Flutter

분류	Front-end

▼ Flutter 의 시작

1. Flutter 및 Android 환경 설정 시 확인해야 할 기본 사항

1. **Flutter SDK 및 Android Studio 설치 상태 확인**
   - Flutter SDK, Android Studio, Android Emulator가 최신 버전인지 확인합니다.
   - Flutter 설치 확인:

     ```
     bash
     코드 복사
     flutter doctor
     ```

     - 출력 결과에서 모든 항목이 `[✓]` 인지 확인.
2. **Android SDK 및 AVD 설정 확인**
   - Android Studio → **SDK Manager**:
     - Android SDK Platform-Tools 및 Emulator 설치.
     - Android 버전(API Level) 및 시스템 이미지 설치.
   - AVD Manager에서 Pixel 시리즈 AVD 생성 및 설정 확인.
3. **GPU 설정 및 Windows 디스플레이 확인**
   - **NVIDIA 설정**:
     - GPU 가속 모드에서 문제가 발생할 수 있으므로, GPU 설정을 필요에 따라 변경(예: Swiftshader, Software 모드).
   - **Windows 디스플레이 설정**:
     - "텍스트, 앱 및 기타 항목의 크기"를 100%로 설정.
     - 다중 모니터 사용 시 디스플레이 연결 확인.
4. **Visual Studio C++ Redistributable 설치**

- Android Emulator 실행에는 **Microsoft Visual C++ Redistributable**가 필요.
- 다운로드 링크에서 설치.

2. 버그 관련 오류 고치는 법 정리

1. DLL 파일 관련 문제

문제: `libandroid-emu-metrics.dll`, `libshadertranslator.dll`, `qt6widgetsAndroidEmu.dll` 등이 없다는 오류 발생.

원인: Android SDK 경로 설정 문제 또는 파일 누락.

해결법:

1. Android SDK 경로 확인:

```
cmd
코드 복사
set ANDROID_SDK_ROOT=C:\Users\SSAFY\AppData\Local\And
roid\Sdk
```

2. 누락된 파일 확인:
   - SDK 내 `emulator/lib64` 디렉토리 확인.
   - 누락된 파일이 있다면 Android Studio 또는 SDK Tools 재설치.

3. C++ 설치해보기
   - 년도 별로 DLL 파일이 다를 수도 있으므로 vs c++ 버전을 연도 별로 다 설치해보기

---

2. AVD 실행 시 "Snapshot Loading Failed"

문제: `Failed to load snapshot 'default_boot'`, `performing a cold boot` 등의 경고.

원인: AVD 스냅샷 파일 손상 또는 스냅샷 설정 충돌.

해결법:

1. **Cold Boot 실행**:

```
cmd
코드 복사
emulator -avd <AVD_NAME> -no-snapshot
```

2. AVD 초기화:

```
cmd
코드 복사
emulator -avd <AVD_NAME> -wipe-data
```

---

3. 에뮬레이터 창이 표시되지 않음

문제: 작업 표시줄에는 실행 중인 에뮬레이터가 보이지만 화면에 창이 표시되지 않음.

원인: 에뮬레이터 창 위치가 화면 밖에 있거나 DPI 설정 문제.

해결법:

1. **창 위치 초기화**:
   - 작업 표시줄 아이콘 → **Shift + 오른쪽 클릭** → **이동** → 화살표 키로 창 이동.
2. AVD 해상도와 DPI 설정 조정:
   - AVD Manager에서 **Resolution**과 DPI를 낮은 값(예: DPI 240)으로 변경.
3. 에뮬레이터 기종 변경 해보기
   - 에뮬레이터의 노트북 같은 경우에는 pro 로 하면 화면 밖으로 나가는 경우가 있으므로 기본 api 8 등 이런 미디움 버전을 이용하면 화면에 바로 나타날 수도 있음.

Flutter 를 이용한 계산기 만들기

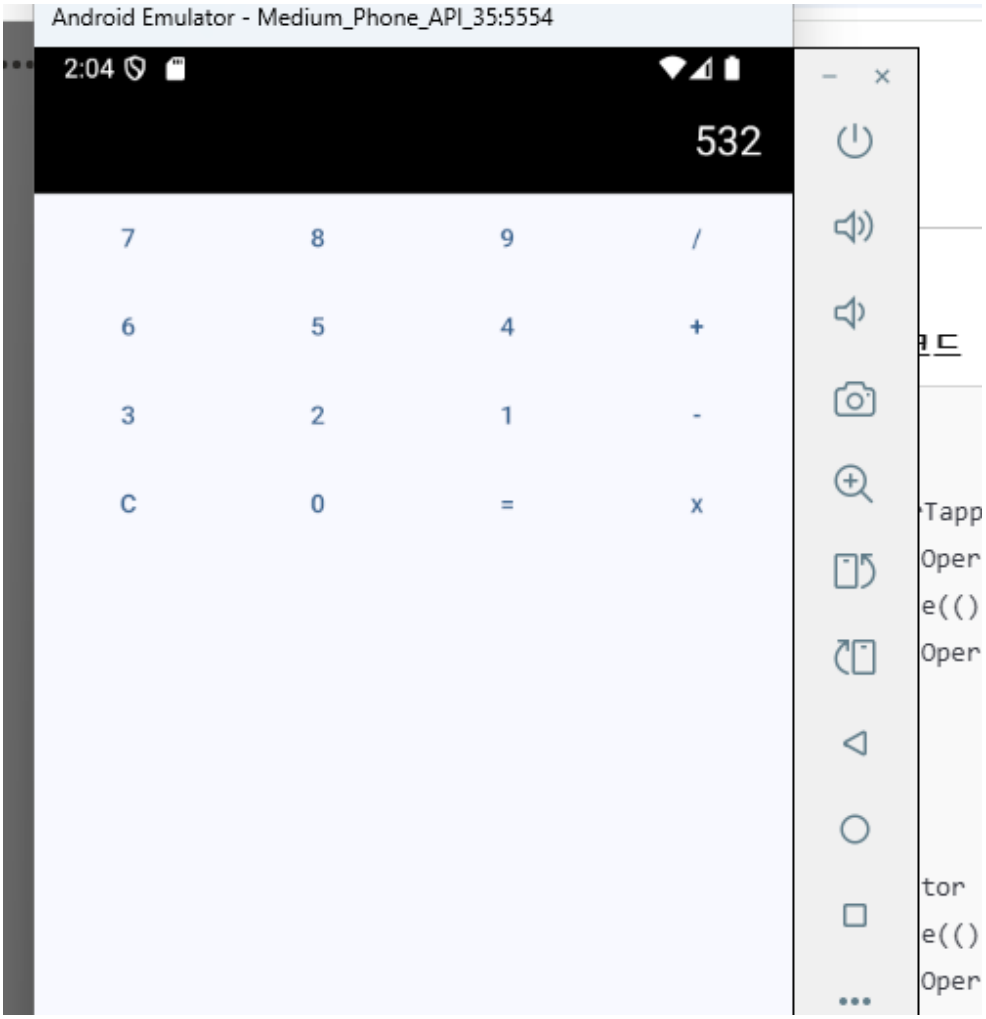
주요코드

```
// 숫자를 연속으로 입력 가능하게 하는 코드 및 operaotor 판단
```

```
void numberTapped(int tappedNumber) {
  if (firstOperand == null) {
    setState(() {
      firstOperand = tappedNumber;
    });
    return;
  }

  if (operator == null) {
    setState(() {
      firstOperand = int.parse("$firstOperand$tappedNumber"
    });
    return;
  }

  if (secondOperand == null) {
    setState(() {
      secondOperand = tappedNumber;
    });
    return;
  }

  setState(() {
    secondOperand = int.parse("${secondOperand!}$tappedNumb
  });
}

// 각각의 연산 기능 구현
void operatorTapped(String tappedOperator) {
  if (tappedOperator == "=") {
    if (firstOperand != null && secondOperand != null && op
      setState(() {
        switch (operator) {
          case "+":
            result = firstOperand! + secondOperand!;
            break;
          case "-":
            result = firstOperand! - secondOperand!;
```

```
            break;
          case "x":
            result = firstOperand! * secondOperand!;
            break;
          case "/":
            result = secondOperand != 0 ? firstOperand! ~/
            break;
        }
        firstOperand = result is int ? result : null;
        secondOperand = null;
        operator = null;
      });
    }
  } else {
    setState(() {
      operator = tappedOperator;
    });
  }
}
```

간단한 UI



주요 문제와 해결 과정

(1) 숫자 입력 오류

- 문제: `25x35`처럼 두 번째 피연산자 입력 시 null 값을 처리하지 않아 `FormatException` 발생.
- 해결:
  - `secondOperand`가 null일 경우 초기화하고, 값이 있으면 문자열 결합 처리.
  - `null safety`를 위해 `secondOperand!` 사용.

(2) 연산 결과 표시

- 문제: 계산 후 결과를 화면에 제대로 출력하지 못함.
- 해결:
  - `result` 변수를 추가하고, `showText()` 메서드에서 우선적으로 출력.

(3) UI/UX 문제

- 문제: 버튼 크기와 텍스트가 비례하지 않아서 가독성 저하.
- 해결:
    - `TextStyle`로 버튼 텍스트 크기 조정, `Expanded`로 버튼 비율 조정.