();
    studentStack = new ArrayStack(numberOfStudents);

    while (true) {
        // Menu chính
        System.out.println("\n===== Student Management System =====");
        System.out.println("1. Add Student");
        System.out.println("2. Edit Student");
        System.out.println("3. Delete Student");
        System.out.println("4. Display All Students");
        System.out.println("5. Search Student by ID");
        System.out.println("6. Sort Students by Marks");
        System.out.println("7. Exit");
        System.out.print("Choose an option: ");
        int choice = scanner.nextInt();
        scanner.nextLine(); // Xóa bộ đệm
```

Implementation Details

1 Implementation Details

- Stack Operations: Students are managed using a stack data structure to ensure Last-In-First-Out (LIFO) order.
- Ranking: Students' ranks are assigned based on predefined mark ranges and are displayed when viewing or searching for students.
- Sorting: The Bubble Sort algorithm is used to sort students based on their marks

```
// Sắp xếp sinh viên theo điểm sử dụng Bubble Sort
    private static void sortStudentsBubbleSort() {  1 usage
        ArrayList<Student> tempStack = new ArrayList<>();
        // Lấy tất cả sinh viên từ stack và lưu vào tempStack
        while (!studentStack.isEmpty()) {
            tempStack.add(studentStack.pop());
        }
        // Bubble Sort
        for (int i = 0; i < tempStack.size() - 1; i++) {
            for (int j = 0; j < tempStack.size() - 1 - i; j++) {
                // So sánh điểm của 2 sinh viên
                if (tempStack.get(j).getMarks() < tempStack.get(j + 1).getMarks()) {
                    // Hoán đổi nếu sinh viên hiện tại có điểm thấp hơn sinh viên tiếp theo
                    Student temp = tempStack.get(j); // Khai báo temp để lưu trữ sinh viên tạm thời
                    tempStack.set(j, tempStack.get(j + 1)); // Đổi vị trí
                    tempStack.set(j + 1, temp); // Đổi vị trí
                }
            }
        }
        // Đưa lại tất cả sinh viên vào stack sau khi đã sắp xếp
        for (int i = tempStack.size() - 1; i >= 0; i--) {
            studentStack.push(tempStack.get(i));
        }

        System.out.println("Students sorted by marks using Bubble Sort.");
        displayAllStudents();
    }
```

LinkedListStack Class Implementation

Class Design:

- Node Class:
  - A node is a simple class that holds a Student object and a reference to the next node in the stack

```
class LinkedListStack {
    private Node top = null;  9 usages

    public LinkedListStack() {  1 usage
    }
```

Methods:

- push(Student student): Adds a student to the top of the stack by creating a new node, pointing it to the current top, and then making it the new top.
- pop(): Removes and returns the top student from the stack. If the stack is empty, it throws an exception.
- peek(): Returns the student at the top of the stack without removing it.
- isEmpty(): Checks if the stack is empty.

```
public void push(Student student) {  2 usages
    Node newNode = new Node(student);
    newNode.next = this.top;
    this.top = newNode;
    System.out.println("Added " + String.valueOf(student) + " to the stack.");
}

public Student pop() {  1 usage
    if (this.top == null) {
        System.out.println("Stack is empty. Cannot pop.");
        throw new EmptyStackException();
    } else {
        Student poppedValue = this.top.data;
        this.top = this.top.next;
        return poppedValue;
    }
}

public Student peek() {  1 usage
    if (this.top == null) {
        System.out.println("Stack is empty. Nothing to peek.");
        throw new EmptyStackException();
    } else {
        return this.top.data;
    }
}
```

Conclusion

This Student Management System provides a simple, effective way to manage student records using stacks, sorting algorithms, and rank-based evaluations based on marks.