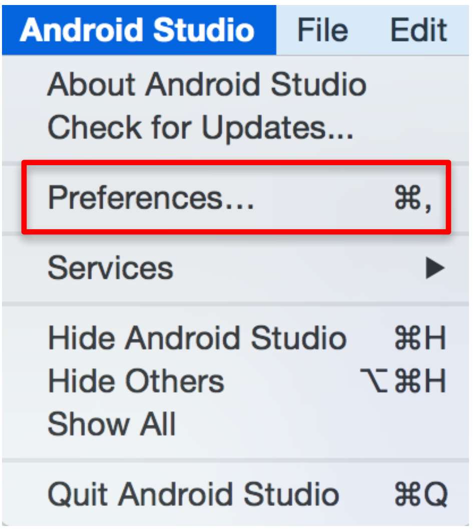
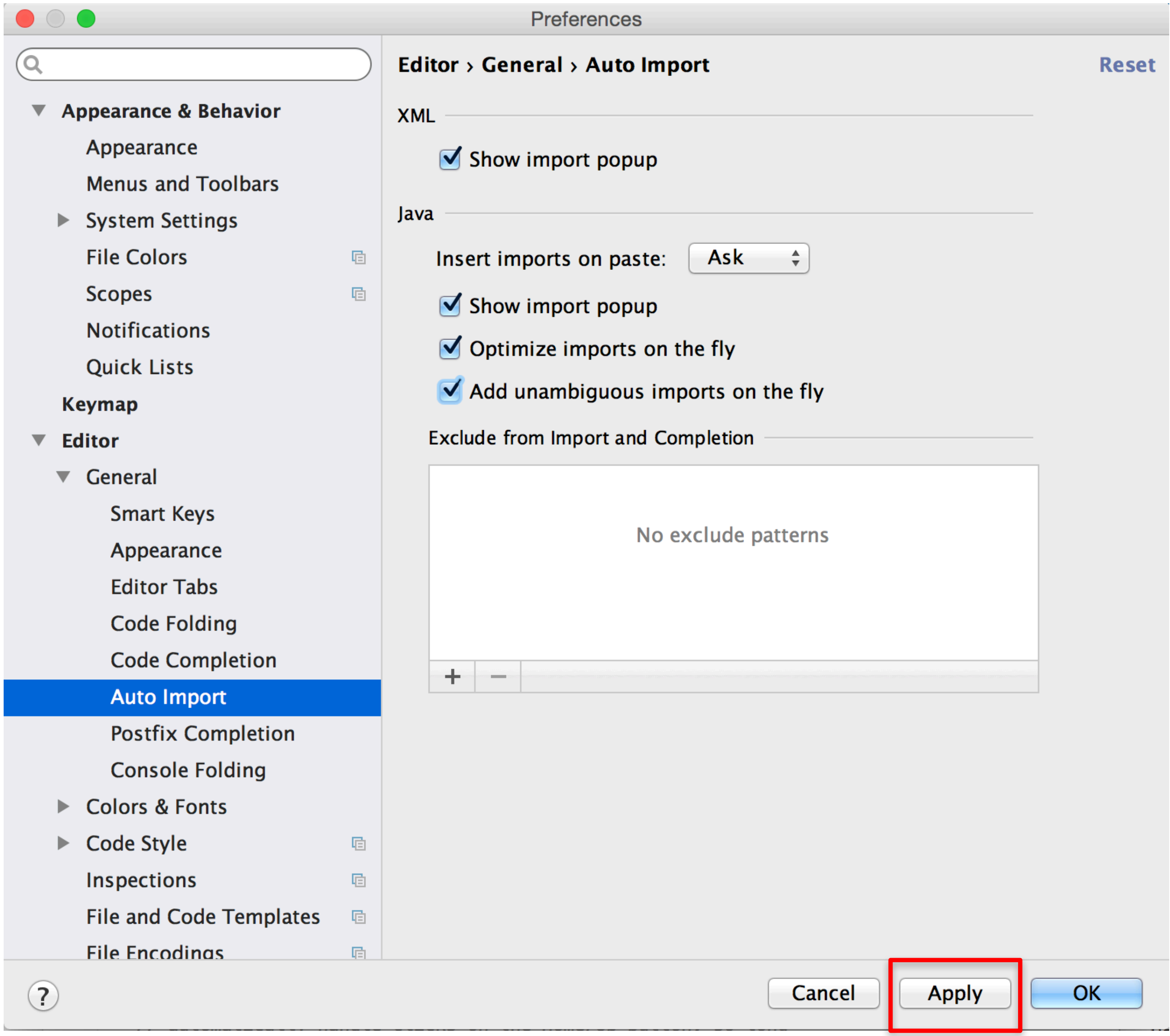
Tips: Android Studio’s Auto Import

- For Windows/Linux,

  File -> Settings -> Editor -> General -> Auto Import -> Java
  - Change “Insert imports on paste” to “All”
  - Check “Add unambiguous imports on the fly” option

- For Mac, follow the instructions on next slide





Editor › General › Auto Import

XML

- [✓] Show import popup

Java

Insert imports on paste: Ask

- [✓] Show import popup
- [✓] Optimize imports on the fly
- [✓] Add unambiguous imports on the fly

Exclude from Import and Completion

No exclude patterns

SQLite
on
Android

http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html#delete%28java.lang.String,%20java.lang.String,%20java.lang.String[]%29

SQLite

- SQLite is in every Android device

- SQLite requires very limited memory during runtime (about 250K Bytes)

- Data Types: TEXT (String), INTEGER (long), REAL (double)

- SQLite does not validate data types (eg. can write integer to a string column). All data types get converted to TEXT, INTEGER, or REAL.

- Default location for database:
  DATA/data/APP_NAME/databases/FILENAME

$\uparrow$

Environment.getDataDirectory()

Create an SQLite Helper class:

- public class MyDBHelper extends SQLiteOpenHelper { }

- The constructor of this helper class should look like

```
public MyDBHelper(Context context) {
     super(context, databaseName, null, databaseVersion);     // calls parent class
                                                                    constructor
}
```

databaseName: public static final String

databaseVersion: public static final int

Define some useful strings within the Helper class:

```
public static final int databaseVersion = 1;
public static final String databaseName = “yourDBName”;

public static final String tableName = “yourTableName”;
public static final String columnName1 = “_id”;        // should use _id as primary key
public static final String columnName2 = “name2”;
public static final String columnName3 = “name3”;

private static final String SQLite_CREATE =
“CREATE TABLE “ + tableName + “ (“ + columnName1 + “ INTEGER PRIMARY KEY
AUTOINCREMENT,” + columnName2 + “ TEXT NOT NULL,” + columnName3 + “ TEXT
NOT NULL);”;

private static final String SQLite_DELETE =
“DROP TABLE IF EXISTS “ + tableName;
```

- Override the following methods of DBHelper

```
@Override
public void onCreate(SQLiteDatabase db) {
   db.execSQL(SQLite_CREATE);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
   // note: our upgrade policy here is simply to discard the data and start all over
   db.execSQL(SQLite_DELETE);     // delete the existing database
   onCreate(db);                  // create a new database
}
```

Note: onCreate and onUpgrade are called when the database is opened, for example, by getWritableDatabase().   onCreate() is only run when the database file did not exist. onUpgrade() is only called when database file exists but the stored version number is lower than requested in constructor.

Note:

Cursor is the class that represents a 2 dimensional table of database.

When you retrieve data using a query statement, the database will create a CURSOR object and return its reference to you. This reference will point to the 0th location (before first location), so you need to move to the first record using moveToFirst().

Create the database class:

- public class MyDB

- Declare the following variables within MyDB:

```
MyDBHelper          DBHelper;
SQLiteDatabase      db;
final Context       context;
```

- The constructor of the class should look like

```
public MyDB(Context ctx) {
    this.context = ctx;
    DBHelper = new MyDBHelper(this.context);
}
```

Define the following database operations within MyDB:

- Open database

```
public MyDB open()  {
    db = DBHelper.getWritableDatabase();
    return this;
}
```

- Close database

```
public void close() {
      DBHelper.close();
}
```

- Insert a record into database (create a method to do the following):

```
ContentValues initialValues = new ContentValues();

initialValues.put(MyDBHelper.columnName2, name2_str);
initialValues.put(MyDBHelper.columnName3, name3_str);

db.insert(MyDBHelper.tableName, null, initialValues);
```

$\uparrow$

returns a type long indicating the row ID of the newly inserted row, return -1 if error occurred

- Delete a record from database (create a method to do the following)

```
db.delete(MyDBHelper.tableName,
          MyDBHelper.columnName1 + "=" + columnNameToBeDeleted, null);
```

$\uparrow$

returns a type int indicating the number of rows deleted.

- Update a record in database (create a method to do the following):

```
ContentValues initialValues = new ContentValues();

initialValues.put(MyDBHelper.columnName2, name2_str);

initialValues.put(MyDBHelper.columnName3, name3_str);

db.update(MyDBHelper.tableName,  initialValues,
              MyDBHelper.columnName1 + "=" + name1_str, null);
```

↑

returns a type int indicating the number of rows updated.

- Retrieve all records from database

```
public Cursor getAllRecords() {

   return db.query(

            MyDBHelper.tableName,

            new String[] {
                     MyDBHelper.columnName1,
                     MyDBHelper.columnName2,
                     MyDBHelper.columnName3 },

           null, null, null, null, null);
}
```

list of columns to return

- Retrieve a particular record from database

```
public Cursor getRecord(long id)  {
 Cursor mCursor = db.query(MyDBHelper.tableName,
                    new String[] {
                          MyDBHelper.columnName1,
                          MyDBHelper.columnName2,
                          MyDBHelper.columnName3 },
                    MyDBHelper.columnName1 + "=" + id,
                    null, null, null, null, null);

   if (mCursor != null) {
       mCursor.moveToFirst();
   }
   return mCursor;
}
```

list of columns to return

In MainActivity, do the following:

- Define the variable

```
MyDB db;
```

- In onCreate(), add

```
db = new MyDB(this);

ArrayList<String[]> myRecords = GetAllRecords();

for(int i = 0; i < myRecords.size(); i++) {

        String[] myRecord = myRecords.get(i);

        // columnName1, columnName2, columnName3 are stored in
        // myRecord[0], myRecord[1],  myRecord[2] respectively
}
```

Create a method that does the following:

- To Add a record,

```
db.open();

db.insertRecord(name2_str, name3_str);

// insertRecord is a user-defined method in MyDB that will call db.insert ()

db.close();
```

Create a method that does the following:

- To delete a record,

```
db.open();

db.deleteRecord(id);          // deleteRecord is a user-defined method in
                              // MyDB  that will call db.delete()

db.close();
```

Similarly, create methods that does the following:

- To Update a record,

```
db.open();

db.updateRecord(id, name2_str, name3_str);

// updateRecord is a user-defined method in MyDB that will call db.update ()

db.close();
```

- To retrieve a record,

```
db.open();

Cursor c = db.getRecord(id);

db.close();
```

• Example: GetRecord()

```
public String[] GetRecord(long id) {
    //---get a record---
    db.open();
    Cursor c = db.getRecord(id);
    String[] record = new String[3];
    if (c.moveToFirst()) {
       String[] temp = {c.getString(0),c.getString(1),c.getString(2)};
       record = temp;
    }
    else
       Toast.makeText(this, "No record found", Toast.LENGTH_LONG).show();

    db.close();
    return record;
 }
```

- To retrieve all records,

```
db.open();

Cursor c = db.getAllRecords();

db.close();
```

```
ArrayList<String[]> myRecords =
new ArrayList<String[]>();

if (c.moveToFirst()){
    do {
        String[] myRecord =
                          {c.getString(0),
                           c.getString(1),
                           c.getString(2)};
        myRecords.add(myRecord);
    } while (c.moveToNext());
}
```

- Example: GetAllRecords()

```
public ArrayList<String[]> GetAllRecords() {

     db.open();

     Cursor c = db.getAllRecords();

     ArrayList<String[]> records = new ArrayList<String[]>();

     if (c.moveToFirst()){
          do {
               String[] record = {c.getString(0),c.getString(1),c.getString(2)};
               records.add(record);
          } while (c.moveToNext());
     }
     db.close();
     return records;
  }
```

Summary

MyDBHelper

- Create a class MyDBHelper that extends SQLiteOpenHelper
- public static final String columnName1 = "_id";
  // need to define _id if use Cursor

MyDB

- Constructor instantiates MyDBHelper
- SQLiteDatabase db = DBHelper.getWritableDatabase();
- Insert/update db using ContentValues. ContentValues are key/value pairs

MainActivity

- MyDB db;

More on SQLite can be found at www.sqlite.org