MAD Code Snippets

OnClick Event Listener (shorter version)

```
btn.setOnClickListener(v → {
        /* Code */
});
```

Getter and Setter Functions

```
public String getName() {
        return name;
}

public void setName(String name) {
        this.name = name;
}
```

Gradient

```
<gradient android:angle="180" android:startColor="#FFFFFF"
android:endColor="#000000" />
```

Toast

```
Toast.makeText(getApplicationContext(), "Hello",
Toast.LENGTH_SHORT).show();
```

WebView

```
WebView web = findViewById(R.id.webView);
web.loadUrl("file://path");
```

WebSettings

```
WebSettings webSettings = webView.getSettings();
webSettings.setJavaScriptEnabled(true);
```

Explicit Intent (Navigating to a different screen)

```
Intent intent = new Intent(MainActivity.this, SecondActivity.class);
startActivity(intent);
```

We can also use `getApplicationContext()` instead of `MainActivity.this`.

Message Passing (using Extras)

```
Intent intent = new Intent(MainActivity.this, SecondActivity.class);
 intent.putExtra("email", email.getText.toString());
 intent.putExtra("message", message.getText.toString());
startActivity(intent);
```

Implicit Intent (Sending Email in Email App)

```
Intent intent = new Intent(Intent.ACTION_SEND);
 intent.putExtra(Intent.EXTRA_EMAIL, new String[]{"sender's email"});
 intent.putExtra(Intent.EXTRA_SUBJECT, "Subject");
 intent.putExtra(Intent.EXTRA_TEXT, "Email message text");
startActivity(Intent.createChooser(intent, "Send Email"));
```

Implicit Intent (Opening Webpage in Browser)

```
Intent intent = new Intent(Intent.ACTION_VIEW);
 intent.setData(Uri.parse("https://www.example.com"));
startActivity(intent);
```

Recycler View

```
rv = view.findViewById(R.id.rv);
 rv.setLayoutManager(new LinearLayoutManager(getContext()));
notesAdapter = new NotesAdapter(notes);
 rv.setAdapter(notesAdapter);
```

Navigation Drawer `activity_main.xml`

```
<DrawerLayout>
    <include layout="@layout/content_layout"/>
    <NavigationView
        app:headerLayout="@layout/header_layout"
        app:menu="@menu/nav_menu"/>
</DrawerLayout>
```

Opening the Navigation Drawer

```
menuButton.setOnClickListener(v→
drawerLayout.openDrawer(GravityCompat.START));
```

Navigation Handling

```
navigationView.setNavigationItemSelectedListener(item → {
        switch (item.getItemId()) {
                case R.id.nav_home:
                        switchFragment(new HomeFragment()); break;
                case R.id.nav_profile:
                        switchFragment(new ProfileFragment()); break;
        }
        drawerLayout.closeDrawer(GravityCompat.START);
        return true;
});
```

Fragment Replacement/Switching

```
private void switchFragment(Fragment fragment) {
getSupportFragmentManager()
        .beginTransaction()
        .replace(R.id.fragment_container, fragment)
        .commit();
}
```

Back-Button (Closing the Drawer)

```
public void onBackPressed() {
if (drawerLayout.isDrawerOpen(GravityCompat.START)) {
        drawerLayout.closeDrawer(GravityCompat.START);
} else { super.onBackPressed(); }
```

SQLite DBHelper

```
public class DBHelper extends SQLiteOpenHelper {
        /* Code */
};
```

SQLite Constructor (for DBHelper)

```
public DBHelper(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
}
```

SQLite Creating a Database

```
public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE " + TABLE_NAME + " ("
         + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
         + COLUMN_TITLE + " TEXT, "
         + COLUMN_DESCRIPTION + " TEXT)");
}
```

SQLite Upgrading a Database

```
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
}
```

SQLite Create Method

```
public boolean insert_data(String title, String description) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(COLUMN_TITLE, title);
        values.put(COLUMN_DESCRIPTION, description);
        return db.insert(TABLE_NAME, null, values) =/= -1;
}
```

SQLite Read Method

```
public Cursor getData() {
        SQLiteDatabase db = this.getReadableDatabase();
        return db.rawQuery("SELECT * FROM " + TABLE_NAME, null);
}
```

SQLite Update Method

```
public boolean updateData(int id, String title, String description) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(COLUMN_TITLE, title);
        contentValues.put(COLUMN_DESCRIPTION, description);
    int result = db.update(TABLE_NAME, values, COLUMN_ID + " = ?",
String.valueOf(id));
    return result > 0; // Returns true if the update was successful
}
```

SQLite Delete Method

```
public boolean deleteData(int id) {
        SQLiteDatabase db = this.getWritableDatabase();
        int result = db.delete(TABLE_NAME, COLUMN_ID + " = ?",
String.valueOf(id));
        return result > 0; // Returns true if the deletion was successful
}
```

Firebase Email Sanitization

```
String sanitizedEmail = email.getText().toString().replace(".", "_");
DatabaseReference ref = users_ref.child(sanitizedEmail);
```

Firebase User Creation (For Signup)

```
auth.createUserWithEmailAndPassword(email.getText().toString(),
password.getText().toString()).addOnCompleteListener(task -> {
        if(task.isSuccessful()){
                ref.setValue(signupData).addOnCompleteListener(dbtask -> {
                        /* If-Else and Toasts */
                });
        } else { /* Toast to show exception */ };
});
```

Firebase User Authentication (For Login)

```
auth.signInWithEmailAndPassword(email.getText().toString(),
password.getText().toString()).addOnCompleteListener(task -> {
        if(task.isSuccessful()){
                ref.setValue(signupData).addOnCompleteListener(dbtask -> {
                        /* If-Else and Toasts */
                });
        } else { /* Toast to show exception */ };
});
```

Firebase Data Fetching → use `addListenerForSingleValueEvent` method with a `ValueEventListener()` method and use `onDataChange()` and `onCancelled()` methods to check for success and failure.

One-Liners

LinearLayout **Vertical Orientation**
`android:orientation="vertical"`

Centering Content (in XML)
`android:gravity="center"`

Getting XML Views into Java (using classes)
`TextView text_view = findViewById(R.id.textView);`

Setting Text (of TextView)
`text_view.setText("Hello World!");

Getting Text (of TextView)

Fragment Manager
`FragmentManager fragmentManager = getSupportFragmentManager();`

Firebase Realtime Database Initialization
`FirebaseDatabase database = FirebaseDatabase.getInstance();`

Firebase Authentication Initialization
`FirebaseAuth auth = FirebaseAuth.getInstance();`

Firebase Database Parent-Node Creation (named "users")
`DatabaseReference myRef = database.getReference("users");`