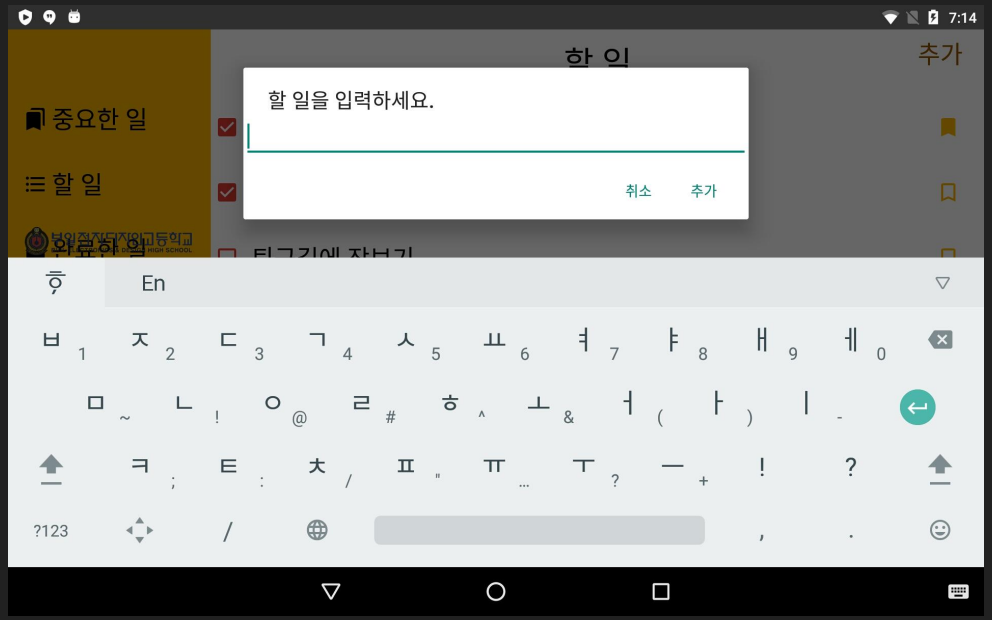
안드로이드 프로그래밍

3주차

수업 내용

- 메인 화면 할 일 목록 영역 만들기
- 데이터베이스?
- 왜 데이터베이스가 필요할까?
- 우리가 만들 앱에는 어떻게 쓸 수 있을까?
- 데이터베이스 기능 구현
- 할 일 등록 기능 구현

오늘 목표



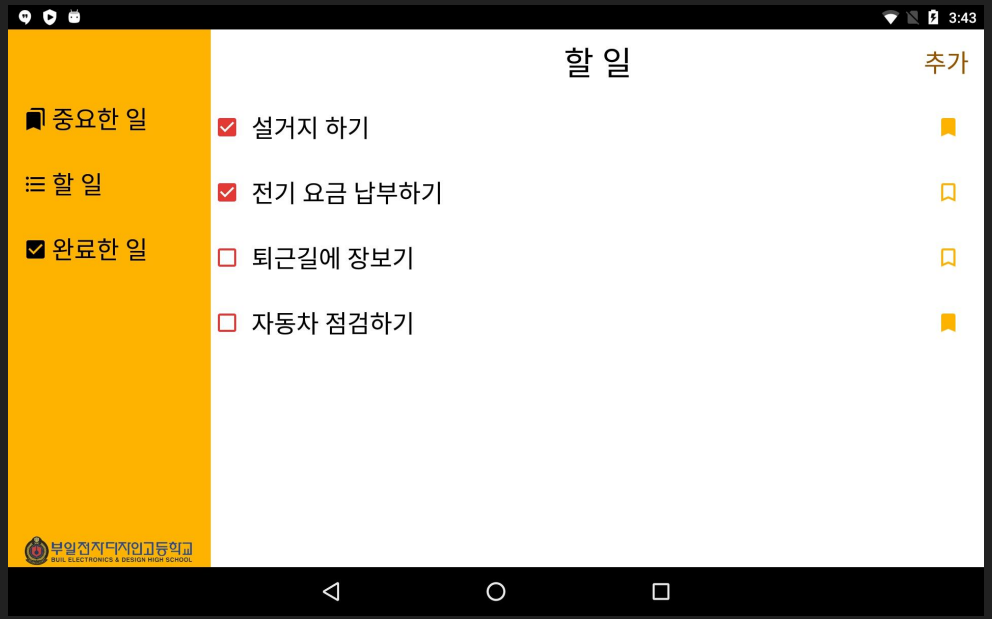
메인 화면 할 일 영역 만들기

- 왼쪽 메뉴를 클릭하면 상단의 제목을 변경 시키기

```
groupMenu.setOnCheckedChangeListener { radioGroup, i ->
    if (i == R.id.btnBookmark) {
        textTitle.text = getString(R.string.bookmark)
    } else if (i == R.id.btnTodo) {
        textTitle.text = getString(R.string.todo)
    } else {
        textTitle.text = getString(R.string.done)
    }
}
```

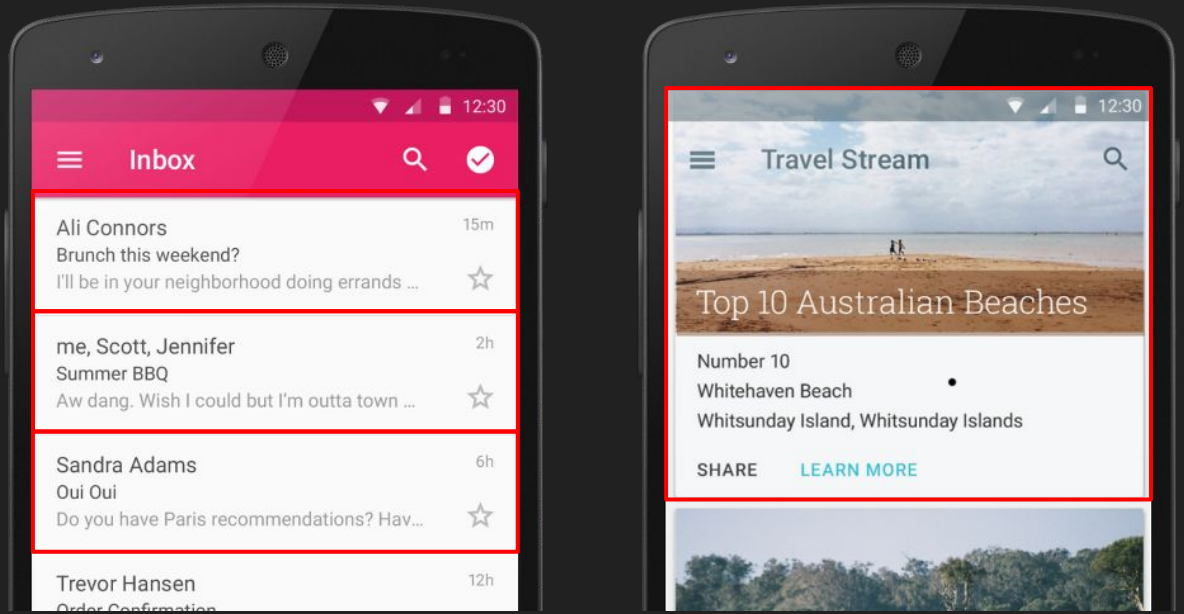
메인 화면 할 일 영역 만들기

- 할 일 목록을 표시해 줄 RecyclerView 추가



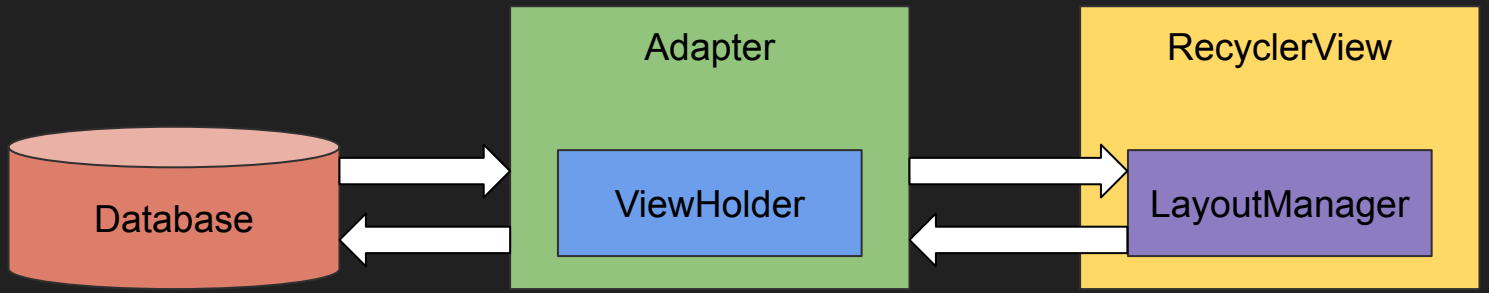
메인 화면 할 일 영역 만들기

- 반복되는 형태의 데이터를 보여주기 위해 **RecyclerView** 이용



메인 화면 할 일 영역 만들기

- 반복되는 형태의 데이터를 보여주기 위해 **RecyclerView** 이용



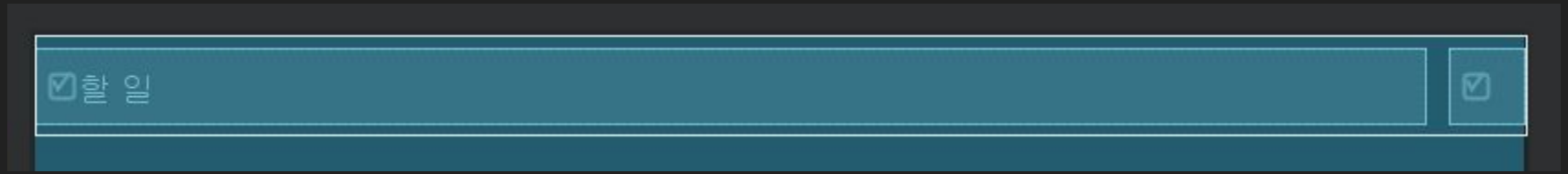
메인 화면 할 일 영역 만들기

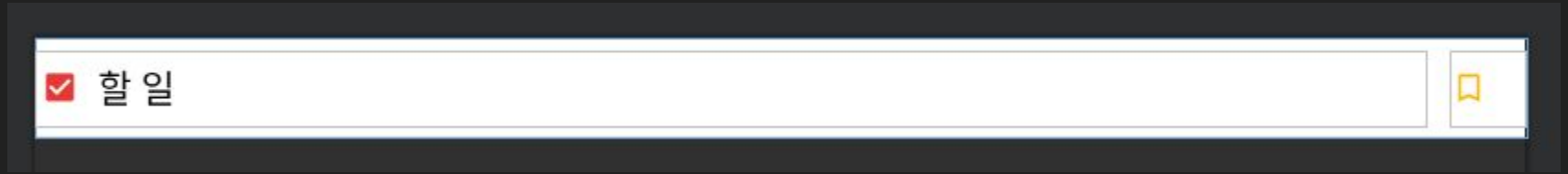
- 할 일 목록을 표시해 줄 RecyclerView 추가

```
<android.support.v7.widget.RecyclerView
        android:id="@+id/rvTodo"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:scrollbars="vertical"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@id/textTitle"
        app:layout_constraintBottom_toBottomOf="parent"/>
```

메인 화면 할 일 영역 만들기

- RecyclerView 아이템들의 모습은 비슷하기때문에 레이아웃을 만들어서 재사용 가능
- 할 일 항목을 표시할 레이아웃 만들기
  - res > layout > layout_todo_item.xml 파일 만들기





메인 화면 할 일 영역 만들기

- layout_todo_item.xml 만들기

```
<android.support.constraint.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="64dp"
        android:background="@android:color/white">

</android.support.constraint.ConstraintLayout>
```

메인 화면 할 일 영역 만들기

- layout_todo_item.xml 만들기

```
<CheckBox
      android:id="@+id/checkDone"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:layout_marginEnd="16dp"
      android:buttonTint="@color/colorAccent"
      android:checked="true"
      android:padding="8dp"
      android:textSize="24sp"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintEnd_toStartOf="@+id/toggleBookmark"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent"/>
```

메인 화면 할 일 영역 만들기

- layout_todo_item.xml 만들기

```
<ToggleButton
        android:id="@+id/toggleBookmark"
        android:layout_width="48dp"
        android:layout_height="48dp"
        android:background="@drawable/selector_bookmark"
        android:backgroundTint="@color/colorPrimary"
        android:textOn=""
        android:textOff=""
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>
```

메인 화면 할 일 영역 만들기

- 할 일 항목에 대한 정보를 담을 TodoItem.kt 생성
  - 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)
```

메인 화면 할 일 영역 만들기

- TodoAdapter.kt 생성

```
class TodoAdapter() : RecyclerView.Adapter<TodoAdapter.TodoViewHolder>() {
    private lateinit var mContext: Context

    private var arrayItem: MutableList<TodoItem> = ArrayList()

    class TodoViewHolder(v: View) : RecyclerView.ViewHolder(v) {
        val checkDone: CheckBox = v.checkDone
        val checkBookmark: CheckBox = v.checkBookmark
    }

    ...생략...
}
```

메인 화면 할 일 영역 만들기

- TodoAdapter.kt 에 아이템 클릭 이벤트 생성

```
private var todoChangeListener: TodoChangeListener? = null

fun setTodoChangeListener(todoChangeListener: TodoChangeListener) {
    this.todoChangeListener = todoChangeListener
}

interface TodoChangeListener {
    fun onDoneChanged(todo: TodoItem)
    fun onBookmarkChanged(todo: TodoItem)
}
```

메인 화면 할 일 영역 만들기

- TodoAdapter.kt 에 아이템 클릭 이벤트 생성

```
holder.checkDone.setOnCheckedChangeListener { compoundButton, b ->
    item.isDone = b
    todoChangeListener?.onDoneChanged(item)
}

holder.toggleBookmark.setOnCheckedChangeListener { compoundButton, b ->
    item.isBookmark = b
    todoChangeListener?.onBookmarkChanged(item)
}
```

메인 화면 할 일 영역 만들기

- RecyclerView에 들어갈 가짜 데이터 만들기

```
val arrayDummy = mutableListOf<TodoItem>()
arrayDummy.add(TodoItem(1, "설거지 하기", true, true))
arrayDummy.add(TodoItem(2, "전기 요금 납부하기", true, false))
arrayDummy.add(TodoItem(3, "퇴근길에 장보기", false, false))
arrayDummy.add(TodoItem(4, "자동차 점검하기", false, true))
```

메인 화면 할 일 영역 만들기

- TodoActivity.kt 에 코드 추가하여 RecyclerView 사용 준비하기

```
val viewManager = LinearLayoutManager(this)
val viewAdapter = TodoAdapter()
viewAdapter.setItems(arrayDummy)
recyclerTodo.apply {
    setHasFixedSize(true)
    layoutManager = viewManager
    adapter = viewAdapter
}
```

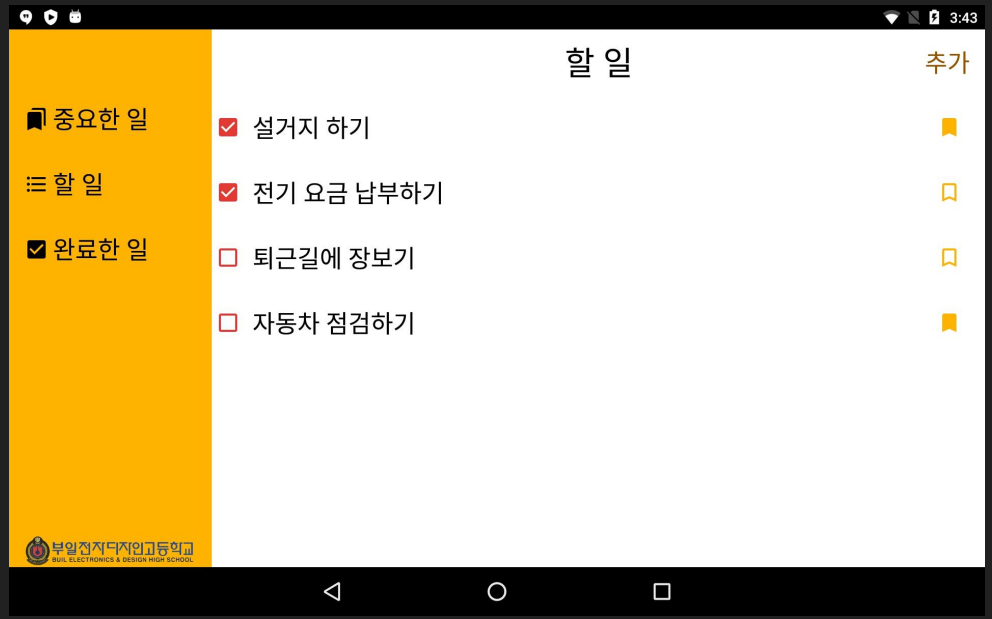
메인 화면 할 일 영역 만들기

- TodoActivity.kt 에 아이템 클릭 이벤트 연결

```
viewAdapter.setTodoChangeListener(object : TodoAdapter.TodoChangeListener {
    override fun onDoneChanged(todo: TodoItem) {
        Toast.makeText(this@TodoActivity, "Done 클릭", Toast.LENGTH_SHORT).show()
    }

    override fun onBookmarkChanged(todo: TodoItem) {
        Toast.makeText(this@TodoActivity, "Bookmark 클릭", Toast.LENGTH_SHORT).show()
    }
})
```

메인 화면 할 일 영역 만들기

- 실행



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

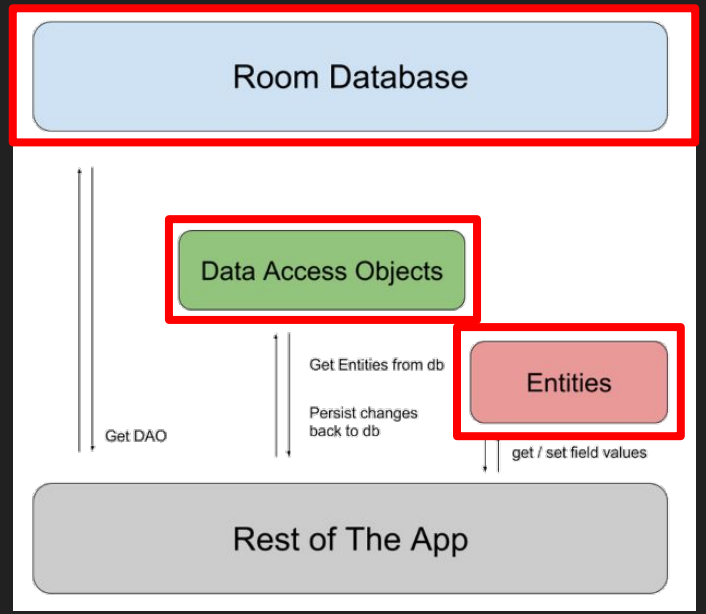
```
class TodoItem(val id: Long, val name: String, var isDone: Boolean, var isBookmark: Boolean)
```

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

```
@Entity(tableName = "todo")
class TodoItem(
    @PrimaryKey(autoGenerate = true) val  id: Long,
    @ColumnInfo(name = "name") val  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")
    fun getAll(): List<TodoItem>

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

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

    // 데이터베이스에 같은 데이터가 있으면(id 값이 같은 데이터가 있으면) 덮어쓰기, 수정 기능처럼 사용 가능
    @Insert(onConflict = REPLACE)
    fun insert(vararg todo: TodoItem)

    @Delete
    fun delete(todo: TodoItem)

    @Query("DELETE from todo")
    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

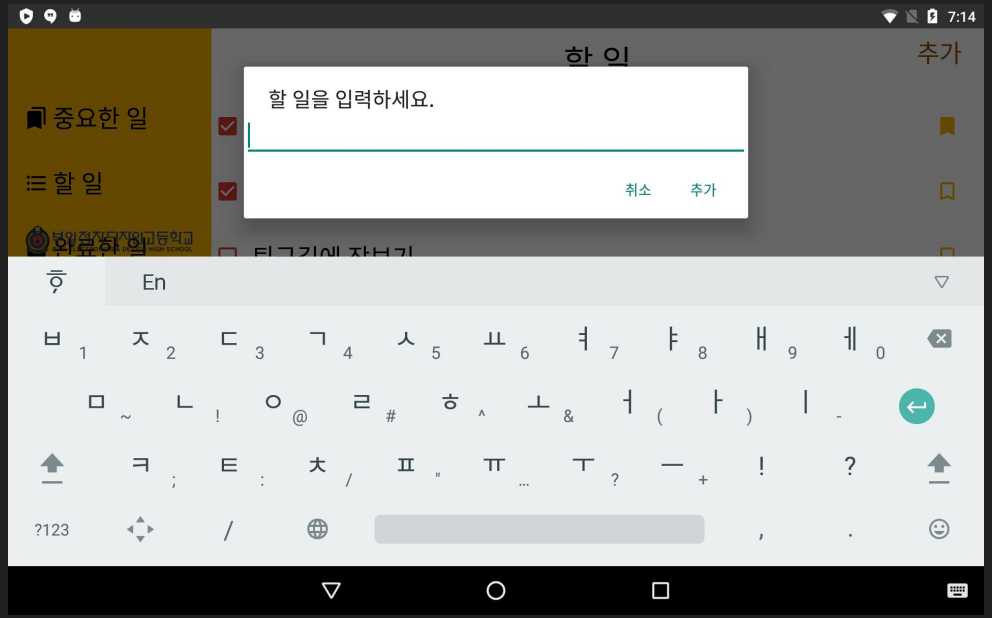
```
lateinit var db: TodoDatabase

...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

```
btnAdd.setOnClickListener {
    val editText = EditText(this)
    val dialog = AlertDialog.Builder(this)
    dialog.setTitle(getString(R.string.msg_please_insert))
    dialog.setPositiveButton(getString(R.string.add)) {dialogInterface, i ->
        // 할 일 등록
    }
    dialog.setNegativeButton(getString(R.string.cancel), null)
    dialog.create()
    dialog.setView(editText)
    dialog.show()
}
```

할 일 등록 기능 구현

- 전에 생성한 데이터베이스 객체로 데이터베이스에 할 일 등록

```
// 할 일 등록
insertTodo(editText.text.toString())

...생략...

fun insertTodo(jobName: String) {
    val todo = TodoItem(0, jobName, false, false)
    db.todoDao().insert(todo)
}
```

할 일 등록 기능 구현

- 하는 코드를 추가한 뒤에 실행하면 다음과 같은 에러가 남

```
java.lang.IllegalStateException: Cannot access database on the main thread since it may potentially lock the UI for a long period of time.
    at androidx.room.RoomDatabase.assertNotMainThread(RoomDatabase.java:267)
    at androidx.room.RoomDatabase.beginTransaction(RoomDatabase.java:351)
    at com.kimjunu.todayiamdone.TodoDao_Impl.insert(TodoDao_Impl.java:75)
    at com.kimjunu.todayiamdone.TodoActivity$onCreate$2$dialog$1.onClick(TodoActivity.kt:47)
    at androidx.appcompat.app.AlertController$ButtonHandler.handleMessage(AlertController.java:167)
    at android.os.Handler.dispatchMessage(Handler.java:102)
    at android.os.Looper.loop(Looper.java:148)
    at android.app.ActivityThread.main(ActivityThread.java:5417) <1 internal call>
    at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:726)
    at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:616)
```

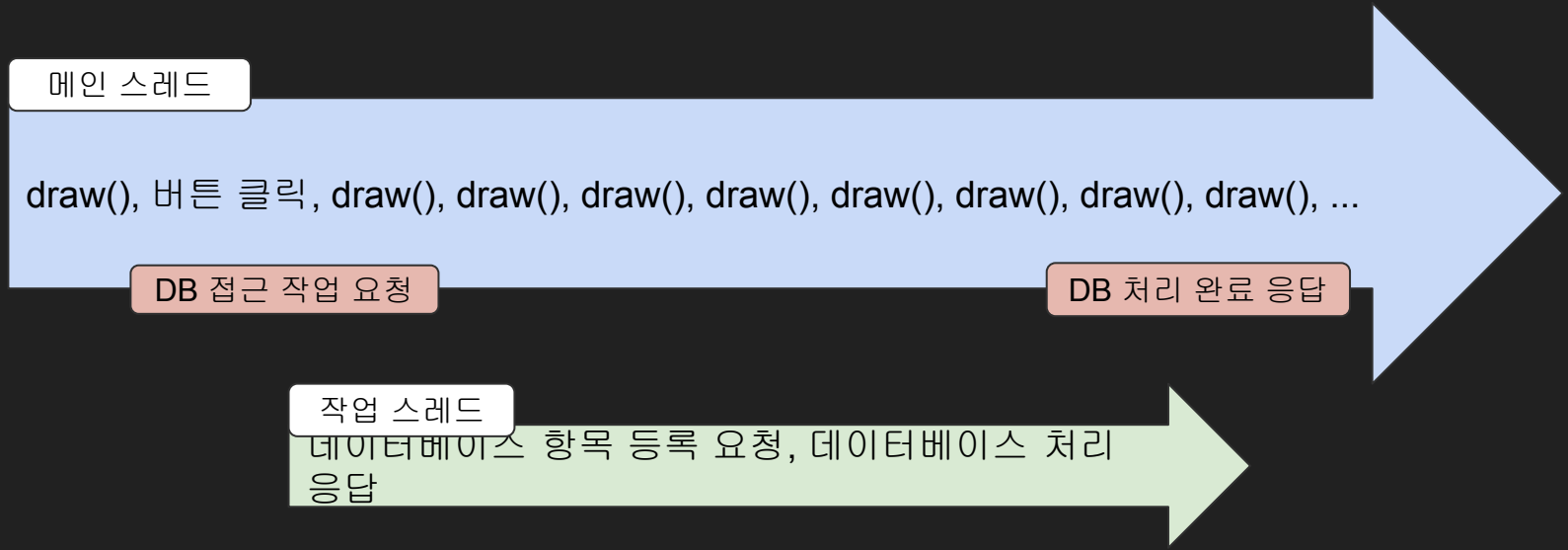
할 일 등록 기능 구현

Cannot access database on the main thread since it may potentially lock the UI for a long period of time.

- UI를 장시간 멈추게 할 수 있기 때문에 메인 스레드(UI 스레드)에서 데이터베이스에 접근할 수 없음
- 데이터베이스 접근, 네트워크 처리, 파일 처리 등 앱에서 요청하고 시스템의 응답을 기다려야 하기때문에 앱의 화면을 멈추면서까지 무한정 기다릴 수 없음
- 그래서 메인 스레드와 별도의 스레드를 하나 더 만들어서 데이터베이스 접근 기능을 구현해야 함

할 일 등록 기능 구현

- 별도 스레드에서 데이터베이스 처리 기능 구현: 코루틴 이용



할 일 등록 기능 구현

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

할 일 등록 기능 구현

- 비동기 처리를 위한 coroutine 라이브러리 추가: app > build.gradle 파일 수정

```
dependencies {
    ...생략...

    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-core:1.3.0'
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.3.0'
}
```

할 일 등록 기능 구현

- 데이터베이스에 접근하는 코드를 Coroutine 코드로 감싸기
- 백그라운드 스레드에서 실행하기 위해 Dispatcher.IO 명시

```
fun insertTodo(jobName: String) {
    CoroutineScope(Dispatchers.IO).launch {
        val todo = TodoItem(0, jobName, false, false)
        db.todoDao().insert(todo)
    }
}
```