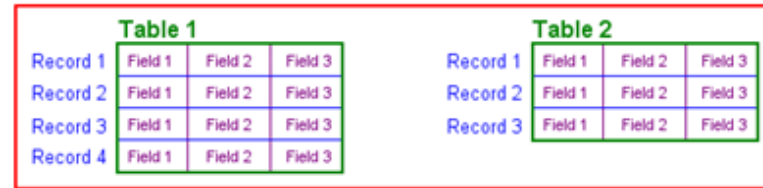


Database *Tables* provide the basic level of data structure in a database. A database can contain multiple tables and each table is designed to hold information of a specific type.



It is helpful to view a database table as being similar to a spreadsheet (see MS Excel):

- Columns represent data fields in the corresponding table.

	A	B	C	D	E	F
1	Year	Quarter	Date	Product Key	Prod Key	Sales Amount
2	2013	Q1	25-Mar	346	11003	1912.1544
3	2013	Q1	2/18/2013	336	14501	413.1463
4	2013	Q1	17-Jan	310	21768	2171.2942
5	2013	Q1	27-Feb	346	25863	1912.1544

-Rows are also sometimes referred to as *records* or *entries* and represent information each new record that is saved to a table is stored in a row.

Each *table* has a name that must be unique within that particular database.

Database Schema define the characteristics of the data stored in a database table.

More, it is also used to define the structure of entire databases and the relationship between the various tables contained in a database.

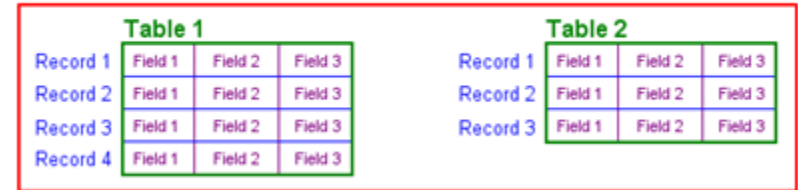
Primary Key: a database table must contain one column that can be used to identify each row in the table uniquely.

Remember: an index is used to query data faster, speed up sort operation, and enforce unique constraints.

.

A Relational Database System contains one or more objects called *tables*. The data or information for the database are stored in these tables. There are many RDSs

Database



- 4th Dimension
- Adabas D
- Alpha Five
- Apache Derby
- Aster Data
- Amazon Aurora
- Altibase
- CA Datacom
- CA IDMS
- Clarion
- ClickHouse
- Clustrix
- CSQL
- CUBRID
- DataEase
- Database Management Library
- Dataphor
- dBase
- Derby (aka Java DB)
- Empress Embedded Database
- Exasol
- EnterpriseDB
- eXtremeDB
- FileMaker Pro
- Firebird
- FrontBase
- Google Fusion Tables
- Greenplum
- GroveSite
- H2
- Helix database
- HSQLDB
- IBM DB2
- IBM Lotus Approach
- IBM DB2 Express-C
- Infobright
- Informix
- Ingres
- InterBase
- InterSystems Caché
- LibreOffice Base
- Linter
- MariaDB
- MaxDB
- MemSQL
- Microsoft Access
- Microsoft Jet Database Engine (part of Microsoft Access)
- Microsoft SQL Server
- Microsoft SQL Server Express Server)
- Microsoft Visual FoxPro
- Mimer SQL
- MonetDB
- mSQL
- MySQL
- Netezza
- NexasDB
- NonStop SQL
- NuoDB
- Omnis Studio
- Openbase
- OpenLink Virtuoso (Open Source Edition)
- OpenLink Virtuoso Universal Server
- OpenOffice.org Base
- Oracle
- Oracle Rdb for OpenVMS
- Panorama
- Paradox
- Pervasive PSQL
- Polyhedra
- PostgreSQL
- Postgres Plus Advanced Server
- Progress Software
- RDM Embedded
- RDM Server
- R:Base
- SAND CDBMS
- SAP HANA
- SAP Adaptive Server Enterprise
- SAP IQ (formerly known as Sybase IQ)
- SQL Anywhere (formerly known as Sybase Adaptive Server Anywhere and Watcom SQL)
- solidDB
- SQLBase
- SQLite
- SQream DB
- SAP Advantage Database Server (formerly known as Sybase Advantage Database Server)
- Teradata
- Tiberio
- TimesTen
- Trafodion
- txtSQL
- Unisys RDMS 2200
- UniData
- UniVerse
- Vectorwise
- Vertica
- VoltDB

*But for all of these there is/exist **SQL** (Structured Query Language)*

SQL = Structured Query Language.

SQL is used to communicate with a database.

i.e.

SQL is a programming language that is used to communicate with and manipulate databases.

The SQL programming language was first developed in the 1970s by IBM researchers Raymond Boyce and Donald Chamberlin. The programming language, known then as SEQUEL, was created following the publishing of Edgar Frank Todd's paper, "A Relational Model of Data for Large Shared Data Banks," in 1970.

According to ANSI (American National Standards Institute), SQL is the standard language for relational database management systems.

SQL statements are used to perform tasks such as:

- update data on a database
- retrieve data from a database.
- query the database

Some common Relational DataBase Management Systems (RDBMS) that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access etc.

- Rather than trying to write an SQL for their own databases, many companies use a database management system that has SQL already built in to it. Developed and distributed by Oracle, MySQL is one of the most popular SQL database management systems currently available. The software is an open source version, which means it can be downloaded and used for free.
- According to the web hosting service GoDaddy, MySQL is a sophisticated and powerful relational database management system used by many websites to create and change content quickly.

Wikipedia:

a mobile database is “either a stationary database that can be connected to by a mobile computing device [...] over a mobile network, or a database which is actually stored by the mobile device,”

we refer to databases that run on the mobile device itself.

Database in Android Studio

What is SQLite ?

SQLite is an embedded, relational database management system (RDBMS).

- ⇔ *SQLite* is provided in the form of a **library**, linked into applications
- ⇔ NO standalone database server
- ⇔ All database operations are handled internally using functions from SQLite library.

SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, is contained in a single disk file.

The database [file format](#) is **cross-platform** - you can freely copy a database between 32-bit and 64-bit systems

SQLite is written in C language,

Android apps are written in Java

SO

to bridge this “language gap”, the Android SDK includes a set of classes that provide a Java layer on top of the *SQLite* database management system.

In SQLite we can have:

Unique indexes, used not only for performance, but also for data integrity. A unique index does not allow any duplicate values to be inserted into the table.

Composite index, an index on two or more columns of a table

Indexes should not be used in small tables, tables with columns that contain a high number of NULL values

1. Cursor class

```
java.lang.Object
└─ android.database.AbstractCursor
    └─ android.database.AbstractWindowedCursor
        └─ android.database.sqlite.SQLiteCursor
```

exposes results from a query on a [SQLiteDatabase](#)

Include methods:

void close() – Releases all resources used by the cursor and closes it.

getCount() – Returns the number of rows/records contained within the result set.

moveToFirst() – Moves to the first row/record within the result set.

moveToLast() – Moves to the last row/record in the result set.

moveToNext() – Moves to the next row/record in the result set.

move() – Moves by a specified offset from the current position in the result set.

get<type>() – Returns the value of the specified <type> contained at the specified column index of the row at the current cursor position (variations consist of *getString()*, *getInt()*, *getShort()*, *getFloat()* and *getDouble()*).

Android SQLite Java Classes

SQLiteDatabase class methods:

insert() – Inserts a new row into a database table.

delete() – Deletes rows from a database table.

query() – Performs a specified database query and returns matching results via a Cursor object.

execSQL() – Executes a single SQL statement that does not return result data.

rawQuery() – Executes an SQL query statement and returns matching results in the form of a Cursor object.

SQLiteOpenHelper class methods:

onCreate() – Called when the database is created for the first time.

onUpgrade() – Called in the event that the application code contains a more recent database version number reference.

getWritableDatabase() – Opens or creates a database for reading and writing. Returns a reference to the database in the form of a SQLiteDatabase object.

getReadableDatabase() – Creates or opens a database for reading only. Returns a reference to the database in the form of a SQLiteDatabase object.

close() – Closes the database.

And many many more methods exist to work with databases on Android

(could be find in <https://developer.android.com/reference/android/database/sqlite/>)

SQLite advantages for a mobile database :

The code for SQLite is in the [public domain](#) and is thus free for use for any purpose, commercial or private.

SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file.

full offline modus for apps that depend on stored data

stable and predictable performance independent from network availability

personal data can be stored only on your device => safe

Developers can define exactly the data schema they want

Developers have full control, e.g. handwritten SQL queries

SQLite advantages for a mobile database :

SQL is a powerful and established query language, and SQLite supports most of it
Debuggable data: developers can grab the database file and analyze it

Rock-solid, widely used technology, established since the year 2000

Using SQLite, you don't need to setup a database server & interface.

Android stores your database in your app's private folder. Your data is secure, because by default this area is not accessible to other apps or the user.

SQLite has a native support on Android

SQLite disadvantages for a mobile database :

Using SQLite means a lot of code and thus inefficiencies (also in the long run with the app maintenance)

No compile time checks (e.g. SQL queries)

The performance of SQLite is unreliable (different devices -> different performance)

SQL is another language to master

SQL queries can get long and complicated

SQLite is a little slow because need to access the file system.

Only for a local and small database, access is fast: less than 5ms for read and query.

On a mobile device, could be a limitation of the storage, so no big databases. If you need a database for a huge amount to data, better use a remote DB on an external server

ORM Object-Relational Mapper

Keeps SQLite abstraction
letting database entities to Java objects easier.

REALM (June 2014)

Realm Mobile Database is a database designed for mobile devices from the ground up.

Realm is not an abstraction built on top of SQLite (like ORM), a whole new database engine was created.

Rather than a relational model, it is based on an object store.

Its core consists of a self-contained C++ library.

It currently supports Android, iOS(Objective-C and Swift), Xamarin, and React Native.

More at <https://realm.io/>

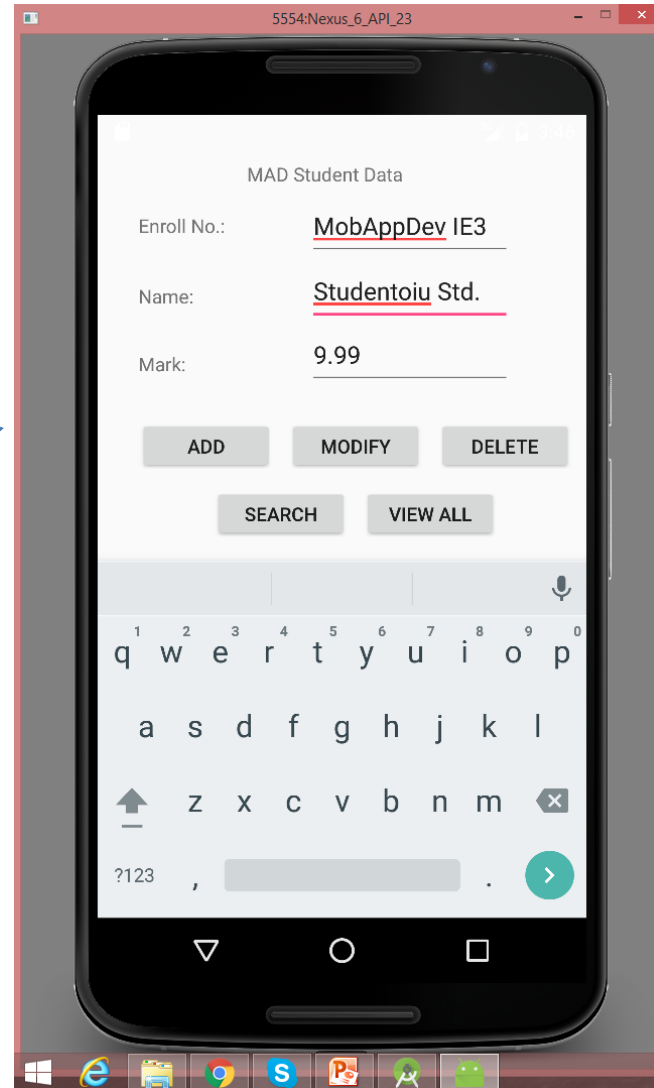
Goal:

An Android app for managing a sample student database (EnrollNumber, StudentName and StudentMarks). The user's operations will be : *Add, Modify, Delete, Search* and *ViewAll* records.

The user interface layout
(based on an *AbsoluteLayout* style)

Absolute layouts are less flexible and harder to maintain than other types of layouts without absolute positioning.

! it's a little deprecated !



content_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/myLayout"
    android:stretchColumns="0"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
<TextView android:text="@string/title"
    android:layout_x="110dp"
    android:layout_y="10dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
<TextView android:text="@string/enroll_no"
    android:layout_x="30dp"
    android:layout_y="48dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
<EditText android:id="@+id/editEnrollno"
    android:layout_x="155dp"
    android:layout_y="33dp"
    android:layout_width="150dp"
    android:layout_height="50dp"/>
<TextView android:text="@string/name"
    android:layout_x="30dp"
    android:layout_y="100dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
<EditText android:id="@+id/editName"
    android:inputType="text"
    android:layout_x="155dp"
    android:layout_y="81dp"
    android:layout_width="150dp"
    android:layout_height="50dp"/>
<TextView android:text="@string/marks"
    android:layout_x="30dp"
    android:layout_y="150dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
<EditText android:id="@+id/editMarks"
    android:inputType="numberDecimal"
    android:layout_x="155dp"
    android:layout_y="128dp"
    android:layout_width="150dp"
    android:layout_height="50dp"/>
<Button android:id="@+id/btnAdd"
    android:text="@string/add"
    android:layout_x="30dp"
    android:layout_y="200dp"
    android:layout_width="100dp"
    android:layout_height="40dp"/>
<Button android:id="@+id/btnDelete"
    android:text="@string/delete"
    android:layout_x="250dp"
    android:layout_y="200dp"
    android:layout_width="100dp"
    android:layout_height="40dp"/>
<Button android:id="@+id/btnModify"
    android:text="@string/modify"
    android:layout_x="140dp"
    android:layout_y="200dp"
    android:layout_width="100dp"
    android:layout_height="40dp"/>
<Button android:id="@+id/btnSearch"
    android:text="@string/view"
    android:layout_x="85dp"
    android:layout_y="250dp"
    android:layout_width="100dp"
    android:layout_height="40dp"/>
<Button android:id="@+id/btnViewAll"
    android:text="@string/view_all"
    android:layout_x="195dp"
    android:layout_y="250dp"
    android:layout_width="100dp"
    android:layout_height="40dp"/>
</AbsoluteLayout>

```

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">MAD Student Database App</string>
  <string name="title">MAD Student Data</string>
  <string name="enroll_no">Enroll No.: </string>
  <string name="name">Name: </string>
  <string name="marks">Mark: </string>
  <string name="add">Add</string>
  <string name="delete">Delete</string>
  <string name="modify">Modify</string>
  <string name="view">Search</string>
  <string name="view_all">View All</string>
</resources>
```

MainActivity.java (1)

```
package com.example.mafteiu_scai.mystudents;

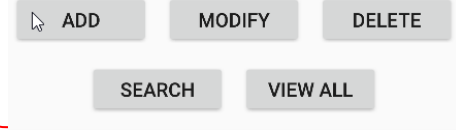
import android.app.Activity;
import android.app.AlertDialog.Builder;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import com.example.mafteiu_scai.mystudents.R;
```

import SQLite classes.

Called when the activity is first created.

```
public class MainActivity extends Activity implements OnClickListener {
    EditText editEnrollno, editName, editMarks;
    Button btnAdd, btnDelete, btnModify, btnSearch, btnViewAll;
    SQLiteDatabase db;
```

Enroll No.:	MobAppDev IE3
Name:	Studentoiu Std.
Mark:	9.99



```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.content_main);
    editEnrollno = (EditText) findViewById(R.id.editEnrollno);
    editName = (EditText) findViewById(R.id.editName);
    editMarks = (EditText) findViewById(R.id.editMarks);
    btnAdd = (Button) findViewById(R.id.btnAdd);
    btnDelete = (Button) findViewById(R.id.btnDelete);
    btnModify = (Button) findViewById(R.id.btnModify);
    btnSearch = (Button) findViewById(R.id.btnSearch);
    btnViewAll = (Button) findViewById(R.id.btnViewAll);
    btnAdd.setOnClickListener(this);
    btnDelete.setOnClickListener(this);
    btnModify.setOnClickListener(this);
    btnSearch.setOnClickListener(this);
    btnViewAll.setOnClickListener(this);
    db = openOrCreateDatabase("MADStudentsDB", Context.MODE_PRIVATE, null);
    db.execSQL("CREATE TABLE IF NOT EXISTS student(enrollno VARCHAR,name VARCHAR,marks VARCHAR);");
}
```

Called when when a button is pressed.

Path to database file to open and/or create

the structure of table student

MainActivity.java (2)

```
public void onClick(View view) {
```

```
    if (view == btnAdd) {
        if (editEnrollno.getText().toString().trim().length() == 0 ||
            editName.getText().toString().trim().length() == 0 ||
            editMarks.getText().toString().trim().length() == 0) {
            showMessage("Error", "Please enter all student data");
            return;
        }
        db.execSQL("INSERT INTO student VALUES('" + editEnrollno.getText() + "','" + editName.getText() +
            "','" + editMarks.getText() + "')");
        showMessage("Success", "Record added");
        clearText();
    }
```

add students to the database under certain conditions

```
    if (view == btnDelete) {
        if (editEnrollno.getText().toString().trim().length() == 0) {
            showMessage("Error", "Please enter Student Enroll Number");
            return;
        }
        Cursor c = db.rawQuery("SELECT * FROM student WHERE enrollno='" + editEnrollno.getText() + "'", null);
        if (c.moveToFirst()) {
            db.execSQL("DELETE FROM student WHERE enrollno='" + editEnrollno.getText() + "'");
            showMessage("Success", "Record Deleted");
        } else {
            showMessage("Error", "Invalid Enroll Number");
        }
        clearText();
    }
```

delete a student from table, based in enroll number

The rawQuery method has two parameters:

-String query: The select statement

-String[] selection args: The arguments if a WHERE clause is included in the select statement

MainActivity.java (3)

```
if (view == btnModify) {  
    if (editEnrollno.getText().toString().trim().length() == 0) {  
        showMessage("Error", "Please enter Enroll Number");  
        return;  
    }  
    Cursor c = db.rawQuery("SELECT * FROM student WHERE enrollno=" + editEnrollno.getText() + "", null);  
    if (c.moveToFirst()) {  
        db.execSQL("UPDATE student SET name=" + editName.getText() + ",marks=" + editMarks.getText() +  
            " WHERE enrollno=" + editEnrollno.getText() + "");  
        showMessage("Success", "Student Data Modified");  
    } else {  
        showMessage("Error", "Invalid Enroll Number");  
    }  
    clearText();  
}  
if (view == btnSearch) {  
    if (editEnrollno.getText().toString().trim().length() == 0) {  
        showMessage("Error", "Please enter Student enroll number");  
        return;  
    }  
    Cursor c = db.rawQuery("SELECT * FROM student WHERE enrollno=" + editEnrollno.getText() + "", null);  
    if (c.moveToFirst()) {  
        editName.setText(c.getString(1));  
        editMarks.setText(c.getString(2));  
    } else {  
        showMessage("Error", "Invalid Enroll Number");  
        clearText();  
    }  
}
```

modify students's data

Search a record (student), based on enroll number

MainActivity.java (4)

```

if (view == btnViewAll) {
    Cursor c = db.rawQuery("SELECT * FROM student", null);
    if (c.getCount() == 0) {
        showMessage("Error", "No students found in database");
        return;
    }
    StringBuffer buffer = new StringBuffer();
    while (c.moveToNext()) {
        buffer.append("Enroll Number: " + c.getString(0) + "\n");
        buffer.append("Name: " + c.getString(1) + "\n");
        buffer.append("Mark: " + c.getString(2) + "\n\n");
    }
    showMessage("MAD Student Data", buffer.toString());
}

```

View all records (students) of database

```

}
public void showMessage(String title, String message) {
    Builder builder = new Builder(this);
    builder.setCancelable(true);
    builder.setTitle(title);
    builder.setMessage(message);
    builder.show();
}

```

Display error/warning messages for user

```

public void clearText() {
    editEnrollno.setText("");
    editName.setText("");
    editMarks.setText("");
    editEnrollno.requestFocus();
}
}

```

*Clear the **TextEdit** forms*

A possible style to learn from Internet

Please GO TO:

<https://examples.javacodegeeks.com/android/core/database/android-database-example/>

Take a look to this page

Ignore the IDE used (Eclipse)

Listen to the following questions, think and give good answers(you can use the Internet)

1 What is the purpose of this application?

2 Who could be users?

3 How the UI looks (drawn on the whiteboard)

1 What is the purpose of this application?

A database for

2 Who could be users?

3 How the UI looks (drawn on the whiteboard)

Go to the second paragraph (Creating the layout of the main AndroidDatabaseExample)
Listen to the following questions, think and give good answers (you can use the Internet)

What means the following?

1 `android:layout_width="match_parent"`

2 `android:layout_height="match_parent"`

3 `android:background="#ffffff"`

4 `android:orientation="vertical"`

The answers:

1 `android:layout_width="match_parent"`

2 `android:layout_height="match_parent"`

"match_parent" makes the view expand to as much as possible (horizontal and vertical) within the parent view.

! in general, parent_view isn't the same sizes as mobile's display

3 `android:background="#ffffff"`

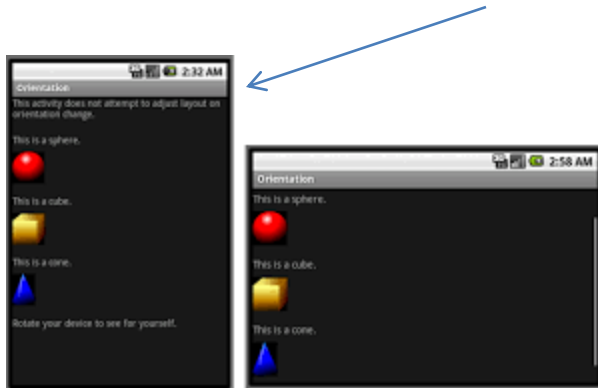
RGB ff=percent of red,

ffffff = 33%red + 33%green + 33%blue = white

000000 =..... = black

4 `android:orientation="vertical"`

specify the layout direction



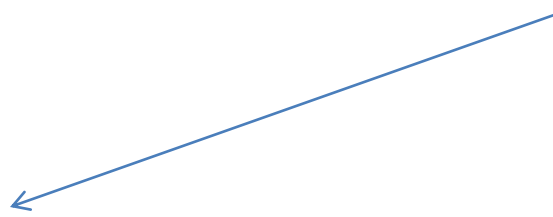
Important note: write all these on a sheet of paper: our memory is limited

What is ?

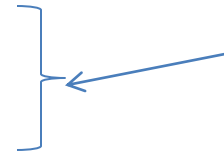


```
<ListView  
  android:id="@android:id/list"  
  android:layout_width="match_parent"  
  android:layout_height="wrap_content" >  
</ListView>
```

What is ?



Which will be the result?



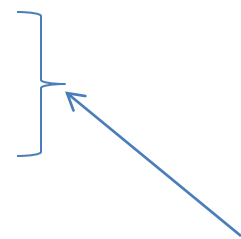
What is ? : A List Control



What is ? ID on strings.xml



```
<ListView  
    android:id="@android:id/list"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content" >  
</ListView>
```



Which will be the result?

match_parent: will occupy the complete area that the parent view has (it will be as big as it's parent)

wrap_content: space big enough for its contents to get enclosed (does not waste space)

Go to section 3, read the java source code and try to answer to the following questions:

1 What type of class is Book class and for what scope is it?

1 What type of class is Book class and for what scope is it?

Answers:

publica class that can be viewed/run from other classes

Methods used for input data from users and update them

The name of the database ?

The name and structure/columns of the used table

What datatypes for this structure are used ?

The name and the structure of the database?

BookDB

The name and structure/columns of the used table

Books


Id
Title
Author



columns for each book in part

What datatypes for this structure are used ?

Int
String
String



What libraries should be imported to operate with databases in Android apps?

What libraries should be imported to operate with databases in Android apps?

```
import android.database.Cursor;  
import android.database.sqlite.SQLiteDatabase;  
import android.database.sqlite.SQLiteOpenHelper;
```

What is the function used to create the database and what parameter has?

What is the function used to create the database and what parameter has?

```
db.execSQL(CREATE_BOOK_TABLE);
```

where `CREATE_BOOK_TABLE` is a string

```
String CREATE_BOOK_TABLE = "CREATE TABLE books ( " + "id INTEGER PRIMARY KEY  
AUTOINCREMENT, " + "title TEXT, " + "author TEXT )";
```

to create the table, you would call

```
db.execSQL(DATABASE_CREATE)
```

which would execute your table create statement on your application's SQLite database.

Note: is the first operation needed if no databases exist

**in which source file
and where (which lines)
and how many entries/records are added in database directly
from the code (without user input/type)?**

in which source file
and where (which lines)
and how many entries/records are added in database directly from
the code (without user input)?

AndroidDatabaseExample.java

Lines 28-40

13 records/entries

How many activities are in this app and what are their names?

How many activities are in this app and what are their names?

Only one: *BookActivity* described in *BookActivity.java* file

In fact, a main activity

1. Project realization - 1 point (*copy-paste projects will be eliminated*)
2. Purpose, utility, users - 1 point
3. New Idea (Creativity, Innovation) – 0.5 point / each contribution
4. Number of activities in the project: 0.5 points / activity or fragment
5. Number of active control types used in the layout: 0.25 points / control type
6. A database: 1 point
7. Correct and complete answers to 1-2 questions from source code: 3 points
(0 points -> project elimination)
8. Project documentation: 1 point

Deadline for project; before 12th week

It will contain: apk, source code archive

Deadline for documentation: when you will present the report

Presentations: weeks 12, 13 and 14

You can send and present before (send your project two days before the day when you want to present)

Title page

This page will include:

- the name of the discipline
- the student's name
- project name
- data of dispatch
- recipient's name

Summary/ Abstract

In summary, description of the work / project (not problematic in general)

The purpose / utility of the project

The purpose/goal of the project, for what it is used. Utilities such as "user distraction" are inappropriate for student status. And no phrases such as "I chose this project to learn to program in Java" or "pass the exam"

Users

For who is your app, who uses it? You must mention: age, studies, profession and arguments for all of these.

My contribution

.....

Introduction

A resume of the problem, using references. More, a short description of some apps that solve the same problem.

Functionality

One (or more) UML diagram is enough.

UI

One UI diagram is enough

Running the app (user's manual)

Explain this part from user's point of view. Do not include algorithms, data structures, implementation. All input data , output data, menus and functionality must be explained. In fact, this is the User's Manual

App's structure (technical manual)

This section represents the Technical Manual. Must be treated from programmers ' point of view. Each part of the code must be explained (functions, classes, data structures, algorithms). Others programmers must understand all about your app.

Conclusions and Future work

Conclusions, comparisons with a similar apps

What you want to add or how you want to improve your app in future.

Reference

All external library, figures, definitions, math relations, etc etc etc.... who do not belong to you, must be written on this section