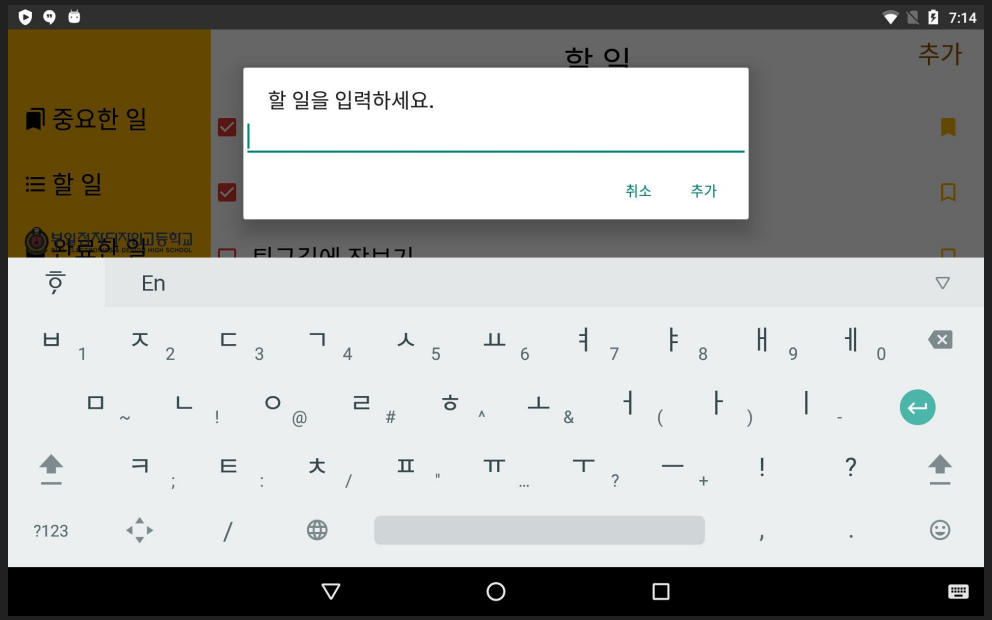
안드로이드 프로그래밍

4주차

수업 내용

- 데이터베이스?
- 왜 데이터베이스가 필요할까?
- 우리가 만들 앱에는 어떻게 쓸 수 있을까?
- 데이터베이스 기능 구현
- 할 일 등록 기능 구현
- 비동기 함수를 동기적으로 처리
- 할 일 조회 기능 구현
- 할 일 완료 기능 구현
- 할 일 북마크 기능 구현
- 할 일 삭제 기능 구현

오늘 목표



데이터베이스?

데이터베이스(영어: database, DB)는 여러 사람이 공유하여 사용할 목적으로 체계화해 통합, 관리하는 데이터의 집합이다. 작성된 목록으로써 여러 응용 시스템들의 통합된 정보들을 저장하여 운영할 수 있는 공용 데이터들의 묶음이다.

몇 개의 자료 파일을 조직적으로 통합하여 자료 항목의 중복을 없애고 자료를 구조화하여 기억시켜 놓은 자료의 집합체라고 할 수 있다.

- https://ko.wikipedia.org/wiki/데이터베이스

왜 데이터베이스가 필요할까?

- 데이터 중복 최소화하여 일관성, 무결성, 보안성, 최신 데이터 유지
  - 파일의 경우 중복된 데이터를 가질수 있음
    - ex) 같은 내용의 파일이 이름만 다른 경우, 이 문제를 해결 할 방법이 없음
  - 데이터베이스는 중복된 데이터가 없으니 저장 공간이 절약됨

왜 데이터베이스가 필요할까?

- 윈도우, 맥, 웹, 모바일 등 어떤 플랫폼, 프로그램에서 접근 가능하도록 데이터의 논리적, 물리적 독립성이 유지 됨
  - 파일로 데이터를 관리할 경우 운영체제에 따라 데이터에 접근 할 수 없음
  - 데이터베이스 드라이버를 이용하여 데이터베이스에 처리 기능을 개발할 수 있음

우리가 만들 앱에는 어떻게 쓸 수 있을까?

- 데이터베이스에 저장하기 위해 할 일 항목의 속성을 정의
  - 항목을 구분할 값을 저장하는 속성
  - 할 일 내용을 저장하는 속성
  - 완료했다는 정보를 저장하는 속성
  - 중요한 일인지에 대한 정보를 저장하는 속성

우리가 만들 앱에는 어떻게 쓸 수 있을까?

- 데이터베이스에 저장하기 위해 할 일 항목의 속성을 정의
  - 항목을 구분하기 위한 id
  - 할 일 내용을 저장할 name
  - 완료했다는 정보를 저장할 isDone
  - 중요한 일인지에 대한 정보를 저장할 isBookmark

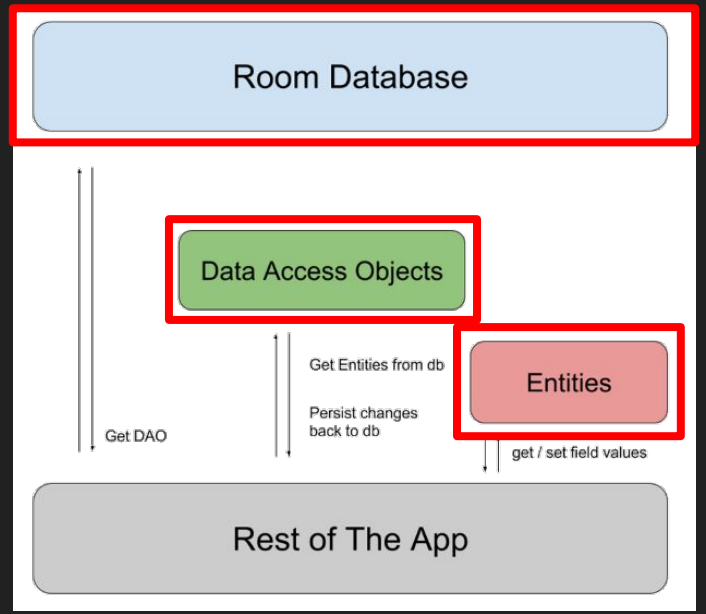
```
class TodoItem(val id: Long, val name: String, var isDone: Boolean, var isBookmark: Boolean)
```

우리가 만들 앱에는 어떻게 쓸 수 있을까?

- SQLite, MySQL, PostgreSQL, Oracle, mariaDB 등등 많은 종류가 있음
- 원격 서버에 데이터를 저장하지 않는 경우, SQLite를 이용하여 로컬 데이터베이스를 이용하여 개발
- 안드로이드에서는 Room 을 이용하여 개발 가능

우리가 만들 앱에는 어떻게 쓸 수 있을까?

- Room 을 이용한 앱의 구조



데이터베이스 기능 구현

- 데이터베이스 처리용 라이브러리(Room) 추가: app > build.gradle 파일 수정

```
apply plugin: 'kotlin-kapt'

dependencies {
    ...생략...

    def room_version = "2.2.5"
    implementation "androidx.room:room-runtime:$room_version"
    kapt "androidx.room:room-compiler:$room_version"
    implementation "androidx.room:room-ktx:$room_version"
}
```

데이터베이스 기능 구현

- Entity 생성: TodoItem.kt 수정

```kotlin
@Entity(tableName = "todo")
class TodoItem(
    @PrimaryKey(autoGenerate = true) var id: Long,
    @ColumnInfo(name = "name") var name: String,
    @ColumnInfo(name = "is_done") var isDone: Boolean,
    @ColumnInfo(name = "is_bookmark")var isBookmark: Boolean
)
```

데이터베이스 기능 구현

- Data Access object(DAO) 생성: TodoDao.kt 생성

```kotlin
@Dao
interface TodoDao {
    @Query("SELECT * FROM todo")
    suspend fun getAll(): List<TodoItem>

    @Query("SELECT * FROM todo WHERE is_bookmark = :isBookmark")
    suspend fun getAllByBookmark(isBookmark: Boolean): List<TodoItem>

    @Query("SELECT * FROM todo WHERE is_done = :isDone")
    suspend fun getAllByDone(isDone: Boolean): List<TodoItem>

    @Insert(onConflict = REPLACE)
    suspend fun insert(vararg todo: TodoItem)

    @Delete
    suspend fun delete(todo: TodoItem)

    @Query("DELETE from todo")
    suspend fun deleteAll()
}
```

데이터베이스 기능 구현

- Room Database 생성: TodoDatabase.kt 생성

```
@Database(entities = [TodoItem::class], version = 1)
abstract class TodoDatabase : RoomDatabase() {
    abstract fun todoDao(): TodoDao
}
```

데이터베이스 기능 구현

- Room Database 생성: TodoActivity.kt 수정하여 데이터베이스 객체 생성

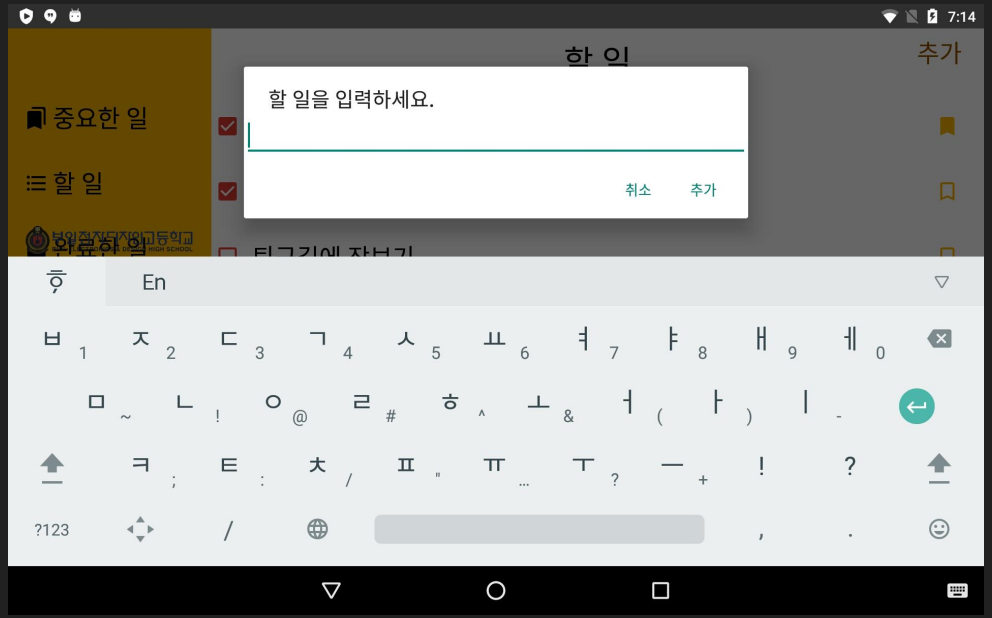
```
lateinit var db: TodoDatabase

...생략...

db = Room.databaseBuilder(
    applicationContext, TodoDatabase::class.java, "todo-database"
).build()
```

할 일 등록 기능 구현

- 화면 오른쪽 상단의 ‘추가’ 버튼을 누르면 새 창이 뜨고, 빈 칸에 할 일을 입력한 후 ‘추가’ 버튼을 눌러 창을 닫으면 데이터베이스에 할 일이 추가되는 기능

할 일 등록 기능 구현

- ‘추가’ 버튼을 누르면 새 창을 띄우는 기능



할 일 등록 기능 구현

- ‘추가’ 버튼을 누르면 새 창을 띄우는 기능: TodoActivity.kt 에 구현
    - View 를 가져오기

```
val builder = AlertDialog.Builder(this);

val dialogView = layoutInflater.inflate(R.layout.layout_insert_todo, null)

val editTodo = dialogView.findViewById<EditText>(R.id.editTodo)
val checkAlarm = dialogView.findViewById<CheckBox>(R.id.checkAlarm)
```

할 일 등록 기능 구현

- ‘추가’ 버튼을 누르면 새 창을 띄우는 기능: TodoActivity.kt 에 구현
    - View 를 설정한 뒤 다이얼로그 띄우기

```
builder.setView(dialogView)
  .setPositiveButton(getString(R.string.add)) { dialogInterface, i ->
  }
  .setNegativeButton(getString(R.string.cancel)) { dialogInterface, i ->
  }
  .show()
```

할 일 등록 기능 구현

- ‘추가’ 버튼을 누르면 새 창을 띄우는 기능: TodoActivity.kt 에 구현
    - 할 일 등록

```
val todo = TodoItem(0, editTodo.text.toString(), false, false)
insertTodo(todo)
```

할 일 등록 기능 구현

- 데이터베이스 객체로 데이터베이스에 할 일 등록하는 함수 구현

```
fun insertTodo(todo: TodoItem) = runBlocking {
    db.todoDao().insert(todo)
}
```

비동기 함수를 동기적으로 처리

- runBlocking { … } 을 쓰는 이유
- suspend 로 정의한 함수가 처리 되기를 기다리기 위해서 사용
- **데이터베이스 접근**, **네트워크 처리**, **파일 처리** 등 앱에서 요청하고 시스템의 응답을 기다려야 하는 함수에서 사용하기도 함
- 하지만 UI를 장시간 멈추게 할 수 있기 때문에 사용에 주의가 필요함
- 더 좋은 방법은 코루틴을 만들어서 메인 스레드와 별도로 동작하는 데이터베이스 접근 기능을 구현해야 함

비동기 함수를 동기적으로 처리

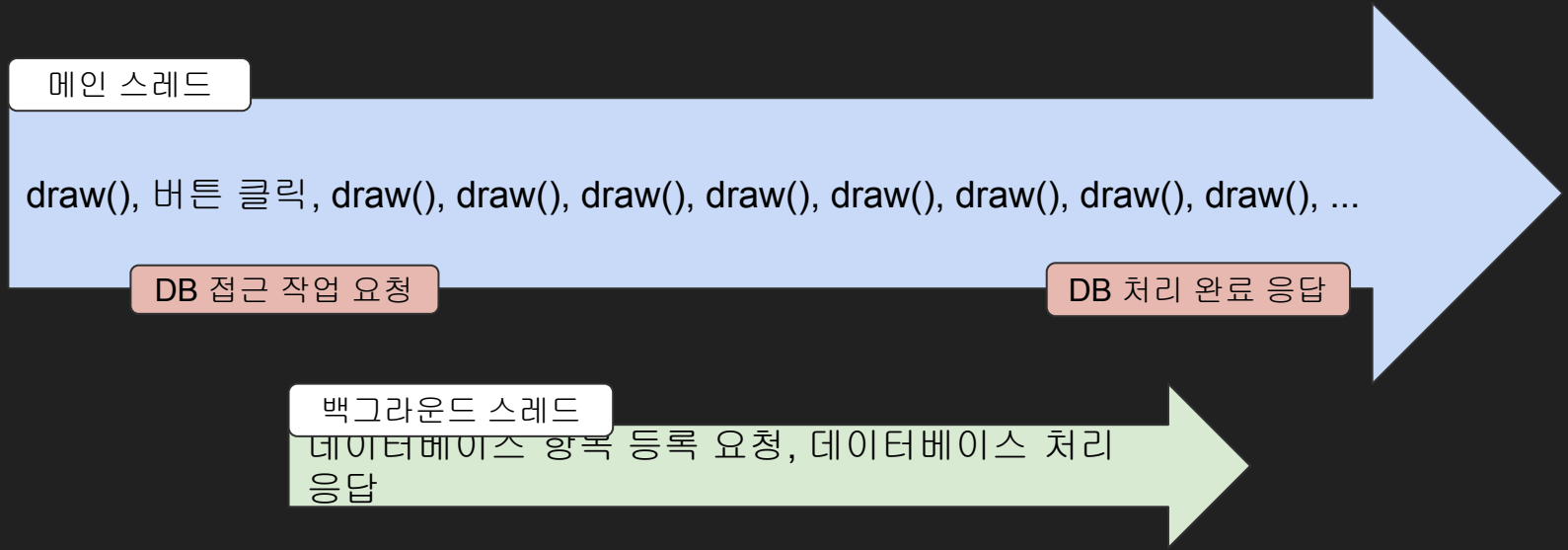
코루틴은 비동기적으로 실행되는 코드를 간소화하기 위해 Android에서 사용할 수 있는 동시 실행 설계 패턴입니다. 코루틴은 Kotlin 버전 1.3에 추가되었으며 다른 언어에서 확립된 개념을 기반으로 합니다.

Android에서 코루틴은 기본 스레드를 차단하여 앱이 응답하지 않게 만들 수도 있는 장기 실행 작업을 관리하는 데 도움이 됩니다.

- https://developer.android.com/kotlin/coroutines?hl=ko

비동기 함수를 동기적으로 처리

- 별도 스레드에서 데이터베이스 처리 기능 구현: 코루틴 이용



할 일 조회 기능 구현

- runBlocking 블록에서 데이터베이스에서 할 일 조회
- 조회한 데이터를 Adapter에 설정한 후 RecyclerView UI 업데이트

```
fun getAllTodo() = runBlocking {
    val todos = db.todoDao().getAll()
    viewAdapter.setItems(todos)
    runOnUiThread {
        viewAdapter.notifyDataSetChanged()
    }
}
```

할 일 완료 기능 구현

- checkDone 체크 박스를 눌러서 발생하는 이벤트에 리스너를 연결

```
checkDone.setOnCheckedChangeListener { compoundButton, b ->
    if (adapterPosition >= 0) {
        arrayItem[adapterPosition].isDone = b
        todoChangeListener?.onDoneChanged(arrayItem[adapterPosition])
    }
}
```

할 일 북마크 기능 구현

- toggleBookmark 체크 박스를 눌러서 발생하는 이벤트에 리스너를 연결

```
toggleBookmark.setOnCheckedChangeListener { compoundButton, b ->
    if (adapterPosition >= 0) {
        arrayItem[adapterPosition].isBookmark = b
        todoChangeListener?.onBookmarkChanged(arrayItem[adapterPosition])
    }
}
```

할 일 북마크 기능 구현

- viewAdapter 에서 발생한 이벤트에 리스너를 연결하여 데이터베이스에 업데이트

```
viewAdapter.setTodoChangeListener(object : TodoAdapter.TodoChangeListener {
    override fun onDoneChanged(todo: TodoItem) {
        insertTodo(todo)
    }

    override fun onBookmarkChanged(todo: TodoItem) {
        insertTodo(todo)
    }
})
```

할 일 삭제 기능 구현

- checkDone 체크 박스를 길게 눌러서 발생하는 이벤트에 리스너를 연결

```
checkDone.setOnLongClickListener {
    if (adapterPosition >= 0) {
        todoLongClickListener?.onLongClicked(arrayItem[adapterPosition])
    }

    true
}
```

할 일 삭제 기능 구현

- checkDone 을 길게 누르면 다이얼로그가 띄워지고 확인 버튼 누르면 데이터베이스에서 할 일 삭제

```
viewAdapter.setTodoLongClickListener(object : TodoAdapter.TodoLongClickListener {
    override fun onLongClicked(todo: TodoItem) {
        val builder = AlertDialog.Builder(this@TodoActivity);

        builder.setMessage(getString(R.string.msg_want_delete))
            .setPositiveButton(getString(R.string.delete)) { dialogInterface, i ->
                deleteTodo(todo)
            }
            .setNegativeButton(getString(R.string.cancel)) { dialogInterface, i ->
            }
            .show()
    }
})
```