# ROOMDATABASE & ADVANCED KOTLIN CHEATSHEET

## CRUD Operations with RoomDatabase

### Create:

Add new data.

```
@Insert
fun addWish(wish: Wish)
```

### Read:

Retrieve data.

```
@Query("SELECT * FROM wish_table")
fun getAllWishes(): List<Wish>
```

### Update:

Modify existing data.

```
@Update
fun updateWish(wish: Wish)
```

### Delete:

Remove data.

```
@Delete
fun deleteWish(wish: Wish)
```

## Repositories & Coroutines

**Repository:** Mediator between data sources and business logic.

### Coroutine Example:

```
@Query("SELECT * FROM wish_table")
fun getAllWishes(): Flow<List<Wish>>
```

**Lateinit:** For non-null properties initialized later.

```
lateinit var viewModel:
MyViewModel
```

## Entity & DAO

### Entity:

Represents a table.

```
@Entity(tableName = "wish_table")
data class Wish(
    @PrimaryKey(autoGenerate =
true) val id: Int,
    @ColumnInfo(name = "wish_desc")
val description: String
)
```

### DAO:

Interface for database operations.

```
@Dao
interface WishDao {
@Insert fun addWish(wish: Wish)
@Query("SELECT * FROM wish_table") fun
getAllWishes():
List<Wish>
@Update fun updateWish(wish: Wish)
@Delete fun deleteWish(wish: Wish)
}
```

## Object Graph & Dependency Injection

**Object Graph:** Manages dependencies.

### Dependency Injection Example:

```
@AndroidEntryPoint
class MyActivity : AppCompatActivity()
{
@Inject lateinit var viewModel:
MyViewModel
}
```

## Setting Up RoomDatabase

### Database Setup:

```
@Database(entities = [Wish::class],
version = 1)
abstract class AppDatabase :
RoomDatabase() {
abstract fun wishDao(): WishDao
}
```

## Abstract Classes in

**Definition:** Base class not meant to be instantiated.

### Example:

```
abstract class WishDao {
@Insert abstract fun addWish(wish:
Wish)
@Query("SELECT * FROM wish_table")
abstract fun getAll
Wishes(): List<Wish>
}
```

## Singleton Pattern

**Definition:** Ensures a class has only one instance.

### Example:

```
object DatabaseHelper {
fun getInstance(): DatabaseHelper =
this
}
```

## Setting Up RoomDatabase

### Application Class:

```
class MyApp : Application() {
    override fun onCreate() {
        super.onCreate()
        // Initialize dependencies
    }
}
```

### Snackbar Example:

```
Snackbar.make(view, "Item Deleted",
Snackbar.LENGTH_LONG).
setAction("Undo") {
// Handle undo action
}.show()
```

### Swipe to Delete Example:

```
val itemTouchHelperCallback = object :
ItemTouchHelper.Sim
pleCallback(0, ItemTouchHelper.LEFT) {
override fun onSwiped(viewHolder:
RecyclerView.ViewHol
der, direction: Int) {
// Handle swipe action
}
}
ItemTouchHelper(itemTouchHelperCallbac
k).attachToRecyclerV
iew(recyclerView)
```

tutorials.EU