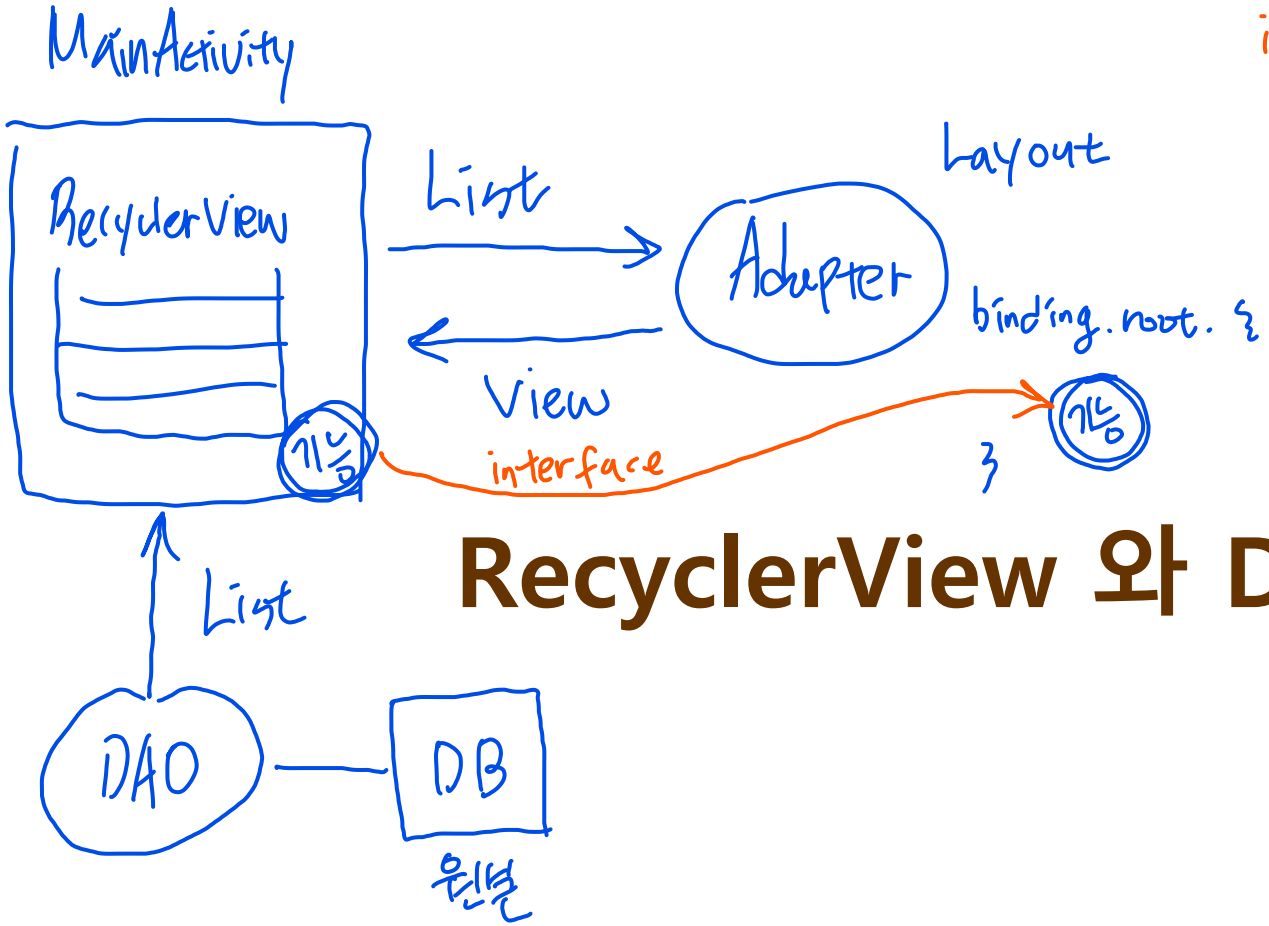


interface : 함수 선언만 존재!

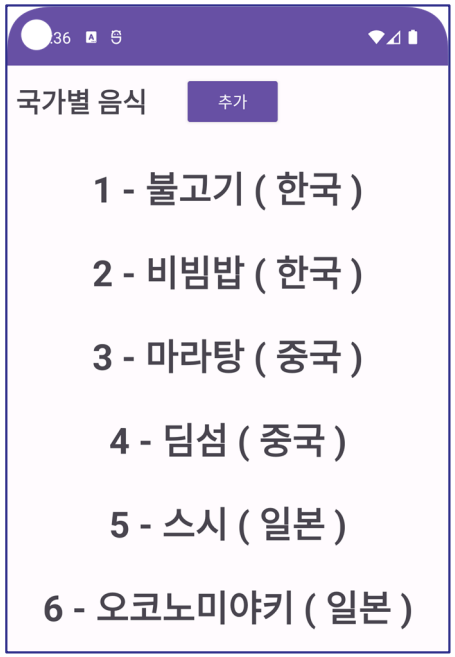
RecyclerView 와 Database

- http://developer.android.com

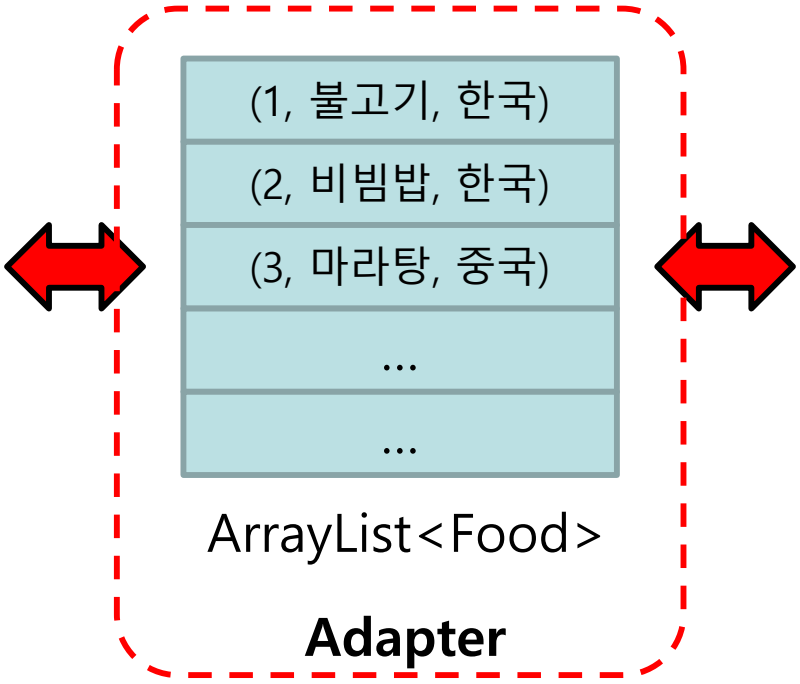
RecyclerView와 DB의 연결 01

DB의 내용을 RecyclerView에 출력 시

- RecyclerView는 Adapter를 통해 ArrayList<DTO> 형태의 컬렉션을 전달 받음
- 새로운 데이터 입력, 수정, 그리고 삭제 시 고려사항
  - 동일한 데이터가 List와 DB에 존재 (변경 시 둘 다 수정 필요) → DB를 갱신할 경우 List의 갱신도 필수



RecyclerView



_id	food	country
1	불고기	한국
2	비빔밥	한국
3	마라탕	중국
...	...	...

DB Table

원본

RecyclerView와 DB의 연결 02

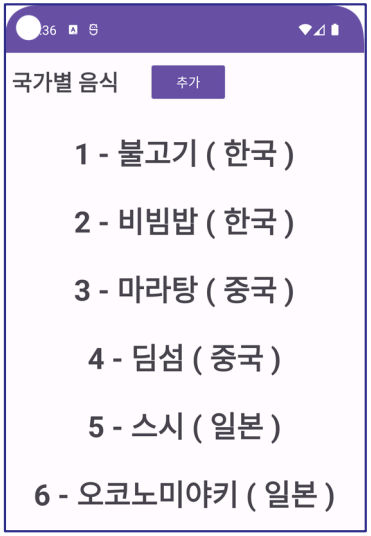
```
23  override fun onCreate(savedInstanceState: Bundle?) {
24      ...
        foods = getAllFoods()              // DB 에서 모든 food를 가져옴
        adapter = FoodAdapter(foods)       // adapter 에 데이터 설정
34      binding.rvFoods.adapter = adapter  // RecylcerView 에 adapter 설정
35  }
36
37  fun getAllFoods() : ArrayList<FoodDto> {
38      val helper = FoodDBHelper(this)
39      val db = helper.readableDatabase
40  //      val cursor = db.rawQuery("SELECT * FROM ${FoodDBHelper.TABLE_NAME}", null)
41      val cursor = db.query(FoodDBHelper.TABLE_NAME, null, null, null, null, null, null)
42
43      val foods = arrayListOf<FoodDto>()
44      with (cursor) { this: Cursor!
45          while (moveToNext()) {
46              val id = getInt( getColumnIndex(BaseColumns._ID) )
47              val food = getString ( getColumnIndex(FoodDBHelper.COL_FOOD) )
48              val country = getString ( getColumnIndex(FoodDBHelper.COL_COUNTRY) )
49              val dto = FoodDto(id, food, country)
50              foods.add(dto)
51          }
52      }
53      return foods
54  }
```

- 멤버변수
- DB의 기능별 함수
- DB의 테이블에서 레코드를 읽어와 Food DTO에 기록 후 foods 리스트에 추가

Update 의 수행 1

RecyclerView의 항목 업데이트 시 고려사항

- 선택한 항목 확인 → Position 으로 DTO 확인
- 선택한 정보를 수정화면에 표시 → DTO 를 전달
- 수정한 정보를 DB에 반영 후 액티비티 종료
- 결과에 따라 RecyclerView 의 화면 갱신



선택한 항목 확인 및 전달

- Adapter 의 외부 리스너 추가 후 사용

• MainActivity

```
27  override fun onCreate(savedInstanceState: Bundle?) {
28      ...
39      adapter.setOnItemClickListener(object : FoodAdapter.OnItemClickListener {
40          override fun onItemClick(view: View, position: Int) {
41              val intent = Intent(this@MainActivity, UpdateActivity::class.java)
42              intent.putExtra("dto", foods.get(position) )
43              startActivityForResult(intent, REQ_UPDATE)
44          }
45      })
46  }
```

• RecyclerView 의 클릭 위치에 해당하는 dto 를 전달

• 업데이트 수행 또는 취소를 확인하기 위해 결과 요청

Update 의 수행 2

- UpdateActivity

```
override fun onCreate(savedInstanceState: Bundle?) {
    ...
    /*RecyclerView 에서 선택하여 전달한 dto 를 확인*/
    val dto = intent.getSerializableExtra("dto") as FoodDto

    updateBinding.etUpdateId.setText(dto.id.toString())
    updateBinding.etUpdateFood.setText(dto.food)
    updateBinding.etUpdateCountry.setText(dto.country)

    updateBinding.btnUpdateFood.setOnClickListener{ it: View!
        dto.food = updateBinding.etUpdateFood.text.toString()
        dto.country = updateBinding.etUpdateCountry.text.toString()
        if (updateFood(dto) > 0) {
            setResult(RESULT_OK)
        } else {
            setResult(RESULT_CANCELED)
        }
        finish()
    }

    updateBinding.btnUpdateCancel.setOnClickListener{ it:
        setResult(RESULT_CANCELED)
        finish()
    }
}
```

• RecyclerView 의 클릭 위치에 해당하는 dto

• 성공여부만 전달하면 되므로 RESULT_OK 지정

```
fun updateFood(dto: FoodDto): Int {
    val helper = FoodDBHelper(this)
    val db = helper.writableDatabase
    val updateValue = ContentValues()
    updateValue.put(FoodDBHelper.COL_FOOD, dto.food)
    updateValue.put(FoodDBHelper.COL_COUNTRY, dto.country)
    val whereCaluse = "${BaseColumns._ID}=?"
    val whereArgs = arrayOf(dto.id.toString())
    val result =  db.update(FoodDBHelper.TABLE_NAME,
        updateValue, whereCaluse, whereArgs)
    helper.close()
    return result
}
```

Update 의 수행 3

- foods 리스트는 레퍼런스 변수이므로 Adapter 에 연결한 리스트 사용 필요

- MainActivity

```
52  override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
53      when (requestCode) {
54          REQ_UPDATE -> {
55              if (resultCode == RESULT_OK) {
56                  foods.clear()
57                  foods.addAll(getAllFoods())
58                  adapter.notifyDataSetChanged()
59              } else {
60                  Toast.makeText(this@MainActivity, "취소됨", Toast.LENGTH_SHORT).show()
61              }
62          }
63      }
64  }
```

- Adpater 에 연결해놓은 foods 리스트를 비운 후 업데이트 DB 내용을 읽어와 추가
- RecylcerView 갱신

추가와 삭제의 수행

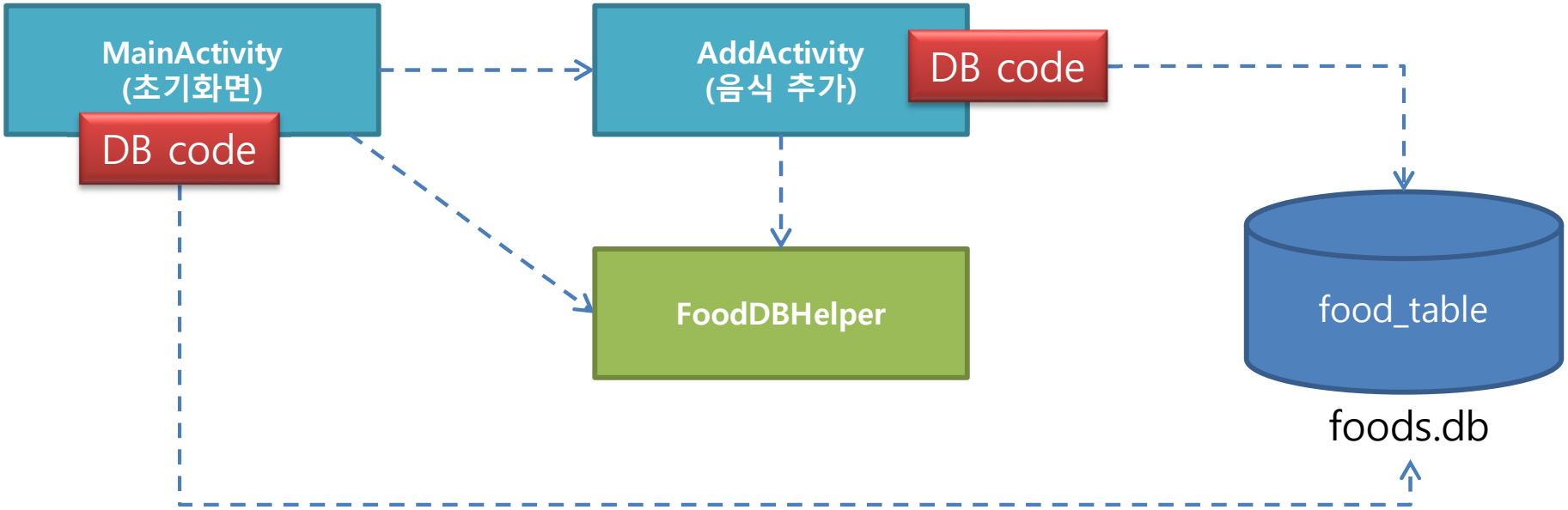
추가 수행

- 추가액티비티에서 DB에 새로운 데이터 추가 후 수행 결과를 setResult(ResultOK) 로 전달 후 종료
- 추가 성공 시 RecyclerView 갱신

삭제 수행

- RecyclerView 를 [롱클릭할 때] 삭제하도록 가정할 경우
- Adapter 에 필요한 롱클릭 외부 리스너 추가
- [롱클릭 시 대화상자 출력]
- RecyclerView 의 롱클릭 위치의 position 값으로 dto 확인
- dto 의 id 값을 사용하여 DB 삭제 수행
- 삭제 성공 시 RecyclerView 갱신

참고 - 데이터베이스 관리 클래스 (1/2)

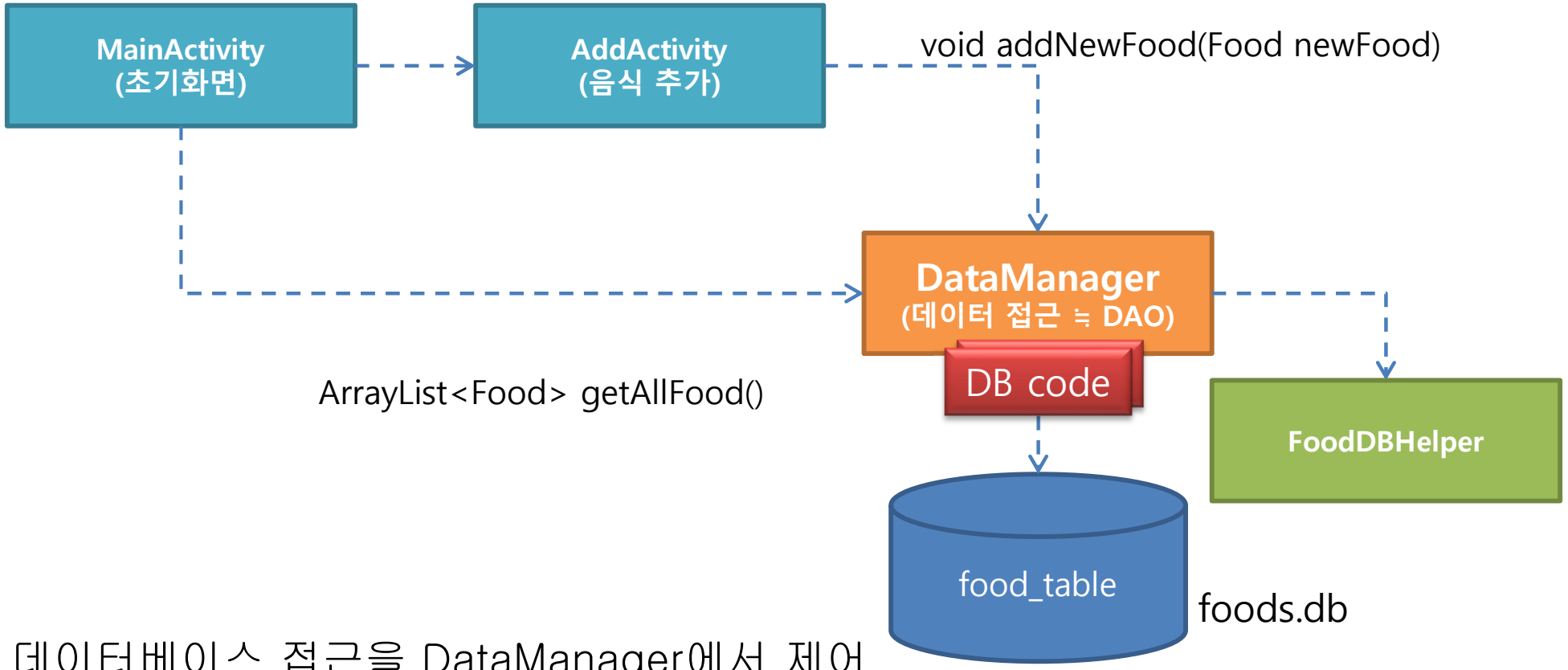


1. 데이터베이스 접근을 각각의 activity에서 제어
   - Database Helper 객체를 멤버로 생성
2. 데이터베이스 사용이 필요한 activity가 있을 때마다 다음 작업 수행
   - 데이터베이스 접근이 필요한 부분에 관련 코드 작성

★ 데이터베이스 코드가 activity마다 분산되어 있으므로 데이터베이스 관련 activity를 모두 수정해야만 함

★ activity의 본 역할은 화면 제어이므로 역할의 분할 필요

참고 - 데이터베이스 관리 클래스 (2/2)



1. 데이터베이스 접근을 DataManager에서 제어
   - Database Helper 객체를 멤버로 생성
2. 데이터베이스 사용이 필요한 activity가 있을 때마다 다음 작업 수행
   - 데이터베이스가 필요한 activity에 DataManager 객체를 멤버로 생성
   - DataManager 클래스에 필요한 메소드 추가
   - activity에서 DataManager의 메소드를 호출
   - activity의 onPause() 에서 DataManager의 close 관련 메소드 호출 필요

실습

실습1. 추가 버튼을 눌러 새로운 항목 추가 구현

- startActivityForResult() 로 AddActivity 실행
- AddActivity 에서 DB 추가 작업 실행
- 추가 성공 시 RecyclerView 갱신

실습2. RecyclerView 롱클릭 시 삭제 구현

- Adpater 에 외부 리스너 추가
- 롱클릭 시 DB에서 해당 항목 삭제 수행
- 삭제 성공 시 RecyclerView 갱신
- 추가내용 – 삭제 진행 시 [대화상자]에서 실행

실습3. 모든 DB 기능을 FoodDao 에 구현 후 FoodDao 를 사용하도록 수정