INTRODUCING ROOM DATABASE

TOPICS

- Review **Storage Options in android**
  - **Files**
  - **Shared Preferences**
- Database in android
- Room database
  - Entity class
  - Data access object class
  - Database class
- Examples

STORAGE OPTIONS IN ANDROID

- **File IO**
  - Internal file storage: Store app-private files on the device
  - External file storage: Store files on the shared external file system. This is usually for shared user files, such as photos.
- **Shared preferences : store key /value pairs**
- **SQLite Databases   Store structured data in a private database.**
- **Network Connection**   Store data on the web with your own network server.

REVIEW
SHARED PREFERENCES

- Uses Key value pair
- We can save and retrieve key-value pairs of primitive data types
- Android uses XML Files for Shared preferences

Key	Value
age	22
password	abcd12
userName	user
address	Toronto

SAVING DATA IN DATABASES

- Relational database organizes data into table (columns and rows) with a unique key identifying each row. Rows
-

id	Name	phone	email	address
1	Joe S.	555-555-5555	joe@email.ca	Oakville
2	Tim S.	555-444-3335	tim@email.ca	Ottawa
3	Jane S.	555-789-5555	jane@email.ca	Toronto

PRIMARY KEY , FOREIGN KEY

Users Info

id	Name	email	add
500	Jane	j@ja.ca	----
501	Joe	-----	------
502	---	----	---

Courses

id	T1_id	C1	C2
200	5001	PRO6--	___
201	---	----	---
203	---	----	---

USING A DATABASE IN ANDROID

- Android offer SQLite database
- SQLite is an open-source SQL database that stores data to a text file on a device
- It support all relational database features

- Direct use of SQLite requires a huge amount of code dedicated to
- - converting the SQLite structured data into Java objects,
- - preparing SQL statements to store those objects back into the database.
-

WHAT IS ROOM

- Persistence library provides an abstraction layer over SQLite
- allows you to define
  - - object model (Entity class)
  - - SQL queries you want to execute (Dao class)
- Room API will create database and implements the boilerplate Data Access Objects (DAO) classes.
- Room is an excellent choice because all the heavy lifting is done at compile time by generating source code for your application.

ADDING ROOM TO THE PROJECT

Add Room Library to the project by doing the following

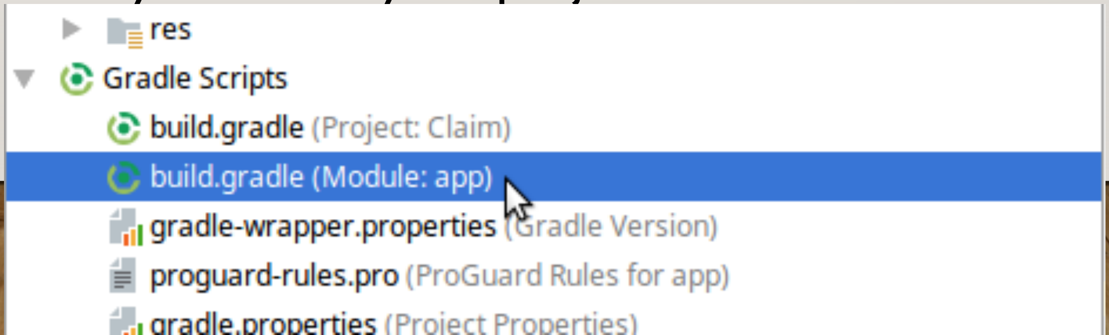
1- Open Gradle file-> build.gradle (module.ap)

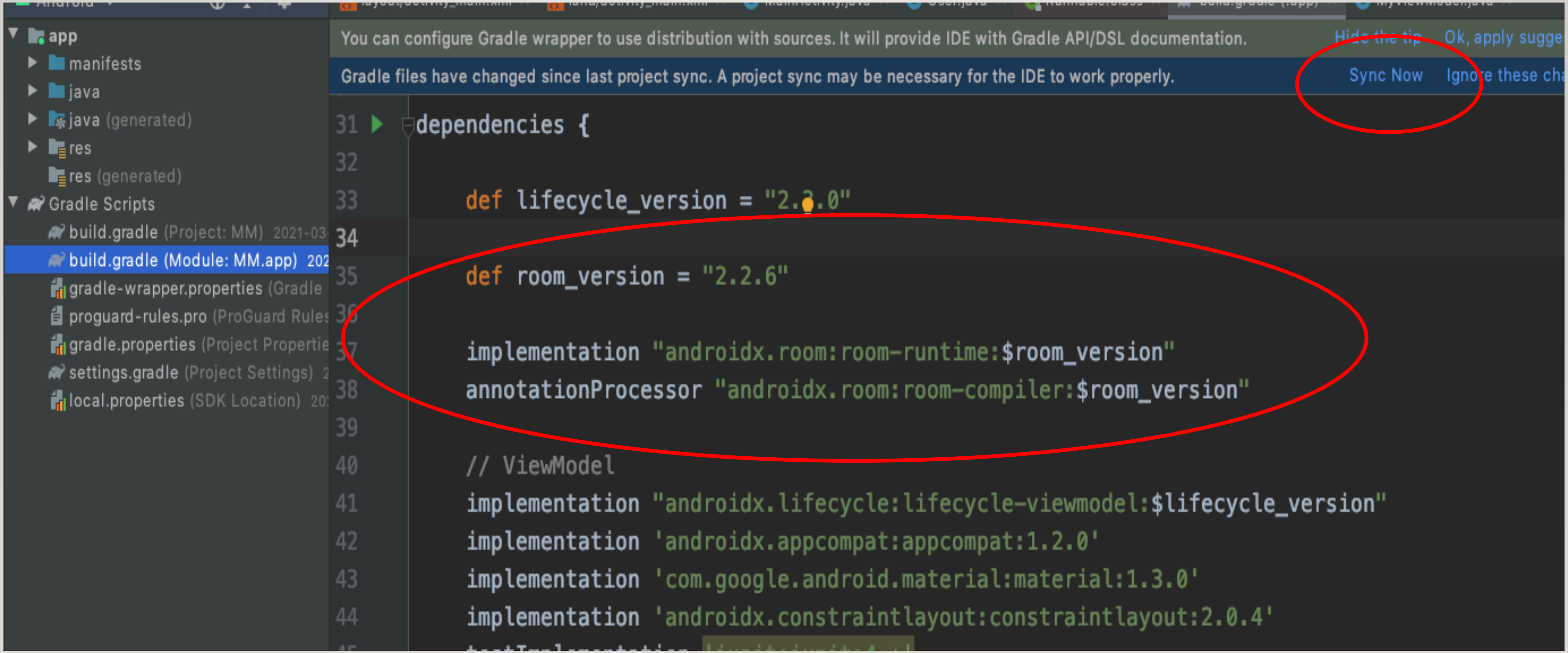
2- Add the following lines to dependencies

- def room_version = "2.2.6"

```
implementation "androidx.room:room-runtime:$room_version"
annotationProcessor "androidx.room:room-compiler:$room_version"
```

- Click on Sync Now link at the top of the editor to synchronize your project with its Gradle file


res
Gradle Scripts
build.gradle (Project: Claim)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)



app
manifests
java
java (generated)
res
res (generated)
Gradle Scripts
build.gradle (Project: MM) 2021-03
build.gradle (Module: MM.app) 202
gradle-wrapper.properties (Gradle
proguard-rules.pro (ProGuard Rules
gradle.properties (Project Propertie
settings.gradle (Project Settings) 2
local.properties (SDK Location) 20
Hide the tip
Ok, apply sugge
Sync Now
Ignore these cha

You can configure Gradle wrapper to use distribution with sources. It will provide IDE with Gradle API/DSL documentation.

Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

```
dependencies {

    def lifecycle_version = "2.3.0"

    def room_version = "2.2.6"

    implementation "androidx.room:room-runtime:$room_version"
    annotationProcessor "androidx.room:room-compiler:$room_version"

    // ViewModel
    implementation "androidx.lifecycle:lifecycle-viewmodel:$lifecycle_version"
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.3.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
```

CREATE THE FOLLOWING CLASSES

- - **Entity Model class** :
    - Represents a table within the database.
- - **Dao class** (data access object)
    - Contains the methods used for accessing the database.
- - **Database class**

ENTITY CLASS

- An Entity class represents a table in database
- Must be annotated with @Entity
- Fields must either be public or have getters and setters
- At least one field must be marked as a primary key using the @PrimaryKey annotation
- Use @ColumnInfo if you want to assign column name

```java
import androidx.room.Entity;
import androidx.room.PrimaryKey;

@Entity
public class User {

    @PrimaryKey
    private int id;
    @ColumnInfo(name="user_name")
    private String name;

    @ColumnInfo(name="email")
    private String email;

//add  getters, setters and constructor
}
```

DAO (DATA ACCESS OBJECT) CLASS

- Will be used to define queries to
- - write data to database
- - retrieve data from databases
- - delete and update queries

```
@Dao
public abstract class UserDao {

    @Insert
    public abstract void insert(User user);

    @Delete
    public abstract void delete(User  user);

    @Update
    public abstract void update(User  user);

    @Query("select * from User")
    public abstract List<User> getAll();

    @Query("select * from User where id = :id")
    public abstract User getUser( int id);

}
```

DATABASE CLASS:

- Abstract class that extends RoomDatabase.
- Contains the database holder
- annotated with @Database
- Contain abstract method to retrieve the Data Access Object implementations you created earlier (these methods will be implemented by the subclass generated by Room)
-

DATABASE CLASS

```
@androidx.room.Database(entities = User.class,version = 1)
public abstract class Database  extends RoomDatabase {

    public abstract  UserDao userDao();
    private static Database instance ;

    public static Database getInstance(Context context){

      if(instance == null){

        instance = Room.databaseBuilder(context, Database.class,"User_DB")
                     .allowMainThreadQueries()
                     .build();
      }

        return  instance;
    }
```

GETTING DATABASE INSTANCE

```
// reference to db
MyDatabase userDb;

// get instance inside onCreate
userDb = MyDatabase.getInstance(this);
```

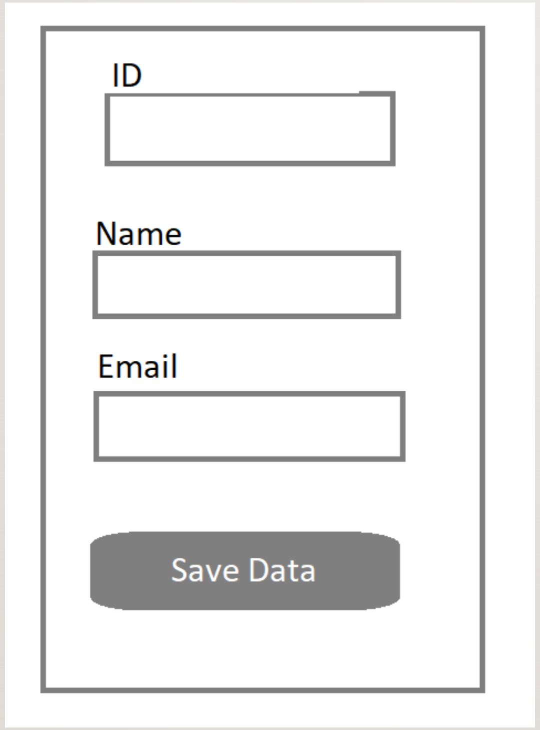
ADDING USER TO DB

```java
public void saveData(View view) {
   int idStr = idEt.getText().toString();
   int id = Integer.valueOf(idStr);
   String name = nameEt.getText().toString();
   String email = emailEt.getText().toString();
   User user = new User(id, name , email);
   userDb.userDao().insert(user);
}
```

READING USERS INFO

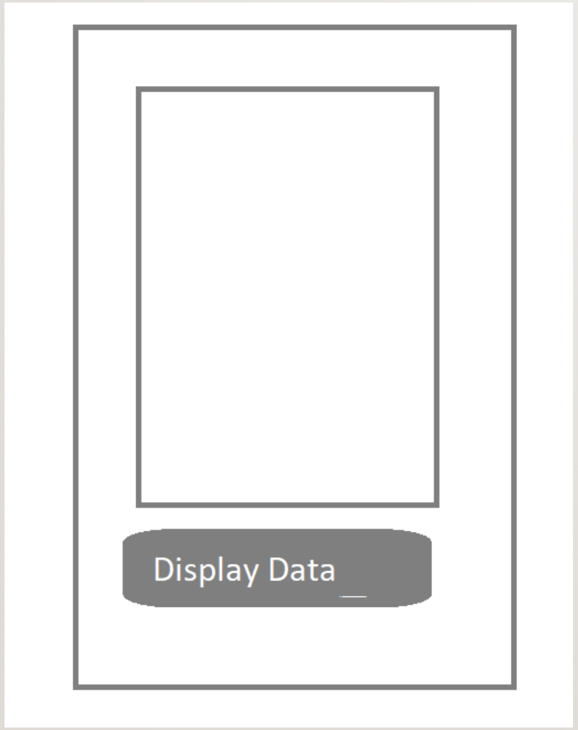
```
public void readData(View view) {
List<User> users = userDb.userDao().getAll();
ListView listView = findViewById(R.id.listView);
ArrayAdapter adapter
= new ArrayAdapter(this,
  android.R.layout.simple_list_item_1 ,users);
listView.setAdapter(adapter);
}
```

EXAMPLE



- Create android app that contain two activities
- Main Activity uses the layout shown in figure and add user information(ID, name and email) using room database

EXAMPLE



- Second activity: uses the layout shown in the figure and display users info in a listView

REFERENCES

- https://developer.android.com/training/data-storage/room/#java
- https://developer.android.com/jetpack/androidx/releases/room
- https://developer.android.com/reference/androidx/room/package-summary