1) 소스코드, 개발환경
   A. https://github.com/SpicyYeol/Vital_sign_project (권한 필요)
   B. Branch: innopia_demo
   C. Vital_sign_project/demo/MyApplication/app
   D. IDE: Android Studio
2) 디자인 패턴, 라이브러리
   A. MVVM (Model View ViewModel)
      1. 비즈니스 로직은 "XXXViewModel"에 작성
      2. 화면 작성 시"XXXFragment" <-> "XXXViewModel"로 나누어서 관리
   B. Observer 패턴
      1. Rxjava, LiveData 사용
   C. DataBase
      1. Room 라이브러리 사용
      2. Repository 패턴 – "XXXRepository"를 통해 DB 접근
      3. 테이블 구조 변경 시 migrate 작성 필요
      4. 시간 정보는 UTC로 저장
   D. 그래프 라이브러리
      1. https://github.com/PhilJay/MPAndroidChart
      2. License: Apache License, Version 2.0
3) 카메라 / 이미지
   A. BaseCameraFragment.java
      1. 카메라 기기, 비율 자동 선택
      2. YUV -> RGB 계산
         - ImageUtils.convertYUV420ToARGB8888(...)
      3. 카메라 사용이 필요한 화면에서 상속받아 사용
         - processImage(...) Override 필요
   B. Recognizer.java
      1. 얼굴 인식 모듈
      2. recognizeFace(...) 메소드 호출
   C. MultiBoxTracker.java
      1. Object Tracker
      2. onFrame(...) 메소드에 Luminance 이미지 전달
      3. 좌표 리턴 X, 직접 draw
4) Splash (앱 시작 화면)
   A. SplashActivity.java
      1. 필요한 권한 상승 요청 (카메라, 음성, 저장소)
      2. 일정 시간 후 MainAcitivty 호출
5) Main
   A. MainActivity.java
      1. Fragment container가 포함된 Activity
      2. Main, Analysis, Result, History fragment inflate
   B. MainFragment.java
      1. Splash 이후에 출력되는 main 화면
      2. 최근 검사 데이터 출력

3. Analysis, History화면 연결 (버튼 클릭)
4. 측정 완료 시 Result 화면 연결

C. MainViewModel.java
1. Repository를 통해 DB 조회
2. 결과를 MainFragment에 notify

6) Analysis

A. Old Analysis Method
1. 기존 방식 (Non dnn)
2. 관련 클래스 이름은 'old'를 붙임
3. Vital 모듈에 calculateVital(…)를 호출해 계산
4. 관련 parameter는 BpmAnalysisViewModel에 정의
   - Total Frame size = 30 * 30

B. New Analysis Method
1. 새로 추가한 방식 (APNET)
2. 관련 클래스 이름은 'new'를 붙임
3. 32개의 frame을 torch module의 input으로 전달 (현재는 dummy data 사용)
4. 관련 parameter는 BpmAnalysisViewModel에 정의
   - Frame window size = 32
   - Total Frame size = 32 * 30
   - Face width * Face height = 128 * 128

C. BpmAnalysisFragment.java
1. 카메라 사용을 위해 BaseCameraFragment 상속
2. New/Old method 전환 시 관련 Layout의 visibility를 Visible <-> Gone 설정
3. FaceImage 추출
4. FaceImageModel(Image + Time) ViewModel에 전달 (addFaceImageModel(…))
5. Line Chart 초기화 (디자인 변경 시 이 부분 수정)

D. BpmAnalysisViewModel.java
1. New/Old method 전환 시 관련 변수 초기화 및 Fragment에 notify
2. FaceImageModel을 전달 받고 현재 측정 모드에 따라 처리
3. 진행도를 Fragment에 전달(Observer)
4. Old mode
   - Frame 1개를 이용해 결과 계산(Vital.calculateVital) 및 Fragment에 전달
   - OldVitalAnalysisRepository를 통해 DB 저장
5. New mode
   - Frame 32개를 모은 후 결과 계산(Torch module) 및 Fragment에 전달
   - 32개를 다 모으기 전에 얼굴 인식에 실패하는 경우 리셋
   - NewVitalAnalysisRepository를 통해 DB 저장
6. 측정 완료 후 MainActivity에 결과 notify
7. Torch module변경 필요 시
   - app/src/main/assets에 저장
   - TORCH_MODULE_NAME에 정의된 모듈 이름 변경

7) Result

A. ResultFragment.java
1. 측정 결과를 위한 화면

2. ResultViewModel 로부터 결과 수신(LiveData observe)

B. ResultViewModel.java

1. Old, New 최근 결과 확인 (by Repository.getLatestXXX())
2. 더 최근에 측정된 결과에 따라 Old or New layout 변경(BVP 그래프 유무)

8) History

A. HistoryFragment.java

1. 기간 별 데이터 통계를 그래프로 출력
2. New/Old method 전환 시 관련 그래프의 visibility를 Visible <-> Gone 설정
3. 날짜 선택 시 ViewModel에 선택 시간 전달

B. HistoryViewModel.java

1. 선택 시간을 전달받고 Repository를 통해 DB조회
2. Old
   - HistoryDataType으로 선택된 데이터를 구분/조회
   - DB조회 결과(OldVitalAnalysisEntity)를 HistoryDataModel로 변경/전달
3. New
   - DB조회 결과를 List형태로 전달