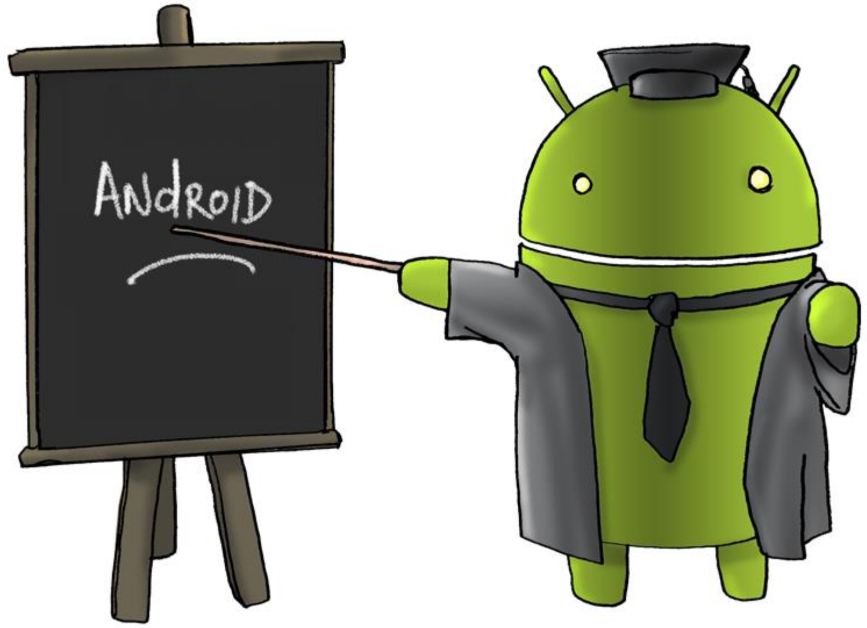
Mobile Application Development

BSCS-7

**Lecture #
18, 19**



Storage Options

- Android provides several options to save persistent application data depends on your specific needs, such as whether data should be private to your application or accessible to other applications (and user) and how much space your data requires.
- Android provides a way to expose even your private data to other applications — with a content provider. A content provider is an optional component that exposes read/write access to your application data, subject to whatever restrictions you want to impose.

1. Shared Preferences

- Store private primitive data in key-value pairs.
- The SharedPreferences class provides a general framework that allows you to save and retrieve persistent key-value pairs of primitive data types. You can use SharedPreferences to save any primitive data: booleans, floats, ints, longs, and strings. This data will persist across user sessions (even if your application is killed).

2. Internal Storage

- Store private data on the device memory.
- You can save files directly on the device's internal storage. By default, files saved to the internal storage are private to your application and other applications cannot access them (nor can the user). When the user uninstalls your application, these files are removed.
- **Saving cache files**
  - These are temporary files which may be deleted by system when internal storage is low.

Storage Options

- These are removed when application is uninstalled.
- However, you should not rely on the system to clean up these files for you. You should always maintain the cache files yourself and stay within a reasonable limit of space consumed, such as 1MB. When the user uninstalls your application, these files are removed.

3. External Storage

- Store public data on the shared external storage.
- Every Android-compatible device supports a shared "external storage" that you can use to save files. This can be a removable storage media (such as an SD card) or an internal (non-removable) storage. Files saved to the external storage are world-readable and can be modified by the user when they enable USB mass storage to transfer files on a computer.
- **CAUTION!** External storage can become unavailable without warning, if it is removed.

4. SQLite Databases

- Store structured data in a private database.

5. Network Connection

- Store data on the web with your own network server.
- To do network operations, use classes in the following packages:
  - java.net.*
  - android.net.*

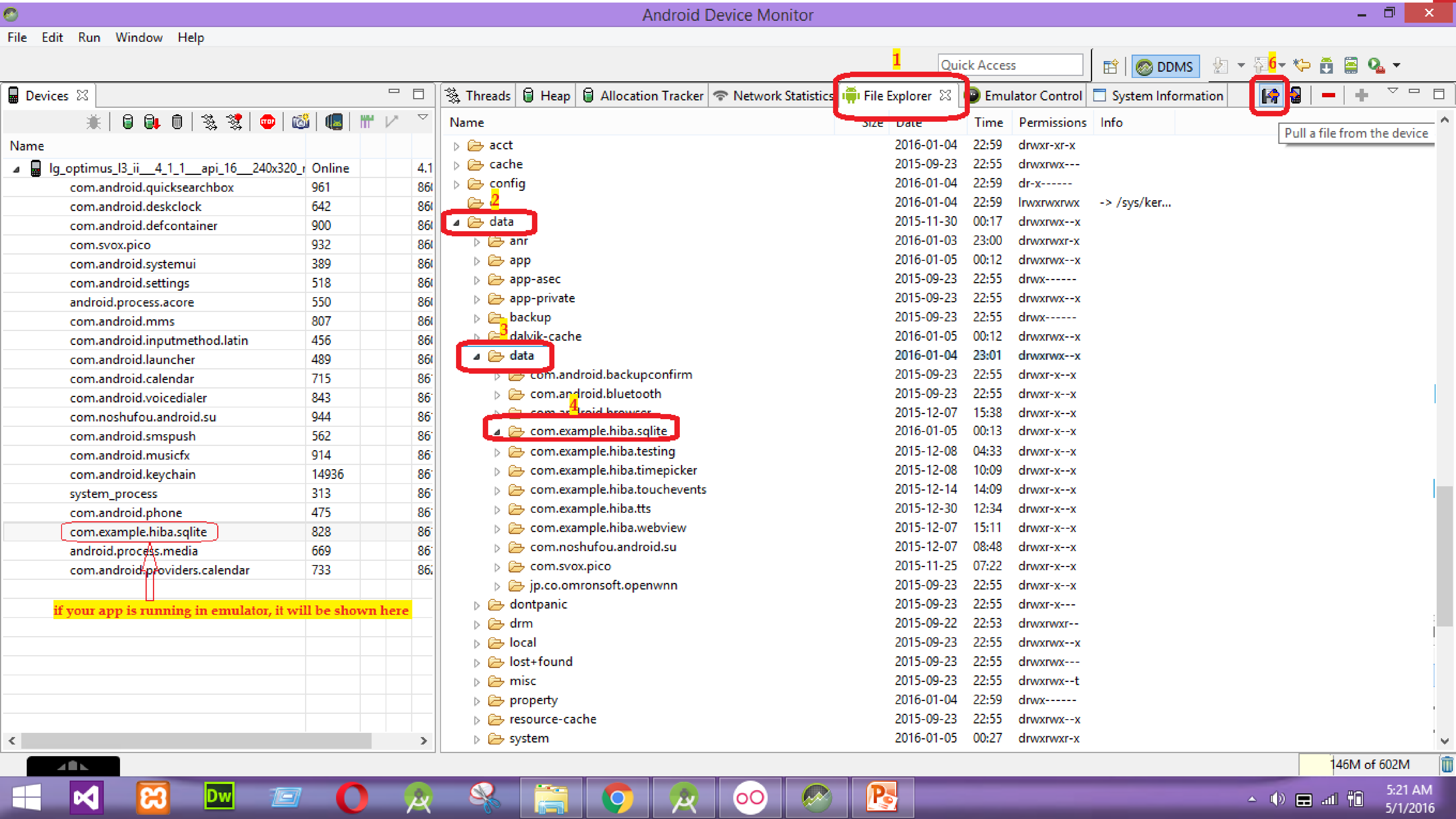
DataBase Management using SQLite

- SQLite is an open-source SQL database that stores data to a text file on a device.
- SQLite is a software library that implements a self-contained, server-less, zero-configuration, transactional SQL database engine. SQLite is the most widely deployed SQL database engine in the world. It was designed in year 2000.
- SQLite supports all the relational database features. In order to access this database, you don't need to establish any kind of connections for it like JDBC,ODBC etc.
- SQLite transactions are fully ACID-compliant.
- ACID(Atomicity, Consistency, Isolation, Durability)
- SQLite is case insensitive.

DataBase Management using SQLite

Examining the Database Files

- Databases are stored in the **/data/data/<package-name>/databases** directory.
- Run your app in emulator and click on **Tools > Android > Android Device Monitor** from Android Studio.
- All steps are given in next slide.
- This database is saved on mobile. If you want to see its contents in your local PC, you can export it by clicking the button **Pull a file from the Device.** Save it wherever you want.
- If you have SQLite installed on your computer, you can use its terminal to view this database.
- Another way is to use FireFox plugin. Open FireFox, go to settings. Click on AddOns. Search for AddOns called **SQLite.**
- If you don’t see it, go to following link;
- **https://addons.mozilla.org/en-US/firefox/addon/sqlite-manager/**
- Install the **SQLite Manager** from there. Restart your browser now.
- Go to Settings, and click on Customize. Drag and drop SQLite Manager in Tools to quickly view it every time.
- Now you can Browse you database. Be sure to select All Files before browsing otherwise your database will not be shown.


Android Device Monitor

File Edit Run Window Help

Quick Access | DDMS

Devices | Threads | Heap | Allocation Tracker | Network Statistics | File Explorer | Emulator Control | System Information

1

6

Pull a file from the device

Name		
lg_optimus_l3_ii__4_1_1__api_16__240x320_r	Online	4.1
com.android.quicksearchbox	961	86
com.android.deskclock	642	86
com.android.defcontainer	900	86
com.svox.pico	932	86
com.android.systemui	389	86
com.android.settings	518	86
android.process.acore	550	86
com.android.mms	807	86
com.android.inputmethod.latin	456	86
com.android.launcher	489	86
com.android.calendar	715	86
com.android.voicedialer	843	86
com.noshufou.android.su	944	86
com.android.smspush	562	86
com.android.musicfx	914	86
com.android.keychain	14936	86
system_process	313	86
com.android.phone	475	86
com.example.hiba.sqlite	828	86
android.process.media	669	86
com.android.providers.calendar	733	86

if your app is running in emulator, it will be shown here

Name	Size	Date	Time	Permissions	Info
acct		2016-01-04	22:59	drwxr-xr-x	
cache		2015-09-23	22:55	drwxrwx---	
config		2016-01-04	22:59	dr-x------	
2		2016-01-04	22:59	lrwxrwxrwx	-> /sys/ker...
data		2015-11-30	00:17	drwxrwx--x	
anr		2016-01-03	23:00	drwxrwxr-x	
app		2016-01-05	00:12	drwxrwx--x	
app-asec		2015-09-23	22:55	drwx------	
app-private		2015-09-23	22:55	drwxrwx--x	
backup		2015-09-23	22:55	drwx------	
3 dalvik-cache		2016-01-05	00:12	drwxrwx--x	
data		2016-01-04	23:01	drwxrwx--x	
com.android.backupconfirm		2015-09-23	22:55	drwxr-x--x	
com.android.bluetooth		2015-09-23	22:55	drwxr-x--x	
4 com.android.browser		2015-12-07	15:38	drwxr-x--x	
com.example.hiba.sqlite		2016-01-05	00:13	drwxr-x--x	
com.example.hiba.testing		2015-12-08	04:33	drwxr-x--x	
com.example.hiba.timepicker		2015-12-08	10:09	drwxr-x--x	
com.example.hiba.touchevents		2015-12-14	14:09	drwxr-x--x	
com.example.hiba.tts		2015-12-30	12:34	drwxr-x--x	
com.example.hiba.webview		2015-12-07	15:11	drwxr-x--x	
com.noshufou.android.su		2015-12-07	08:48	drwxr-x--x	
com.svox.pico		2015-11-25	07:22	drwxr-x--x	
jp.co.omronsoft.openwnn		2015-09-23	22:55	drwxr-x--x	
dontpanic		2015-09-23	22:55	drwxr-x---	
drm		2015-09-22	22:53	drwxrwxr--	
local		2015-09-23	22:55	drwxrwx--x	
lost+found		2015-09-23	22:55	drwxrwx---	
misc		2015-09-23	22:55	drwxrwx--t	
property		2016-01-04	22:59	drwx------	
resource-cache		2015-09-23	22:55	drwxrwx--x	
system		2016-01-05	00:27	drwxrwxr-x	

146M of 602M

5:21 AM
5/1/2016

SQLiteOpenHelper

android.database.sqlite.SQLiteOpenHelper

- It is a helper class to manage database creation and version management.

Public Constructors	
SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version)	Create a helper object to create, open, and/or manage a database.
SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version, DatabaseErrorHandler errorHandler)	Create a helper object to create, open, and/or manage a database.

Public Methods		
synchronized void	close()	Close any open database object.
String	getDatabaseName()	Return the name of the SQLite database being opened, as given to the constructor.
SQLiteDatabase	getReadableDatabase()	Create and/or open a database.
SQLiteDatabase	getWritableDatabase()	Create and/or open a database that will be used for reading and writing.
abstract void	onCreate(SQLiteDatabase db)	Called when the database is created for the first time.
void	onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion)	Called when the database needs to be downgraded.
void	onOpen(SQLiteDatabase db)	Called when the database has been opened.
abstract void	onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)	Called when the database needs to be upgraded.

SQLiteDatabase

android.database.sqlite.SQLiteDatabase

- Database names must be unique within an application, not across all applications.

Public Methods		
static SQLiteDatabase	create(SQLiteDatabase.CursorFactory factory)	Create a memory backed SQLite database.
int	delete(String table, String whereClause, String[] whereArgs)	Convenience method for deleting rows in the database.
static boolean	deleteDatabase(File file)	Deletes a database including its journal file and other auxiliary files that may have been created by the database engine.
void	execSQL(String sql)	Execute a single SQL statement that is NOT a SELECT or any other SQL statement that returns data.
void	execSQL(String sql, Object[] bindArgs)	Execute a single SQL statement that is NOT a SELECT/INSERT/UPDATE/DELETE.
long	getMaximumSize()	Returns the maximum size the database may grow to.
final String	getPath()	Gets the path to the database file.
int	getVersion()	Gets the database version.
long	insert(String table, String nullColumnHack, ContentValues values)	Convenience method for inserting a row into the database.
boolean	isOpen()	Returns true if the database is currently open.
boolean	isReadOnly()	Returns true if the database is opened as read only.
static SQLiteDatabase	openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags, DatabaseErrorHandler errorHandler)	Open the database according to the flags OPEN_READWRITE OPEN_READONLY CREATE_IF_NECESSARY and/or NO_LOCALIZED_COLLATORS.

SQLiteDatabase

android.database.sqlite.SQLiteDatabase

Public Methods		
static SQLiteDatabase	openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags)	Open the database according to the flags OPEN_READWRITE OPEN_READONLY CREATE_IF_NECESSARY and/or NO_LOCALIZED_COLLATORS.
static SQLiteDatabase	openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory, DatabaseErrorHandler errorHandler)	Equivalent to openDatabase(path, factory, CREATE_IF_NECESSARY, errorHandler).
static SQLiteDatabase	openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory)	Equivalent to openDatabase(path, factory, CREATE_IF_NECESSARY).
static SQLiteDatabase	openOrCreateDatabase(File file, SQLiteDatabase.CursorFactory factory)	Equivalent to openDatabase(file.getPath(), factory, CREATE_IF_NECESSARY).
Cursor	query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit)	Query the given table, returning a Cursor over the result set.
Cursor	rawQuery(String sql, String[] selectionArgs, CancellationSignal cancellationSignal)	Runs the provided SQL and returns a Cursor over the result set.
Cursor	rawQuery(String sql, String[] selectionArgs)	Runs the provided SQL and returns a Cursor over the result set.
long	setMaximumSize(long numBytes)	Sets the maximum size the database will grow to.
void	setVersion(int version)	Sets the database version.
String	toString()	Returns a string containing a concise, human-readable description of this object.
int	update(String table, ContentValues values, String whereClause, String[] whereArgs)	Convenience method for updating rows in the database.

ContentValues

android.content.ContentValues

- This class is used to store a set of values.

Public Constructors	
ContentValues()	Creates an empty set of values using the default initial size
ContentValues(int size)	Creates an empty set of values using the given initial size
ContentValues(ContentValues from)	Creates a set of values copied from the given set

Public Methods		
void	clear()	Removes all values.
boolean	containsKey(String key)	Returns true if this object has the named value.
boolean	equals(Object object)	Compares this instance with the specified object and indicates if they are equal.
Object	get(String key)	Gets a value.
Boolean	getAsBoolean(String key)	Gets a value and converts it to a Boolean.
Byte	getAsByte(String key)	Gets a value and converts it to a Byte.
byte[]	getAsByteArray(String key)	Gets a value that is a byte array.
Double	getAsDouble(String key)	Gets a value and converts it to a Double.
Float	getAsFloat(String key)	Gets a value and converts it to a Float.
Integer	getAsInteger(String key)	Gets a value and converts it to an Integer.
Long	getAsLong(String key)	Gets a value and converts it to a Long.
Short	getAsShort(String key)	Gets a value and converts it to a Short.
String	getAsString(String key)	Gets a value and converts it to a String.
void	put(String key, Byte value)	Adds a value to the set.
void	put(String key, Integer value)	Adds a value to the set.

ContentValues

Public Methods		
void	put(String key, Float value)	Adds a value to the set.
void	put(String key, Short value)	Adds a value to the set.
void	put(String key, byte[] value)	Adds a value to the set.
void	put(String key, String value)	Adds a value to the set.
void	put(String key, Double value)	Adds a value to the set.
void	put(String key, Long value)	Adds a value to the set.
void	put(String key, Boolean value)	Adds a value to the set.
void	putAll(ContentValues other)	Adds all values from the passed in ContentValues.
void	putNull(String key)	Adds a null value to the set.
void	remove(String key)	Remove a single value.
int	size()	Returns the number of values.
String	toString()	Returns a string containing a concise, human-readable description of this object.

Cursor

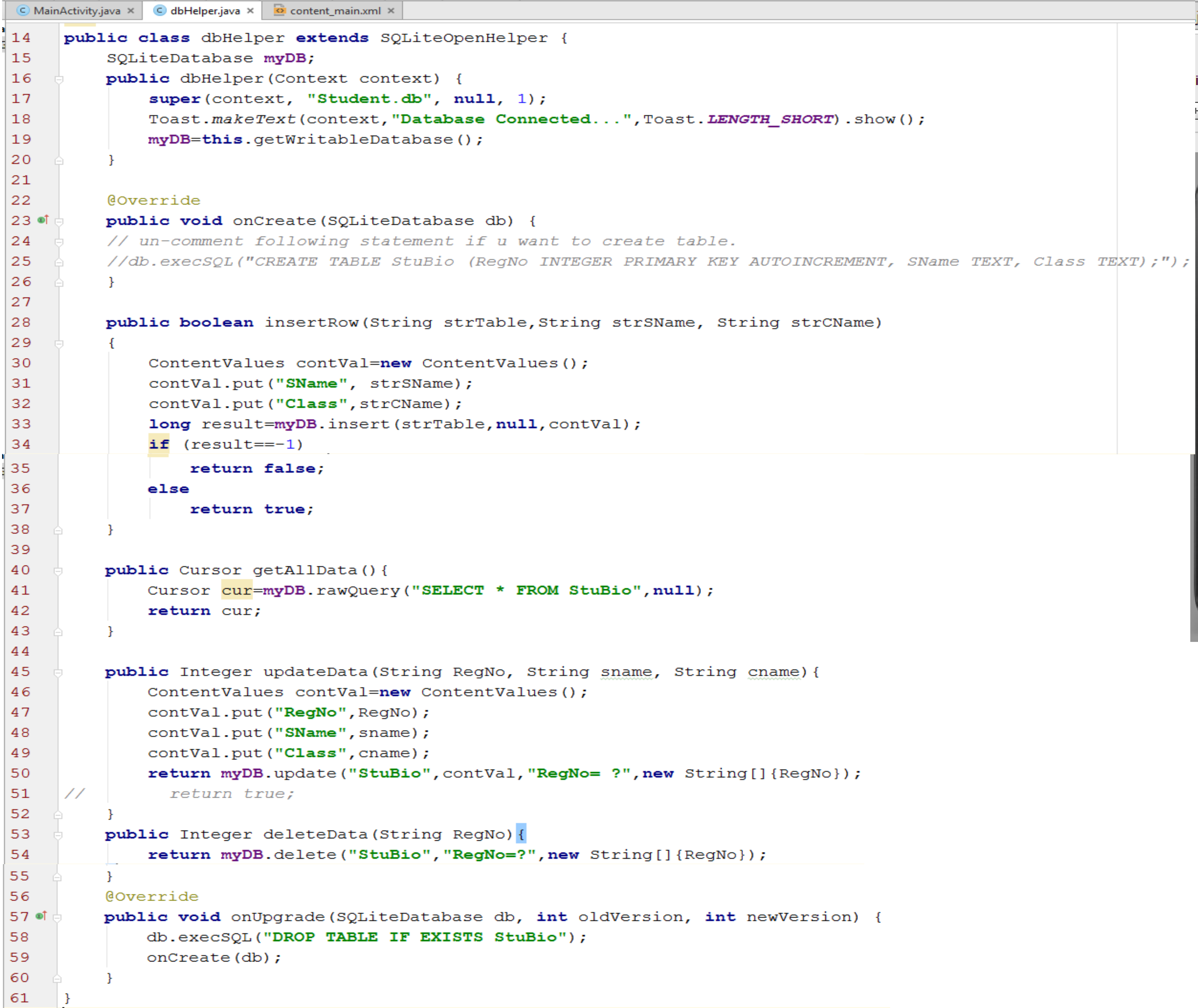
android.database.Cursor

- This interface provides random read-write access to the result set returned by a database query.

Public Methods		
abstract void	close()	Closes the Cursor, releasing all of its resources and making it completely invalid.
abstract void	copyStringToBuffer(int columnIndex, CharArrayBuffer buffer)	Retrieves the requested column text and stores it in the buffer provided.
abstract int	getColumnCount()	Return total number of columns
abstract int	getColumnIndex(String columnName)	Returns the zero-based index for the given column name, or -1 if the column doesn't exist.
abstract int	getColumnIndexOrThrow(String columnName)	Returns the zero-based index for the given column name, or throws IllegalArgumentException if the column doesn't exist.
abstract String	getColumnName(int columnIndex)	Returns the column name at the given zero-based column index.
abstract String[]	getColumnNames()	Returns a string array holding the names of all of the columns in the result set in the order in which they were listed in the result.
abstract int	getCount()	Returns the numbers of rows in the cursor.
abstract double	getDouble(int columnIndex)	Returns the value of the requested column as a double.
abstract Bundle	getExtras()	Returns a bundle of extra values.
abstract float	getFloat(int columnIndex)	Returns the value of the requested column as a float.
abstract int	getInt(int columnIndex)	Returns the value of the requested column as an int.
abstract long	getLong(int columnIndex)	Returns the value of the requested column as a long.
abstract int	getPosition()	Returns the current position of the cursor in the row set.
abstract short	getShort(int columnIndex)	Returns the value of the requested column as a short.
abstract String	getString(int columnIndex)	Returns the value of the requested column as a String.

Cursor

Public Methods		
abstract int	getType(int columnIndex)	Returns data type of the given column's value.
abstract boolean	isAfterLast()	Returns whether the cursor is pointing to the position after the last row.
abstract boolean	isBeforeFirst()	Returns whether the cursor is pointing to the position before the first row.
abstract boolean	isClosed()	return true if the cursor is closed
abstract boolean	isFirst()	Returns whether the cursor is pointing to the first row.
abstract boolean	isLast()	Returns whether the cursor is pointing to the last row.
abstract boolean	isNull(int columnIndex)	Returns true if the value in the indicated column is null.
abstract boolean	move(int offset)	Move the cursor by a relative amount, forward or backward, from the current position.
abstract boolean	moveToFirst()	Move the cursor to the first row.
abstract boolean	moveToLast()	Move the cursor to the last row.
abstract boolean	moveToNext()	Move the cursor to the next row.
abstract boolean	moveToPosition(int position)	Move the cursor to an absolute position.
abstract boolean	moveToPrevious()	Move the cursor to the previous row.



MainActivity.java × dbHelper.java × content_main.xml ×

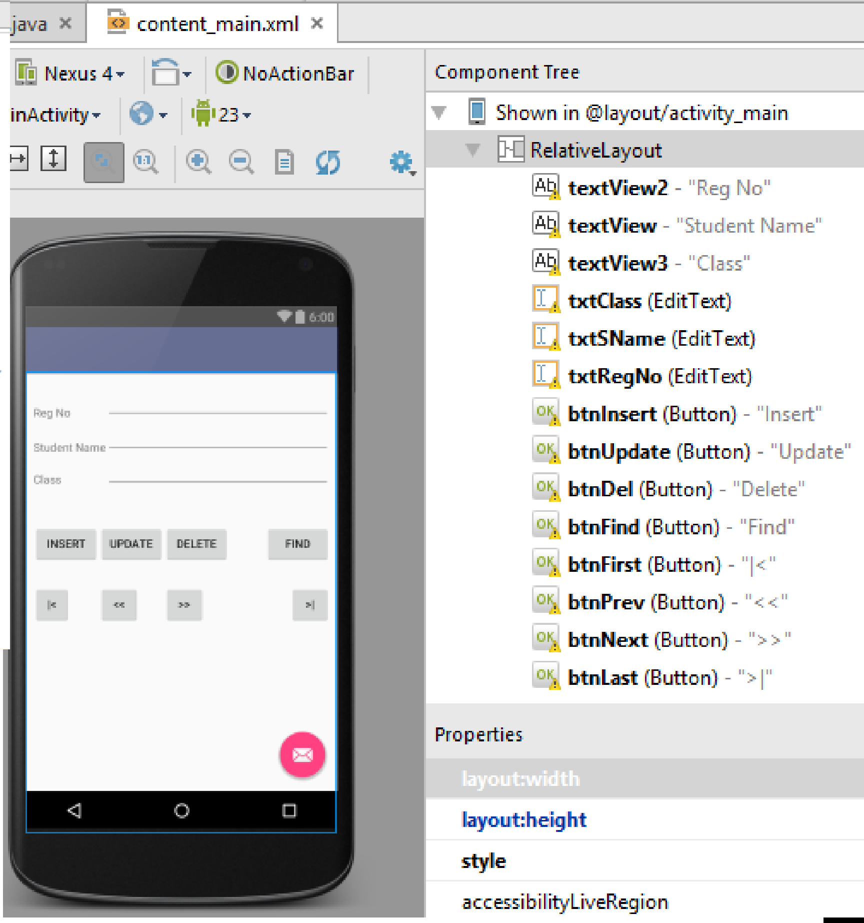
```java
public class dbHelper extends SQLiteOpenHelper {
    SQLiteDatabase myDB;
    public dbHelper(Context context) {
        super(context, "Student.db", null, 1);
        Toast.makeText(context,"Database Connected...",Toast.LENGTH_SHORT).show();
        myDB=this.getWritableDatabase();
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
    // un-comment following statement if u want to create table.
    //db.execSQL("CREATE TABLE StuBio (RegNo INTEGER PRIMARY KEY AUTOINCREMENT, SName TEXT, Class TEXT);");
    }

    public boolean insertRow(String strTable,String strSName, String strCName)
    {
        ContentValues contVal=new ContentValues();
        contVal.put("SName", strSName);
        contVal.put("Class",strCName);
        long result=myDB.insert(strTable,null,contVal);
        if (result==-1)
            return false;
        else
            return true;
    }

    public Cursor getAllData(){
        Cursor cur=myDB.rawQuery("SELECT * FROM StuBio",null);
        return cur;
    }

    public Integer updateData(String RegNo, String sname, String cname){
        ContentValues contVal=new ContentValues();
        contVal.put("RegNo",RegNo);
        contVal.put("SName",sname);
        contVal.put("Class",cname);
        return myDB.update("StuBio",contVal,"RegNo= ?",new String[]{RegNo});
//          return true;
    }
    public Integer deleteData(String RegNo){
        return myDB.delete("StuBio","RegNo=?",new String[]{RegNo});
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS StuBio");
        onCreate(db);
    }
}
```

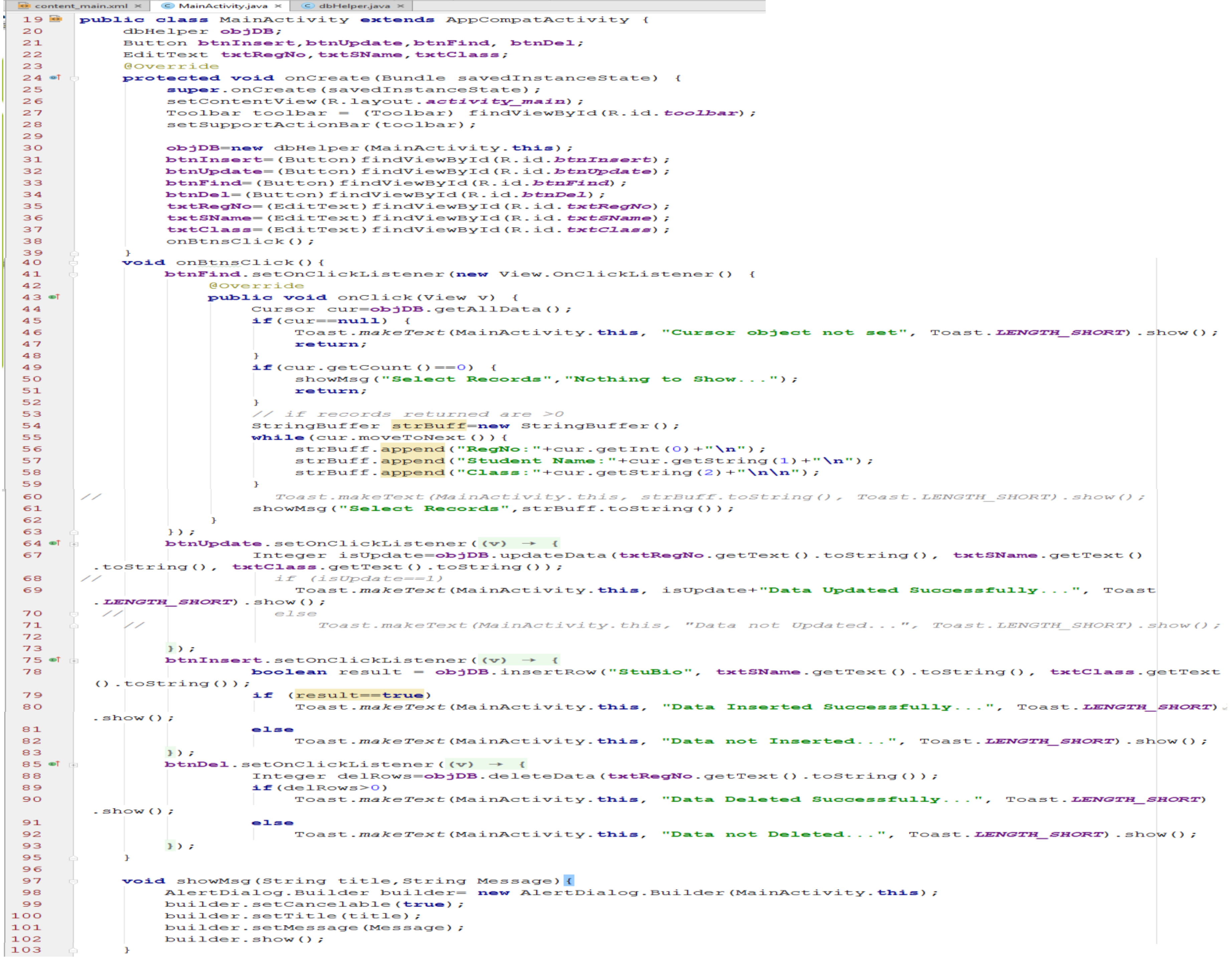


Component Tree

- Shown in @layout/activity_main
  - RelativeLayout
    - textView2 - "Reg No"
    - textView - "Student Name"
    - textView3 - "Class"
    - txtClass (EditText)
    - txtSName (EditText)
    - txtRegNo (EditText)
    - btnInsert (Button) - "Insert"
    - btnUpdate (Button) - "Update"
    - btnDel (Button) - "Delete"
    - btnFind (Button) - "Find"
    - btnFirst (Button) - "|<"
    - btnPrev (Button) - "<<"
    - btnNext (Button) - ">>"
    - btnLast (Button) - ">|"

Properties

layout:width
layout:height
style
accessibilityLiveRegion



```java
public class MainActivity extends AppCompatActivity {
    dbHelper objDB;
    Button btnInsert,btnUpdate,btnFind, btnDel;
    EditText txtRegNo,txtSName,txtClass;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        objDB=new dbHelper(MainActivity.this);
        btnInsert=(Button)findViewById(R.id.btnInsert);
        btnUpdate=(Button)findViewById(R.id.btnUpdate);
        btnFind=(Button)findViewById(R.id.btnFind);
        btnDel=(Button)findViewById(R.id.btnDel);
        txtRegNo=(EditText)findViewById(R.id.txtRegNo);
        txtSName=(EditText)findViewById(R.id.txtSName);
        txtClass=(EditText)findViewById(R.id.txtClass);
        onBtnsClick();
    }
    void onBtnsClick(){
        btnFind.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Cursor cur=objDB.getAllData();
                if(cur==null) {
                    Toast.makeText(MainActivity.this, "Cursor object not set", Toast.LENGTH_SHORT).show();
                    return;
                }
                if(cur.getCount()==0) {
                    showMsg("Select Records","Nothing to Show...");
                    return;
                }
                // if records returned are >0
                StringBuffer strBuff=new StringBuffer();
                while(cur.moveToNext()){
                    strBuff.append("RegNo:"+cur.getInt(0)+"\n");
                    strBuff.append("Student Name:"+cur.getString(1)+"\n");
                    strBuff.append("Class:"+cur.getString(2)+"\n\n");
                }
//                  Toast.makeText(MainActivity.this, strBuff.toString(), Toast.LENGTH_SHORT).show();
                showMsg("Select Records",strBuff.toString());
            }
        });
        btnUpdate.setOnClickListener((v) -> {
                Integer isUpdate=objDB.updateData(txtRegNo.getText().toString(), txtSName.getText()
.toString(), txtClass.getText().toString());
//                  if (isUpdate==1)
                    Toast.makeText(MainActivity.this, isUpdate+"Data Updated Successfully...", Toast
.LENGTH_SHORT).show();
 //                 else
   //                   Toast.makeText(MainActivity.this, "Data not Updated...", Toast.LENGTH_SHORT).show();

        });
        btnInsert.setOnClickListener((v) -> {
                boolean result = objDB.insertRow("StuBio", txtSName.getText().toString(), txtClass.getText
().toString());
                if (result==true)
                    Toast.makeText(MainActivity.this, "Data Inserted Successfully...", Toast.LENGTH_SHORT)
.show();
                else
                    Toast.makeText(MainActivity.this, "Data not Inserted...", Toast.LENGTH_SHORT).show();
        });
        btnDel.setOnClickListener((v) -> {
                Integer delRows=objDB.deleteData(txtRegNo.getText().toString());
                if(delRows>0)
                    Toast.makeText(MainActivity.this, "Data Deleted Successfully...", Toast.LENGTH_SHORT)
.show();
                else
                    Toast.makeText(MainActivity.this, "Data not Deleted...", Toast.LENGTH_SHORT).show();
        });
    }

    void showMsg(String title,String Message){
        AlertDialog.Builder builder= new AlertDialog.Builder(MainActivity.this);
        builder.setCancelable(true);
        builder.setTitle(title);
        builder.setMessage(Message);
        builder.show();
    }
```

Good Luck!