**1차 과제**

과제 요구조건 문제

5번. 테마를 바꾼 뒤에도 카드의 뒤집힌 상태가 유지되어야 한다.

* 카드뷰로 선언된 뷰의 @State 속성값이 테마를 바꾼 뒤에도 변하지 않아야 한다.
* 카드뷰는 불변 객체로 생성 당시 변수에 할당되지 않아 값을 추적할 수 없다.
* 테마를 누를 때마다 버튼이 새로운 뷰를 반환해 이전의 카드뷰 인스턴스가 모두 바뀐다.
* 불 배열을 선언해 카드뷰의 상태를 저장하려고 했으나 불가
* 단순하게 불배열을 선언해 카드 뷰를 생성할 때마다 카드 뷰의 상태 속성에 접근해 이를 불배열에 저장하고 반영하기 -> 불변 객체인 자신의 속성에 컴퓨티드 프로퍼티로 동적으로 값을 할당할 수 없음 -> @State 키워드를 붙여 참조형으로 CardView 배열 선언 후 재할당 시도-> No exact matches in reference to static method ‘buildExpression’
* ‘buildExpression’은 SwiftUI에서 result builder를 사용할 때 흔히 발생하는 에러이다.
* 하지만 현재 상황에서는 ‘buildExpression’ 메소드를 사용하지 않은 상태에서 이 에러가 발생했기 때문에 조금 복잡하다. 이럴 경우 타입 불일치나 뷰나 표현이 SwiftUI 뷰에서 제대로 사용되지 않아 발생하는 이슈와 관련이 있을 수 있다.
* CardView를 생성하는 ForEach가 구조체여서 일단 ForEach부터 정확하게 파악해야 한다.
* 컴퓨티드 프로퍼티가 뷰를 반환하는 게 정확히 어떤 의미인지 파악해야 한다.

**ForEach**

ForEach는 콜렉션에서 뷰를 반환하는 구조체이다. 반복을 통해 여러 개의 뷰를 생성할 때 유용하다. 스위프트의 선언적이고 데이터 주도적인 접근 방식에 적합한 UI 생성 방식이다.

ForEach의 특징

1. 데이터-드리븐: foreach는 데이터 콜렉션이나 클로저를 받아 콜렉션의 각 요소에 해당하는 뷰를 반환한다.
2. 식별가능한 데이터: 각각의 데이터는 고유한 값을 가져 구별 가능해야 한다. ForEach는 Identifiable 프로토콜을 채택해 각각의 요소가 유니크한 식별자를 제공해야 한다.
3. SwiftUI Views의 사용: 주로 SwiftUI의 body 연산에 사용된다. 스택, 리스트, 그리드 외에 다른 SwiftUI 레이아웃의 구성요소가 된다.

struct ContentView: View {

let numbers = [1, 2, 3, 4, 5]

var body: some View {

VStack {

ForEach(numbers, id: \.self) { number in

Text("Number \(number)")

}

}

}

}

위 코드에서 id: \.self는 각각의 요소를 유니크하게 구별하기 위해 사용된다.

추가적인 특징

1. 동적 리스트: ForEach는 SwiftUI에서 동적 리스트를 만들 때 사용한다. 데이터가 변하면 ForEach는 자동으로 변경을 반영하기 위해 UI를 변경한다.
2. 격자나 복잡한 레이아웃: 다른 뷰 컨테이너를 합성해 격자와 같은 보다 복잡한 레이아웃을 만들 때 쓰인다.
3. 유저 액션 핸들링: 동적이고 인터렉티브한 인터페이스를 가능하게 해 각각의 아이템에 대한 유저의 상호작용을 다룰 수 있다.

퍼포먼스 고려사항

1. ForEach는 뷰가 바뀌었을 때만 re-render하기 때문에 동적인 데이터 셋 관리에 효율적이다.