**ReadCSVFile**

public voidreadCSv() {  
**try** {

InputStream inputStream = getResources().openRawResource(R.raw.*students*);  
BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(inputStream));

String header =reader.readLine();

**if**(header!=**null**) { *// Treat Header*}

String studentLine = reader.readLine();  
  
**while** (studentLine!=**null**) {

studentLine= reader.readLine();  
String[] eachStud = studentLine.split(**","**);

Student newStud = **new**Student(eachStud[0] , eachStud[1],eachStud[2] );  
**studentList**.add(newStud);

}

inputStream.close();

} **catch** (Exception err) {

Log.*d*(**TAG**, **"readCSv: "**);  
 }  
}

\*OBS\* If image is only name dogName instead of R.Drawable.DogName should parse as   
int dogDrawable = getResources().getIdentifier(eachDogFields[1],"drawable",getPackageName());

**View Binding**

1. Build.Gradle -> buildFeatures **{** viewBinding true; **}**
2. In the activity->  
    ActivityMainBinding binding = ActivityMainBinding.*inflate*(getLayoutInflater());  
   setContentView(binding.getRoot());
3. Binding methods will allow access to item in layout. (e.g. binding.textViewTitle)

**Database**

1. **Add dependencies in build.gradle**
2. **Create a Class with the Entity Anotation and Column informations (Make sure constructor and getters/setters respects the annotation of the entity e.g Not Null )**

@Entity(tableName = "students")

public class Student {

@NonNull

@PrimaryKey

@ColumnInfo(name="studentid")

private String StudId;

@ColumnInfo(name="studentname")

private String StudName;

@ColumnInfo(name="studentdept")

private String StudDept;

@NonNull

public String getStudId() {

return StudId;

}

1. **Create Interface classNameDao with the queries Insert,select,delete update and etc…**

@Dao

public interface StudentDao {

@Insert(onConflict = OnConflictStrategy.IGNORE)  
void insertStudents(Student... students);

@Insert(onConflict = OnConflictStrategy.REPLACE)  
Long[] insertStudentFromList(List<Student> Student);

@Insert(onConflict = OnConflictStrategy.REPLACE)  
void insertOneStudent(Student student);

@Query("SELECT \* FROM students")  
List<Student> GetAllStudents();

@Delete  
int deleteOneStudent(Student student);

@Delete  
int deleteStudentFromList(List<Student> students);

@Query("Delete From students")  
void deleteAllStudents();

@Query("Delete from students where studentid= :id")  
int deleteStudentWithId(String id);

@Query("UPDATE grades SET studgrade =studgrade\*1.1 WHERE studentid =:StudId")  
int IncreaseGradeForOneStudent(String StudId);

@Query("UPDATE grades SET studgrade=studgrade\*1.1 " + "WHERE studentid IN (:StudIds) "

+ "AND courseid = :CourseId")  
int IncreaseGradesForStudentsInCourse(List<String> StudIds, String CourseId);

1. **Create Database Controller**

@Database(entities = {Student.class, Grade.class}, version = 1, exportSchema = false)

public abstract class CollegeDatabase extends RoomDatabase {

public abstract StudentDao studentDao();

public abstract GradeDao gradeDao();

public abstract StudGradeDao studentGradeDao();

}

1. **Main Activity**
2. Create the main reference for the Database Controller Class

DatabaseController db;

1. Create the DB object on your On Create main Method  
   db =Room.databaseBuilder(getApplicationContext(),CollegeDatabase.class,"college.db").build();
2. Create Method with new Thread to execute the DB commands

private void insertDatabase(ActivityMainBinding binding) {

ExecutorService executorService = Executors.newSingleThreadExecutor();

executorService.execute(() -> {

cdb.studentDao().insertStudentFromList(StudentList);

cdb.gradeDao().insertGrades(GradeList);

List<Student> StudentsFromDB = cdb.studentDao().GetAllStudents();

runOnUiThread(() -> {

binding.listViewStudents.setAdapter(new StudentAdapter(StudentsFromDB));

});

});

}

**RecyclerView**

1. Create a RecyclerViewAdapter Class
2. Create an Inner Clas for the ImageViewHolder extending RecyclerView.ViewHolder

This class will be responsible for binding Items and eventHandlers

public class AnimalHolder extends RecyclerView.ViewHolder {  
 LayoutGalleryitemBinding binding;  
 public AnimalHolder(@NonNull View itemView) {  
 super(itemView);  
 }  
 public AnimalHolder(@NonNull View itemView, LayoutGalleryitemBinding binding) {  
 super(itemView);  
 this.binding = binding;  
 this.binding.getRoot().setOnClickListener(view -> {  
 onItemClickListener.onItemClick(getAdapterPosition());  
 });  
 }  
  
}

1. Create an Inner Interface OnItemClickListener

public interface OnItemClickListener{

public void onItemClick(int i);

}

1. Extend the MainClass Adapter with RecyclerView.Adapater<classname.ImageViewHolder>
2. Create Properties (List with the data, Selected index and the InterFace On ItemClickListener), Constructor with onItemClickListner and List, Getters and Setters
3. OnCreateViewHolder with LayoutInflator using the ViewBinding and returning the Holder
4. OnBindViewHolder to set the data using holder.binding….
5. getItemCount with list.size();

\*Don’t Forget The NotifyDataSetChanged on the RecyclerView\*

public recyclerViewAdapter(List<animal> animalList, OnItemClickListener onItemClickListener) {  
 this.animalList = animalList;  
 this.onItemClickListener = onItemClickListener;  
}  
  
@NonNull  
@Override  
public AnimalHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 LayoutGalleryitemBinding binding = LayoutGalleryitemBinding.*inflate*(LayoutInflater.*from*(parent.getContext()),parent,false);  
 AnimalHolder holder = new AnimalHolder(binding.getRoot(),binding);  
 return holder;  
}  
  
@Override  
public void onBindViewHolder(@NonNull AnimalHolder holder, int position) {  
 holder.binding.txtViewExtItem.setText(animalsList.get(position).getName());  
 holder.binding.imgViewExtItem.setImageResource(animalsList.get(position).getAnimalPicture());  
  
}  
  
@Override  
public int getItemCount() {  
 return animalsList.size();  
}

}

1. Main Class implements adapter OnClickListener

public class RecyclerViewActivity extends AppCompatActivity implements imgRecyclerAdapter.OnItemClickListener{

1. Create adapter object and LayoutManager and set them to RecyclerView

recyclerViewAdapter adpt = new recyclerViewAdapter(animalList, this);  
GridLayoutManager gm = new GridLayoutManager(this,2);  
binding.RecyclerView.setAdapter(adpt);  
binding.RecyclerView.setLayoutManager(gm);

**Format Patterns**

1. **Date formatter**

//d - date fields, or or more digits (e.g. 8,18,31)  
//MMM - three letter code for the mont(e.g JAN,MAY,JUN)  
//yyyy- 4 digit year (e.g., 01,11,10,12)  
//dd - two digit dates(e.g., 01,08,31,18)  
//MM - 2 digit number  
//yy two digit year

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("d-MMM-yyyy");

LocalDate dob = LocalDate.parse(dogDobSTR,formatter);

1. **String Format**

**Preference/Cach**

1. Create Strings in String.xml (Must Have name and content)

<string name="txtCacheName">CacheName</string>

<string name="txtCacheDob">CacheDob</string>

1. Create Overriden method OnPause(Generate-> Override->Search for onPause)  
     
    @Override

protected void onPause() {

SharedPreferences preferences= this.getPreferences(Context.MODE\_PRIVATE);

SharedPreferences.Editor editor = preferences.edit();

name = binding.editTextName.getText().toString();

dob = binding.editTextDob.getText().toString();

editor.putString(getString(R.string.txtCacheName),name);

editor.putString(getString(R.string.txtCacheDob),dob);

editor.apply();

super.onPause();

}

1. Create CallPreferences to run on OnCreate

public void setPreferences() {

SharedPreferences preferences = this.getPreferences(Context.MODE\_PRIVATE);

name = preferences.getString(getString(R.string.txtCacheName),"");

dob = preferences.getString(getString(R.string.txtCacheDob),"");

binding.editTextDob.setText(dob);

binding.editTextName.setText(name);

}

**OTHER DATABASE CODES**

1. **Tuples Class**

public class StudGradeTuple {

@ColumnInfo(name = "studentname")

@NonNull

String StudName;

@ColumnInfo(name="studgrade")

@NonNull

Double StudGrade;

Generate -> Getter and Setters

}

1. Tuple DAO

@Dao

public interface StudGradeDao {

@Query(

"SELECT studentname, studgrade " +

"FROM students as s " +

"INNER JOIN grades as g " +

"ON s.studentid = g.studentid"

)

List<StudGradeTuple> GetStudentNameAndGrades();

@Query(

"SELECT studentname, studgrade " +

"FROM students as s " +

"INNER JOIN grades as g " +

"ON s.studentid = g.studentid " +

"WHERE studgrade>:gradeThreshold "

)

List<StudGradeTuple> GetStudentNamesAndGradesWithHIghScore(double gradeThreshold);

}