Android Exam Cheat Sheet & Study Guide (DMA)

Acest ghid conține **doar codul esențial** și pașii simpli pentru a implementa funcționalitățile comune la examen.

---

1. Networking (Preluare date de pe Net)

Pasul 1: Ai nevoie de clasa `HttpManager`

Unde: Folderul `networking/` -> `HttpManager.java` **Rol:** Se conectează la link și returnează un String (JSON-ul crud).

```
// Doar metoda process() conteaza, restul e boilerlplate
public String process() {
    HttpURLConnection connection = null;
    BufferedReader reader = null;
    StringBuilder result = new StringBuilder();
    try {
        URL url = new URL(urlAddr); // urlAddr vine din constructor
        connection = (HttpURLConnection) url.openConnection();
        connection.connect(); // Deschide conexiunea

        InputStream is = connection.getInputStream();
        reader = new BufferedReader(new InputStreamReader(is));

        String line;
        while ((line = reader.readLine()) != null) {
            result.append(line); // Citim linie cu linie
        }
    } catch (IOException e) { e.printStackTrace(); }
    finally {
        // Inchide tot
        if (connection != null) connection.disconnect();
        if (reader != null) try { reader.close(); } catch (IOException e) {}
    }
    return result.toString();
}
```

Pasul 2: Apelarea în `MainActivity` (Background Thread)

Unde: `MainActivity.java` **Important:** Rețeaua NU se face pe thread-ul principal (UI).

```
// Folosim ExecutorService pt background și Handler pt UI
ExecutorService executor = Executors.newSingleThreadExecutor();
Handler handler = new Handler(Looper.getMainLooper());

executor.execute(() -> {
    // 1. NE CONECTAM (Background)
    HttpManager manager = new HttpManager("https://jsonkeeper.com/b/...");
    String jsonResult = manager.process();
```

```
    // 2. PARSAM DATELE (tot in Background e ok)
    List<Movie> listaNoua = parseMovies(jsonResult);

    handler.post(() -> {
        // 3. ACTUALIZAM UI-ul (Main Thread)
        movieList.clear();
        movieList.addAll(listaNoua);
        adapter.notifyDataSetChanged();
    });
});
```

---

2. Parsing JSON (Transformare Text -> Obiecte)

Unde: `MainActivity.java` (metodă privată e.g. `parseJson`)

```
private List<Movie> parseMovies(String jsonStr) {
    List<Movie> list = new ArrayList<>();
    try {
        // Daca incepe cu paranteza patrata [ este JSONArray
        JSONArray array = new JSONArray(jsonStr);

        for (int i = 0; i < array.length(); i++) {
            JSONObject obj = array.getJSONObject(i);

            // Extragere câmpuri (numele trebuie sa fie identice cu cele din JSON)
            String titlu = obj.getString("title");
            double buget = obj.getDouble("budget");

            // Creare obiect
            Movie m = new Movie(titlu, buget);
            list.add(m);
        }
    } catch (JSONException e) { e.printStackTrace(); }
    return list;
}
```

---

3. Room Database (Salvare date local)

Pasul 1: Entitatea (Tabela)

Unde: `model/Movie.java` **Cod:** Adaugă adnotările `@Entity` și `@PrimaryKey`.

```
@Entity(tableName = "movies")
public class Movie implements Parcelable {
    @PrimaryKey(autoGenerate = true)
    private int id; // camp extra pt baza de date

    @ColumnInfo(name = "movie_title") // optional, daca vrei alt nume la coloana
```

```
    private String title;

    // ... restul campurilor, getteri si setteri
}
```

Pasul 2: DAO (Interfața de acces)

Unde: `database/MovieDao.java` **Cod:** Definește operațiile.

```
@Dao
public interface MovieDao {
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void insert(Movie movie);

    @Query("SELECT * FROM movies")
    List<Movie> getAll();

    @Delete
    void delete(Movie movie);
}
```

Pasul 3: Database Manager (Singleton)

Unde: `database/DatabaseManager.java`

```
@Database(entities = {Movie.class}, version = 1, exportSchema = false)
public abstract class DatabaseManager extends RoomDatabase {
    // 1. Metoda abstracta pt DAO
    public abstract MovieDao getMovieDao();

    // 2. Singleton boilerplate
    private static DatabaseManager instance;
    public static synchronized DatabaseManager getInstance(Context context) {
        if (instance == null) {
            instance = Room.databaseBuilder(context, DatabaseManager.class,
"movie_db")
                    .fallbackToDestructiveMigration() // sterge baza daca schimbi
versiunea
                    .allowMainThreadQueries() // DOAR LA EXAMEN (pt simplitate)
                    .build();
        }
        return instance;
    }
}
```

---

4. SharedPreferences (Setări simple)

Exemplu: Salvarea numelui utilizatorului sau „last opened". **Unde:** `MainActivity.java` (de obicei in `onCreate` sau la un click)

```java
// SALVARE
SharedPreferences pref = getSharedPreferences("MyPrefs", MODE_PRIVATE);
SharedPreferences.Editor editor = pref.edit();
editor.putString("username", "Andrei");
editor.putInt("app_opens", 5);
editor.apply(); // Asincron (mai rapid decat commit)

// CITIRE
SharedPreferences pref = getSharedPreferences("MyPrefs", MODE_PRIVATE);
String user = pref.getString("username", "N/A"); // "N/A" e default
int opens = pref.getInt("app_opens", 0);
```

---

5. Lucrul cu Fișiere (Text Files)

Exemplu: Buton "Export Report" care salveaza un TXT. **Unde:** `MainActivity.java`

```java
// SCRIERE (Export)
private void saveToFile(String content) {
    try {
        // MODE_PRIVATE suprascrie, MODE_APPEND adauga
        FileOutputStream fos = openFileOutput("raport.txt", MODE_PRIVATE);
        OutputStreamWriter writer = new OutputStreamWriter(fos);
        writer.write(content);
        writer.close();
        Toast.makeText(this, "Salvat!", Toast.LENGTH_SHORT).show();
    } catch (IOException e) { e.printStackTrace(); }
}

// CITIRE (Import)
private String readFromFile() {
    try {
        FileInputStream fis = openFileInput("raport.txt");
        BufferedReader reader = new BufferedReader(new InputStreamReader(fis));
        StringBuilder sb = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            sb.append(line).append("\n");
        }
        reader.close();
        return sb.toString();
    ```
} catch (IOException e) { return ""; }
}
```

---

## 6. Bonus: Sistemul de Favorite

Aceasta este o funcționalitate comună. Avem nevoie de:

1. Un câmp boolean `isFavorite` in clasa `Movie`.
```

2. Un buton (inimioară/checkbox) în fiecare element din listă pentru a marca favorit.
3. Un buton în meniu pentru a șterge tot ce **NU** este favorit.

Pasul 1: Modificare Model (`Movie.java`)

Unde: `model/Movie.java`

```
@Entity(tableName = "movies")
public class Movie implements Parcelable {
    // ... restul campurilor

    @ColumnInfo(name = "is_favorite")
    private boolean isFavorite = false; // Default nu e favorit

    public boolean isFavorite() { return isFavorite; }
    public void setFavorite(boolean favorite) { isFavorite = favorite; }
}
```

Important: Dacă ai deja aplicația instalată pe telefon, **dezinstaleaz-o** după ce modifici structura bazei de date, altfel primești eroare că s-a schimbat schema!

Pasul 2: Modificare DAO (`MovieDao.java`)

Unde: `database/MovieDao.java` Adăugăm o metodă să ștergem doar filmele care NU sunt favorite.

```
@Dao
public interface MovieDao {
    // ... metodele existente (insert, getAll)

    @Update
    void update(Movie movie); // Ca sa salvam cand dam click pe inima

    // Sterge tot unde is_favorite este 0 (false)
    @Query("DELETE FROM movies WHERE is_favorite = 0")
    void deleteNonFavorites();
}
```

Pasul 3: Modificare Layout Listă (`item_movie.xml`)

Unde: `res/layout/item_movie.xml` (sau cum se numeste layout-ul pentru `RecyclerView`) Adaugă un CheckBox sau ImageButton.

```
<CheckBox
    android:id="@+id/cbFavorite"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Favorit" />
```

Pasul 4: Modificare Adapter (`MovieAdapter.java`)

Unde: `adapters/MovieAdapter.java` Trebuie să gestionăm click-ul pe CheckBox.

```
// In clasa ViewHolder
CheckBox cbFavorite = itemView.findViewById(R.id.cbFavorite);

// In onBindViewHolder
Movie movie = movieList.get(position);

// 1. Seteaza starea initiala (fara listener ca sa nu declanseze aiurea)
cbFavorite.setOnCheckedChangeListener(null);
cbFavorite.setChecked(movie.isFavorite());

// 2. Asculta modificarile
cbFavorite.setOnCheckedChangeListener((buttonView, isChecked) -> {
    movie.setFavorite(isChecked);

    // Optional: Salveaza direct in baza de date daca ai acces la DAO aici
    // Sau notifica Activity-ul sa salveze
    // Exemplu simplu (daca ai setat DB sa permita main thread queries):
    DatabaseManager.getInstance(context).getMovieDao().update(movie);
});
```

Pasul 5: Buton Meniu "Delete except Favorites"

Unde: `res/menu/main_activity_menu.xml`

```
<item
    android:id="@+id/action_delete_non_fav"
    android:title="Delete Non-Favorites"
    app:showAsAction="never" />
```

Pasul 6: Logica in MainActivity

Unde: `MainActivity.java`

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    if (item.getItemId() == R.id.action_delete_non_fav) {

        // 1. Sterge din baza de date
        databaseManager.getMovieDao().deleteNonFavorites();

        // 2. Actualizeaza lista din memorie (sau reciteste din DB)
        // Varianta in care recitim tot din DB e cea mai sigura:
        List<Movie> remainingMovies = databaseManager.getMovieDao().getAllMovies();
        movieList.clear();
        movieList.addAll(remainingMovies);
        movieAdapter.notifyDataSetChanged();

        Toast.makeText(this, "Non-favorites deleted!", Toast.LENGTH_SHORT).show();
        return true;
    }
```

```
    return super.onOptionsItemSelected(item);
}
```