Topic: Inheritance, Abstract Classes, and Interfaces

The Scenario: "Lehman Campus Payment System"

The college needs a new system to handle different types of payments (Credit Cards, Meal Plans, and Financial Aid). You are tasked with building the class hierarchy for this system.

Your Task

Create Java files (or a set of classes) that implements the following requirements:

1. The Interface: `Payable`

Create an interface named `Payable` that defines the "contract" for any payment method.

- **Method:** `void processPayment(double amount)`
- **Method:** `String getPaymentStatus()`

2. The Abstract Class: `PaymentMethod`

Create an abstract class named `PaymentMethod` that implements `Payable`.

- **Fields:** * `protected String accountHolder`
    - `protected double balance`
- **Constructor:** Initialize both fields.
- **Static Member:** Add a `static int totalTransactions` to keep track of how many payments have been processed across the *entire* system.
- **Abstract Method:** `abstract void validateAccount();`

3. The Concrete Subclass: `CreditCard`

Create a class `CreditCard` that extends `PaymentMethod`.

- **Additional Field:** `private double creditLimit`
- **Requirement:** Override `processPayment`. If the amount exceeds `balance + creditLimit`, print "Transaction Declined." Otherwise, subtract the amount and increment the `totalTransactions` static counter.
- **Requirement:** Use the `super` keyword to call the parent constructor.

4. The Concrete Subclass: `MealPlan`

Create a class `MealPlan` that extends `PaymentMethod`.

- **Requirement:** Override `validateAccount`. Meal plan balances cannot be negative.
- **Requirement:** Implement `processPayment` specifically for meal plan logic (e.g., only allowing payments if the balance is sufficient).

Challenge Logic (Main Method)

In your `main` method, perform the following:

1. Create an `ArrayList<Payable>` called `paymentQueue`.

2. Add one `CreditCard` object and one `MealPlan` object to the list.
3. **Polymorphism in Action:** Loop through the list and call `processPayment(50.0)` on every item.
4. Print the final `totalTransactions` using the class name (not an instance).

Checklist for Success

- [ ] Did you use the `@Override` annotation for clarity?
- [ ] Is your `totalTransactions` variable `static`?
- [ ] Did you remember that you **cannot** instantiate `PaymentMethod` directly?
- [ ] Does your `CreditCard` constructor use `super(...)` as its first line?

Submission: Upload your working code to Github and provide the link before the end of the period for in-class credit!