앱프로그래밍응용 1학기 중간고사 해설지

1번 문제. 앱 개발 방식

문제: 앱 개발 방식에 대한 설명으로 올바른 것은?

1. 모든 앱은 반드시 네이티브 코드로만 작성해야 한다.
2. 플러터(Flutter)는 하나의 코드베이스로 여러 플랫폼에서 앱을 개발할 수 있게 한다.
3. 앱 개발 시 디자인과 기능을 분리하지 않고 한꺼번에 작업해야 한다.
4. 앱 개발은 무조건 빠르게 결과를 내는 것을 목표로 한다.

정답: 2

**해설:**

* 플러터는 하나의 코드베이스를 활용하여 안드로이드와 iOS 앱을 동시에 개발할 수 있습니다. 이를 크로스플랫폼 개발이라 합니다.
* 이를 통해 개발 속도가 빨라지고 유지보수가 용이해집니다.
* 네이티브 코드도 여전히 중요하지만, 플러터는 다양한 장점을 제공하여 개발자들에게 인기 있는 선택입니다.

2번 문제. 플러터 특징

문제: 플러터(Flutter)의 주요 특징으로 옳은 것은?

1. 플러터는 오직 안드로이드 앱 개발만 지원한다.
2. 플러터는 네이티브 성능과 유사한 앱을 개발할 수 있다.
3. 플러터는 다트(Dart) 언어를 사용하지 않는다.
4. 플러터는 UI 디자인을 자동으로 생성한다.

정답: 2

**해설:**

* 플러터는 하나의 코드베이스로 안드로이드와 iOS 앱을 동시에 개발할 수 있습니다.
* 네이티브 성능과 유사한 앱을 개발할 수 있으며, 다트(Dart) 언어를 사용하여 빠르고 효율적인 개발을 지원합니다.
* 플러터는 다양한 위젯을 제공하여 개발자가 직접 UI를 디자인할 수 있도록 돕습니다.

3번 문제. 환경 구성 검사

문제: flutter doctor 명령어의 주요 역할로 옳은 것은?

1. 플러터 프로젝트를 자동으로 생성한다.
2. 플러터 앱의 UI를 자동으로 디자인한다.
3. 플러터 개발 환경의 설정 상태를 점검한다.
4. 플러터 앱을 배포하는 데 사용된다.

정답: 3

**해설:**

* flutter doctor는 플러터 개발 환경이 올바르게 설정되었는지 확인하는 명령어입니다.
* 설치된 SDK, Android Studio, iOS 설정 등 개발에 필요한 도구들의 상태를 점검하고, 누락된 항목을 알려줍니다.
* 이를 통해 개발자는 환경 구성을 빠르게 완료할 수 있습니다.

4번 문제. 변수 타입

문제: 다트(Dart)에서 변수 타입에 대한 설명으로 옳은 것은?

1. 모든 변수는 반드시 var 키워드를 사용해야 한다.
2. dynamic 타입은 런타임에 변수의 타입을 변경할 수 있다.
3. String 타입은 숫자 값을 저장할 수 있다.
4. int 타입은 소수점 값을 저장할 수 있다.

정답: 2

**해설:**

* dynamic 타입은 런타임에 변수의 타입을 변경할 수 있는 유연한 타입입니다.
* String 타입은 문자열만 저장할 수 있으며, 숫자 값은 저장할 수 없습니다.
* int 타입은 정수만 저장할 수 있으며, 소수점 값을 저장하려면 double 타입을 사용해야 합니다.

5번 문제. final, const

문제: final과 const의 차이점으로 옳은 것은?

1. final은 런타임에 값이 결정되며, const는 컴파일 타임에 값이 결정된다.
2. final과 const는 동일하게 동작하며, 차이가 없다.
3. final은 변수의 값을 변경할 수 있지만, const는 변경할 수 없다.
4. final은 상수 값을 나타내며, const는 변수 값을 나타낸다.

정답: 1

**해설:**

* final은 런타임에 값이 결정되며, 한 번 값이 설정되면 변경할 수 없습니다.
* const는 컴파일 타임에 값이 결정되며, 불변 객체를 생성합니다.
* 두 키워드는 모두 값을 변경할 수 없지만, 결정 시점에서 차이가 있습니다.

6번 문제. 타입검사, 형변환

문제: 타입 검사와 형 변환에 대한 설명으로 옳은 것은?

1. is 연산자는 변수의 타입을 변환하는 데 사용된다.
2. as 연산자는 변수의 타입을 검사하는 데 사용된다.
3. is 연산자는 변수의 타입을 검사하는 데 사용된다.
4. 다트에서는 타입 변환이 자동으로만 이루어진다.

정답: 3

**해설:**

* is 연산자는 변수의 타입을 검사하는 데 사용됩니다. 예를 들어, if (x is String)은 x가 String 타입인지 확인합니다.
* as 연산자는 명시적으로 타입을 변환할 때 사용됩니다. 예를 들어, (x as String)은 x를 String 타입으로 변환합니다.
* 다트에서는 자동 타입 변환뿐만 아니라 명시적 타입 변환도 가능합니다.

7번 문제. 익명함수, 람다식

문제: 익명 함수와 람다식에 대한 설명으로 옳은 것은?

1. 익명 함수는 이름이 없는 함수로, 주로 일회성으로 사용된다.
2. 람다식은 항상 여러 줄로 작성되어야 한다.
3. 익명 함수는 변수에 할당할 수 없으며, 즉시 실행만 가능하다.
4. 람다식은 다트에서 지원되지 않는다.

정답: 1

**해설:**

* 익명 함수는 이름이 없는 함수로, 주로 일회성 작업을 위해 사용됩니다.
* 람다식은 다트에서 지원되며, 간결한 문법으로 함수를 표현할 수 있습니다.
* 익명 함수는 변수에 할당하거나 다른 함수의 인수로 전달할 수 있습니다.
* 람다식은 한 줄로 작성할 수 있으며, 간단한 작업에 적합합니다.

8번 문제. 선택 매개변수

문제: 선택 매개변수에 대한 설명으로 옳은 것은?

1. 선택 매개변수는 항상 필수로 값을 전달해야 한다.
2. 선택 매개변수는 기본값을 설정할 수 있다.
3. 선택 매개변수는 함수의 첫 번째 매개변수로만 사용할 수 있다.
4. 선택 매개변수는 중괄호 {}를 사용할 수 없다.

정답: 2

**해설:**

* 선택 매개변수는 기본값을 설정하여 함수 호출 시 값을 전달하지 않아도 사용할 수 있습니다.
* 선택 매개변수는 중괄호 {}를 사용하여 정의하며, 이를 통해 가독성을 높이고 유연한 함수 호출이 가능합니다.
* 필수 매개변수와 함께 사용할 수도 있습니다.

9번 문제. 클래스 정의

문제: 클래스 정의에 대한 설명으로 옳은 것은?

1. 클래스는 객체를 생성할 수 없는 구조체이다.
2. 클래스는 필드와 메서드를 포함할 수 있다.
3. 다트에서는 클래스 정의 시 반드시 생성자를 작성해야 한다.
4. 클래스는 상속을 지원하지 않는다.

정답: 2

**해설:**

* 클래스는 객체 지향 프로그래밍의 핵심 요소로, 필드(변수)와 메서드(함수)를 포함할 수 있습니다.
* 생성자는 선택적으로 작성할 수 있으며, 기본 생성자가 자동으로 제공됩니다.
* 다트에서는 클래스 상속을 지원하며, 이를 통해 코드 재사용성을 높일 수 있습니다.

10번 문제. 상속 vs 인터페이스

문제: 상속과 인터페이스에 대한 설명으로 옳은 것은?

1. 상속의 장점은 코드 재사용이며, 단점은 사용하기 어렵다는 점이다.
2. 인터페이스의 장점은 코드 재사용이며, 단점은 사용하기 어렵다는 점이다.
3. 상속의 장점은 사용하기 쉬우며, 단점은 중복 코드가 발생한다는 점이다.
4. 인터페이스의 장점은 코드 재사용이며, 단점은 상속보다 구현하기 어렵다는 점이다.

정답: 1

**해설:**

* **상속**: 장점은 부모 클래스의 코드를 재사용할 수 있어 효율적인 개발이 가능하다는 점입니다. 하지만 복잡한 구조와 제한으로 인해 사용하기 어려울 수 있습니다.
* **인터페이스**: 장점은 사용하기 간단하고 유연하지만, 동일한 코드가 여러 클래스에 중복되어 작성될 가능성이 있습니다.

11번 문제. List, Map, Set 표기법

문제: List, Map, Set의 표기법에 대한 설명으로 옳은 것은?

1. List는 중괄호 { }를 사용하여 데이터를 정의한다.
2. Map은 대괄호 [ ]를 사용하여 키-값 쌍 데이터를 정의한다.
3. Set은 중괄호 { }를 사용하여 순서가 없는 고유 데이터를 정의한다.
4. Map은 순서가 있는 데이터 구조이다.

정답: 3

**해설:**

* **List**: 대괄호 [ ]를 사용하여 순서가 있는 데이터를 정의하며, 중복된 값을 허용합니다. 예: List<int> numbers = [1, 2, 3, 4];
* **Map**: 중괄호 { }를 사용하여 키-값 쌍 데이터를 정의하며, 각 키는 고유해야 합니다. 예: Map<String, int> scores = {'Alice': 90, 'Bob': 85};
* **Set**: 중괄호 { }를 사용하여 순서가 없는 데이터를 정의하며, 중복된 값은 자동으로 제거됩니다. 예: Set<int> uniqueNumbers = {1, 2, 3};

12번 문제. 함수형 프로그래밍 특징

문제: 함수형 프로그래밍의 특징으로 옳은 것은?

1. 일급 객체는 함수형 프로그래밍에서 지원되지 않는다.
2. forEach는 컬렉션의 각 요소를 반복 처리하는 데 사용된다.
3. where는 컬렉션의 모든 요소를 변환하는 데 사용된다.
4. map은 조건에 맞는 요소만 필터링하는 데 사용된다.

정답: 2

**해설:**

* **일급 객체**: 함수는 변수에 할당하거나 다른 함수의 인수로 전달할 수 있는 일급 객체로 취급됩니다.
* **forEach**: 컬렉션의 각 요소를 반복 처리하는 데 사용됩니다. 예: list.forEach((element) => print(element));
* **where**: 조건에 맞는 요소만 필터링하여 새로운 컬렉션을 반환합니다. 예: list.where((element) => element > 5);
* **map**: 컬렉션의 각 요소를 변환하여 새로운 컬렉션을 생성합니다. 예: list.map((element) => element \* 2);

13번 문제. 컬렉션 if 또는 컬렉션 for

문제: 컬렉션 if와 컬렉션 for의 사용법에 대한 설명으로 옳은 것은?

1. 컬렉션 if는 컬렉션의 모든 요소를 반복 처리하는 데 사용된다.
2. 컬렉션 for는 조건부로 요소를 추가할 수 있다.
3. 컬렉션 if는 조건에 따라 컬렉션에 요소를 추가할 수 있다.
4. 컬렉션 for는 컬렉션의 요소를 필터링하는 데 사용된다.

정답: 3

**해설:**

* **컬렉션 if**: 조건에 따라 컬렉션에 요소를 추가할 수 있습니다. 예: var list = [if (condition) value];
* **컬렉션 for**: 반복문을 사용하여 컬렉션의 요소를 추가하거나 변환할 수 있습니다. 예: var list = [for (var item in items) item \* 2];
* 두 기능 모두 다트에서 컬렉션을 동적으로 구성하는 데 유용합니다.

14번 문제. pubspec.yaml 관련 설명

문제: 플러터 프로젝트의 pubspec.yaml 파일에 대한 설명으로 옳은 것은?

1. pubspec.yaml 파일은 플러터 프로젝트에서 사용되지 않는다.
2. pubspec.yaml 파일은 프로젝트의 의존성과 환경 설정을 정의한다.
3. pubspec.yaml 파일은 플러터 앱의 UI를 자동으로 생성한다.
4. pubspec.yaml 파일은 프로젝트의 실행 파일을 저장한다.

정답: 2

**해설:**

* pubspec.yaml 파일은 플러터 프로젝트의 의존성(패키지), SDK 버전, 그리고 애플리케이션의 메타데이터를 정의하는 데 사용됩니다.
* 이 파일을 수정한 후에는 flutter pub get 명령어를 실행하여 변경 사항을 적용해야 합니다.
* 플러터 프로젝트의 핵심 구성 요소로, 프로젝트 관리에 필수적입니다.

15번 문제. 샘플 앱의 기본구조

문제: 플러터 프로젝트를 생성하면 기본적으로 만들어지는 main.dart 파일의 구성 요소로 옳은 것은?

1. main.dart 파일은 플러터 앱의 진입점으로, main() 함수와 runApp() 함수가 포함된다.
2. main.dart 파일은 앱의 모든 위젯을 정의해야 한다.
3. main.dart 파일은 플러터 앱의 데이터베이스 설정을 포함한다.
4. main.dart 파일은 UI 없이 백엔드 로직만 포함한다.

정답: 1

**해설:**

* main.dart 파일은 플러터 앱의 진입점으로, main() 함수가 포함되어 있습니다.
* main() 함수는 runApp()을 호출하여 앱의 루트 위젯을 실행합니다.
* 기본적으로 MyApp이라는 클래스가 정의되어 있으며, 이는 StatelessWidget 또는 StatefulWidget을 상속받아 앱의 구조를 구성합니다.

16번 문제. 화면 배치에 쓰는 위젯(1)

문제: 플러터에서 화면 배치에 사용되는 위젯에 대한 설명으로 옳은 것은?

1. Column 위젯은 수평으로 위젯을 배치하는 데 사용된다.
2. Row 위젯은 수직으로 위젯을 배치하는 데 사용된다.
3. Stack 위젯은 위젯을 겹쳐서 배치할 수 있다.
4. Container 위젯은 위젯의 배치를 제어할 수 없다.

정답: 3

**해설:**

* **Column**: 위젯을 수직으로 배치합니다. 예: Column(children: [Widget1, Widget2]);
* **Row**: 위젯을 수평으로 배치합니다. 예: Row(children: [Widget1, Widget2]);
* **Stack**: 위젯을 겹쳐서 배치할 수 있습니다. 예: Stack(children: [Widget1, Widget2]);
* **Container**: 크기, 패딩, 마진 등을 설정하여 위젯의 배치를 제어할 수 있습니다.

17번 문제. 화면 배치에 쓰는 위젯(2)

문제: 화면 배치에 사용되는 위젯과 그 역할에 대한 설명으로 옳은 것은?

1. Align 위젯은 위젯의 위치를 설정할 수 있다.
2. Stack 위젯은 위젯을 수직 또는 수평으로만 배치할 수 있다.
3. Column 위젯은 위젯을 겹쳐서 배치할 수 있다.
4. Row 위젯은 위젯의 위치를 제어할 수 없다.

정답: 1

**해설:**

* **Align**: 자식 위젯의 위치를 설정하여 원하는 위치로 배치할 수 있습니다. 예: Align(alignment: Alignment.centerRight, child: Text('Right'));
* **Stack**: 위젯을 겹쳐서 배치할 수 있습니다. 예: Stack(children: [Widget1, Widget2]);
* **Column**: 위젯을 수직으로 배치하며, 세로 방향의 레이아웃을 구성합니다. 예: Column(children: [Widget1, Widget2]);
* **Row**: 위젯을 수평으로 배치하며, 가로 방향의 레이아웃을 구성합니다. 예: Row(children: [Widget1, Widget2]);

18번 문제. 위치, 정렬, 크기를 위한 위젯(1)

문제: 위치, 정렬, 크기를 제어하는 위젯에 대한 설명으로 옳은 것은?

1. Align 위젯은 위젯의 크기를 조정하는 데 사용된다.
2. Center 위젯은 자식 위젯을 화면의 중앙에 배치한다.
3. SizedBox 위젯은 위젯의 정렬을 제어한다.
4. Padding 위젯은 위젯의 위치를 변경하는 데 사용된다.

정답: 2

**해설:**

* **Align**: 자식 위젯의 위치를 정렬할 수 있습니다. 예: Align(alignment: Alignment.topRight, child: Text('Hello'));
* **Center**: 자식 위젯을 화면의 중앙에 배치합니다. 예: Center(child: Text('Hello'));
* **SizedBox**: 위젯의 크기를 조정하거나 간격을 추가할 수 있습니다. 예: SizedBox(width: 100, height: 50);
* **Padding**: 위젯 주위에 여백을 추가하여 위치를 조정합니다. 예: Padding(padding: EdgeInsets.all(8.0), child: Text('Hello'));

19번 문제. 위치, 정렬, 크기를 위한 위젯(2)

문제: 다음 위젯 중 위치와 정렬에 사용되는 위젯은?

1. SizedBox
2. Align
3. Padding
4. 모두 해당된다.

정답: 2, 4

관점에 따라 SizedBox및 Padding도 위치와 정렬에 영향을 주므로 모호하지만 **4번도 정답 처리 함**

**해설:**

* **SizedBox**: 고정된 크기를 설정하거나 두 위젯 사이에 공간을 추가하는 데 사용됩니다. 예: SizedBox(width: 50, height: 20);
* **Align**: 자식 위젯을 지정된 위치에 정렬합니다. 예: Align(alignment: Alignment.bottomRight, child: Text('Align Example'));
* **Padding**: 위젯 주위에 여백을 추가하여 공간을 조정합니다. 예: Padding(padding: EdgeInsets.all(16.0), child: Text('Padding Example'));
* 이 중, 위치와 정렬에 사용되는 위젯은 Align 입니다.

20번 문제. 버튼 등 위젯(1)

문제: 버튼 계열 위젯에 대한 설명으로 옳은 것은?

1. ElevatedButton은 플랫한 디자인의 버튼이다.
2. TextButton은 텍스트만 표시되는 버튼이다.
3. IconButton은 텍스트와 아이콘을 함께 표시한다.
4. FloatingActionButton은 정적인 버튼으로, 화면 이동에 사용된다.

정답: 2

**해설:**

* **ElevatedButton**: 입체적인 디자인의 버튼으로, 그림자 효과를 제공합니다. 예: ElevatedButton(onPressed: () {}, child: Text('Click Me'));
* **TextButton**: 텍스트만 표시되는 버튼으로, 간단한 작업에 적합합니다. 예: TextButton(onPressed: () {}, child: Text('Click Me'));
* **IconButton**: 아이콘만 표시되는 버튼으로, 주로 간단한 액션에 사용됩니다. 예: `IconButton(onPressed: () {}, icon: Icon(Icons

21번 문제. 버튼 등 위젯(2)

문제: 버튼 계열 위젯과 그 역할에 대한 설명으로 옳은 것은?

1. ElevatedButton은 그림자 없이 평면적인 디자인을 제공한다.
2. FloatingActionButton은 화면 하단에 고정된 버튼으로 사용할 수 있다.
3. IconButton은 아이콘만 표시되는 버튼을 지원한다.
4. TextButton은 버튼 클릭 시 물결 효과를 제공하지 않는다.

정답: 3

**해설:**

* **ElevatedButton**: 그림자 효과를 포함한 입체적인 버튼입니다. 예: ElevatedButton(onPressed: () {}, child: Text('Click Me'));
* **FloatingActionButton**: 화면 위에 떠 있는 주요 작업 버튼으로 사용됩니다. 예: FloatingActionButton(onPressed: () {}, child: Icon(Icons.add));
* **IconButton**: 아이콘만 표시되는 버튼으로, 주로 간단한 동작을 처리하는 데 사용됩니다. 예: IconButton(onPressed: () {}, icon: Icon(Icons.settings));
* **TextButton**: 간단한 텍스트 기반 버튼이며, 클릭 시 물결 효과를 제공합니다. 예: TextButton(onPressed: () {}, child: Text('Click Me'));

22번 문제. 입력 관련

문제: 플러터에서 다음의 설명으로 옳은 것은?

1. TextField는 사용자 입력을 받을 수 없는 위젯이다.
2. CheckBox는 단일 선택만 지원한다.
3. Switch는 켜짐/꺼짐 상태를 나타내는 토글 버튼이다.
4. DropDownButton은 단일 선택을 지원하지 않는다.

정답: 3

**해설:**

* **TextField**: 사용자 입력을 받을 수 있는 기본적인 입력 필드입니다. 예: TextField(controller: myController);
* **CheckBox**: 다중 선택을 지원하며, 사용자가 선택 여부를 표시할 수 있습니다. 예: Checkbox(value: isChecked, onChanged: (bool? value) {});
* **Switch**: 켜짐/꺼짐 상태를 나타내는 토글 버튼으로, 상태 변경을 처리합니다. 예: Switch(value: isSwitched, onChanged: (bool value) {});
* **Radio**: 단일 선택을 지원하며, 그룹 내에서 하나의 옵션만 선택할 수 있습니다. 예: Radio(value: option, groupValue: selectedOption, onChanged: (value) {});
* **DropDownButton**: 드롭다운 메뉴를 제공하며, 단일 선택을 지원합니다. 예: DropdownButton(items: dropdownItems, onChanged: (value) {});

23번 문제. 다이얼로그 등 위젯

문제: AlertDialog, DatePicker, TimePicker에 대한 설명으로 옳은 것은?

1. AlertDialog는 사용자 입력을 받을 수 없는 정적인 팝업 창이다.
2. DatePicker는 시간을 선택하는 데 사용된다.
3. TimePicker는 날짜를 선택하는 데 사용된다.
4. AlertDialog는 사용자와 상호작용하는 팝업 창을 제공한다.

정답: 4

**해설:**

* **AlertDialog**: 사용자와 상호작용하는 팝업 창으로, 메시지와 버튼을 포함할 수 있습니다. 예: AlertDialog(title: Text('Title'), content: Text('Content'));
* **DatePicker**: 사용자가 날짜를 선택할 수 있는 위젯입니다. 예: showDatePicker(context: context, initialDate: DateTime.now(), firstDate: DateTime(2000), lastDate: DateTime(2100));
* **TimePicker**: 사용자가 시간을 선택할 수 있는 위젯입니다. 예: showTimePicker(context: context, initialTime: TimeOfDay.now());

24번 문제. 이벤트

문제: GestureDetector와 InkWell에 대한 설명으로 옳은 것은?

1. GestureDetector는 터치 이벤트를 처리할 수 없다.
2. InkWell은 터치 이벤트를 처리하며, 시각적 효과를 제공한다.
3. GestureDetector는 터치 이벤트를 처리하며, 시각적 효과를 제공한다.
4. InkWell은 터치 이벤트를 처리할 수 없다.

정답: 2

**해설:**

* **GestureDetector**: 터치 이벤트를 처리할 수 있는 위젯으로, 다양한 제스처를 감지할 수 있습니다. 예: GestureDetector(onTap: () { print('Tapped!'); });
* **InkWell**: 터치 이벤트를 처리하며, 터치 시 잉크 효과를 제공하는 위젯입니다. 예: InkWell(onTap: () { print('Tapped!'); }, child: Container(width: 100, height: 50));
* 두 위젯 모두 터치 이벤트를 처리할 수 있지만, **InkWell**은 시각적 효과를 추가로 제공합니다.

25번 문제. 애니메이션

문제: 다음의 설명으로 옳은 것은?

1. Hero는 화면 전환 시 위젯 간의 자연스러운 애니메이션을 제공한다.
2. AnimatedContainer는 애니메이션 효과 없이 정적인 컨테이너를 제공한다.
3. SliverAppBar는 스크롤 가능한 앱바를 제공하지 않는다.
4. SliverList는 고정된 크기의 리스트만 지원한다.

정답: 1

**해설:**

* **Hero**: 화면 전환 시 위젯 간의 자연스러운 애니메이션을 제공하며, tag 속성을 사용하여 연결합니다. 예: Hero(tag: 'heroTag', child: Image.asset('image.png'));
* **AnimatedContainer**: 크기, 색상, 패딩 등의 속성을 애니메이션 효과로 변경할 수 있는 컨테이너입니다. 예: AnimatedContainer(duration: Duration(seconds: 1), color: Colors.blue);
* **SliverAppBar**: 스크롤 가능한 앱바를 제공하며, 스크롤에 따라 크기가 조정될 수 있습니다. 예: SliverAppBar(expandedHeight: 200.0, flexibleSpace: FlexibleSpaceBar(title: Text('Title')));
* **SliverFillRemaining**: 남은 화면 공간을 채우는 위젯으로, 스크롤 가능한 레이아웃에서 사용됩니다.
* **SliverList**: 스크롤 가능한 리스트를 생성하며, 동적으로 아이템을 렌더링할 수 있습니다. 예: SliverList(delegate: SliverChildBuilderDelegate((context, index) => ListTile(title: Text('Item $index'))));

26번 문제. 쿠퍼티노 디자인

문제: 쿠퍼티노 디자인에 대한 설명으로 옳은 것은?

1. 쿠퍼티노 디자인은 안드로이드 스타일의 UI를 구현한다.
2. CupertinoButton은 iOS 스타일의 버튼을 제공한다.
3. CupertinoAlertDialog는 다이얼로그를 지원하지 않는다.
4. 쿠퍼티노 디자인은 플러터에서 지원되지 않는다.

정답: 2

**해설:**

* **CupertinoButton**: iOS 스타일의 버튼을 제공하며, 간단한 디자인과 터치 효과를 지원합니다. 예: CupertinoButton(child: Text('Click Me'), onPressed: () {});
* **CupertinoAlertDialog**: iOS 스타일의 다이얼로그를 제공하며, 제목과 메시지, 버튼을 포함할 수 있습니다. 예: CupertinoAlertDialog(title: Text('Title'), content: Text('Content'));
* 쿠퍼티노 디자인은 플러터에서 iOS 스타일의 UI를 구현하는 데 사용되며, CupertinoNavigationBar, CupertinoSlider 등 다양한 위젯을 포함합니다.

27번 문제. 내비에기션의 동작성

문제: 내비게이션의 동작 방식에 대한 설명으로 옳은 것은?

1. 플러터의 내비게이션은 화면 간 이동을 지원하지 않는다.
2. Navigator.push는 새로운 화면을 스택에 추가한다.
3. Navigator.pop은 스택의 모든 화면을 제거한다.
4. 내비게이션은 플러터에서 비동기적으로 동작할 수 없다.

정답: 2

**해설:**

* **Navigator.push**: 새로운 화면을 스택에 추가하여 화면 전환을 처리합니다. 예: Navigator.push(context, MaterialPageRoute(builder: (context) => NewScreen()));
* **Navigator.pop**: 현재 화면을 스택에서 제거하여 이전 화면으로 돌아갑니다.
* 플러터의 내비게이션은 비동기적으로 동작할 수 있으며, async와 await를 활용하여 결과를 처리할 수 있습니다.

28번 문제. push / pop 사용법

문제: Navigator의 push와 pop 메서드에 대한 설명으로 옳은 것은?

1. Navigator.push는 현재 화면을 스택에서 제거한다.
2. Navigator.pop은 새로운 화면을 스택에 추가한다.
3. Navigator.push는 새로운 화면을 스택에 추가한다.
4. Navigator.pop은 스택의 모든 화면을 제거한다.

정답: 3

**해설:**

* **Navigator.push**: 새로운 화면을 스택에 추가하여 화면 전환을 처리합니다. 예: Navigator.push(context, MaterialPageRoute(builder: (context) => NewScreen()));
* **Navigator.pop**: 현재 화면을 스택에서 제거하여 이전 화면으로 돌아갑니다. 예: Navigator.pop(context);
* 플러터의 내비게이션은 스택 구조를 기반으로 동작하며, 화면 간 이동을 효율적으로 처리할 수 있습니다.

29번 문제. routes를 이용한 방법

문제: routes를 이용한 내비게이션에 대한 설명으로 옳은 것은?

1. routes는 화면 간 이동을 지원하지 않는다.
2. MaterialApp의 routes 속성을 사용하여 화면 경로를 정의할 수 있다.
3. routes는 동적으로 화면을 추가할 수 없다.
4. routes는 내비게이션과 관련이 없다.

정답: 2

**해설:**

* **routes**: MaterialApp의 routes 속성을 사용하여 화면 경로를 정의할 수 있습니다. 예:

MaterialApp( routes: { '/': (context) => HomeScreen(), '/details': (context) => DetailsScreen(), },);

30번 문제. 널 안전한 코드

문제: 다트(Dart)에서 널 안전한 코드에 대한 설명으로 옳은 것은?

1. 널 안전성은 변수에 널(Null)을 허용하지 않는 것을 말한다.
2. 널 안전성은 모든 변수에 널을 허용해야 한다.
3. 널 안전성은 컴파일 타임에 널 관련 오류를 방지한다. 널을 허용해야 경우 타입뒤에 '?'를 붙인다.
4. 널 안전성은 런타임에만 오류를 방지한다.

정답: 3

**해설:**

* **널 안전성**: 다트에서 널 안전성은 컴파일 타임에 널 관련 오류를 방지하여 더 안전한 코드를 작성할 수 있도록 합니다.
* 널을 허용하려면 변수 선언 시 ?를 사용해야 합니다. 예: String? nullableString;
* 널이 아닌 값을 보장하려면 !를 사용하여 강제 언래핑할 수 있습니다. 예: nullableString!.