Bullet Points

- Android uses SQLite as its backend database to persist data.
- The SQLiteDatabase class gives you access to the SQLite database.
- A SQLite helper lets you create and manage SQLite databases. You create a SQLite helper by extending the SQLiteOpenHelper class.
- You must implement the SQLiteOpenHelper onCreate() and onUpgrade() methods.
- The database gets created the first time it needs to be accessed. You need to give the database a name and version number, starting at 1. If you don’t give the database a name, it will just get created in memory.
- The onCreate() method gets called when the database first gets created.
- The onUpgrade() method gets called when the database needs to be upgraded.

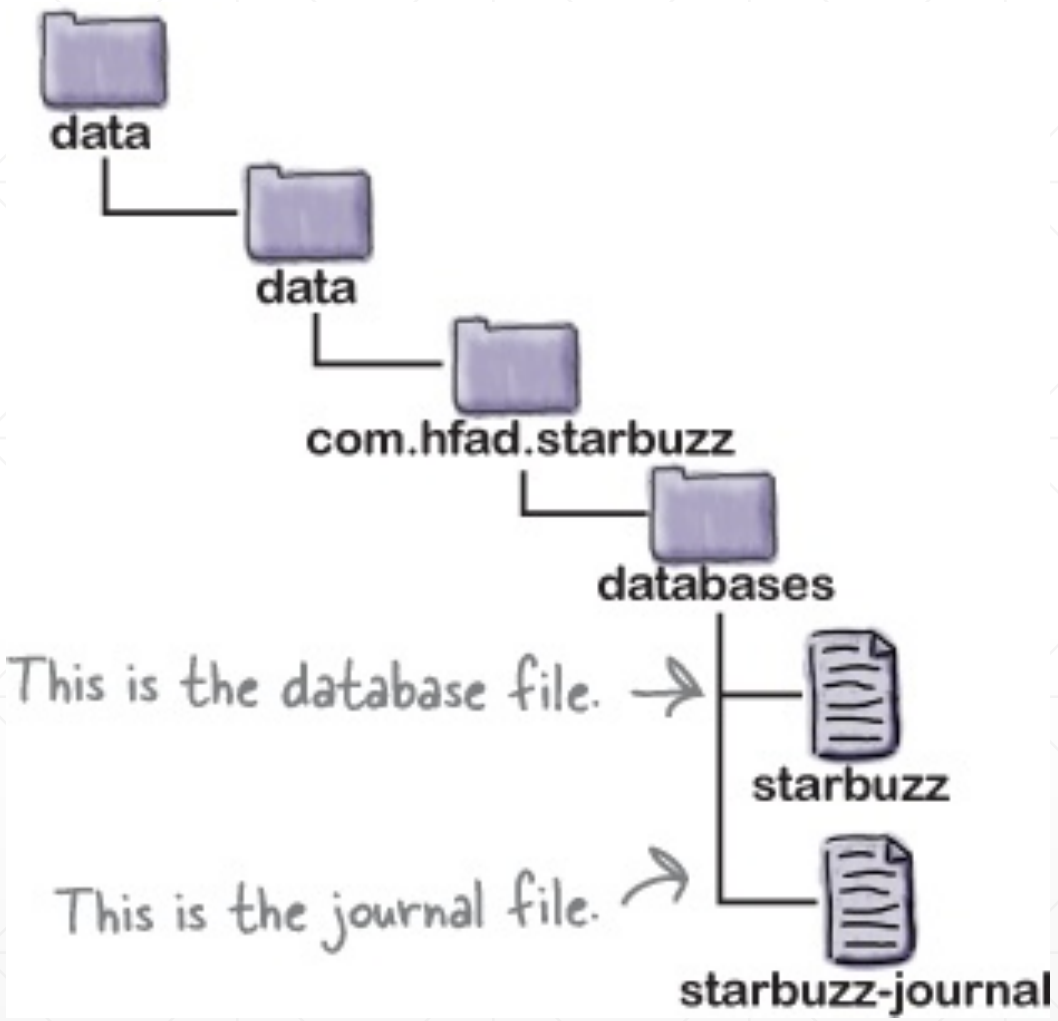
Bullet Points

- Execute SQL using the SQLiteDatabase execSQL(String) method.
- Add records to tables using the insert() method.
- Update records using the update() method.
- Remove records from tables using the delete() method.


10:26
Starbuzz Coffee
Drinks
Food
Stores
The top-level activity displays a list of options.
10:26
Latte
Cappuccino
Filter
Clicking on the Drinks option shows you a list of the available drinks.
10:47
Latte
A couple of espresso shots with steamed milk
When you click on a drink, its details are displayed.

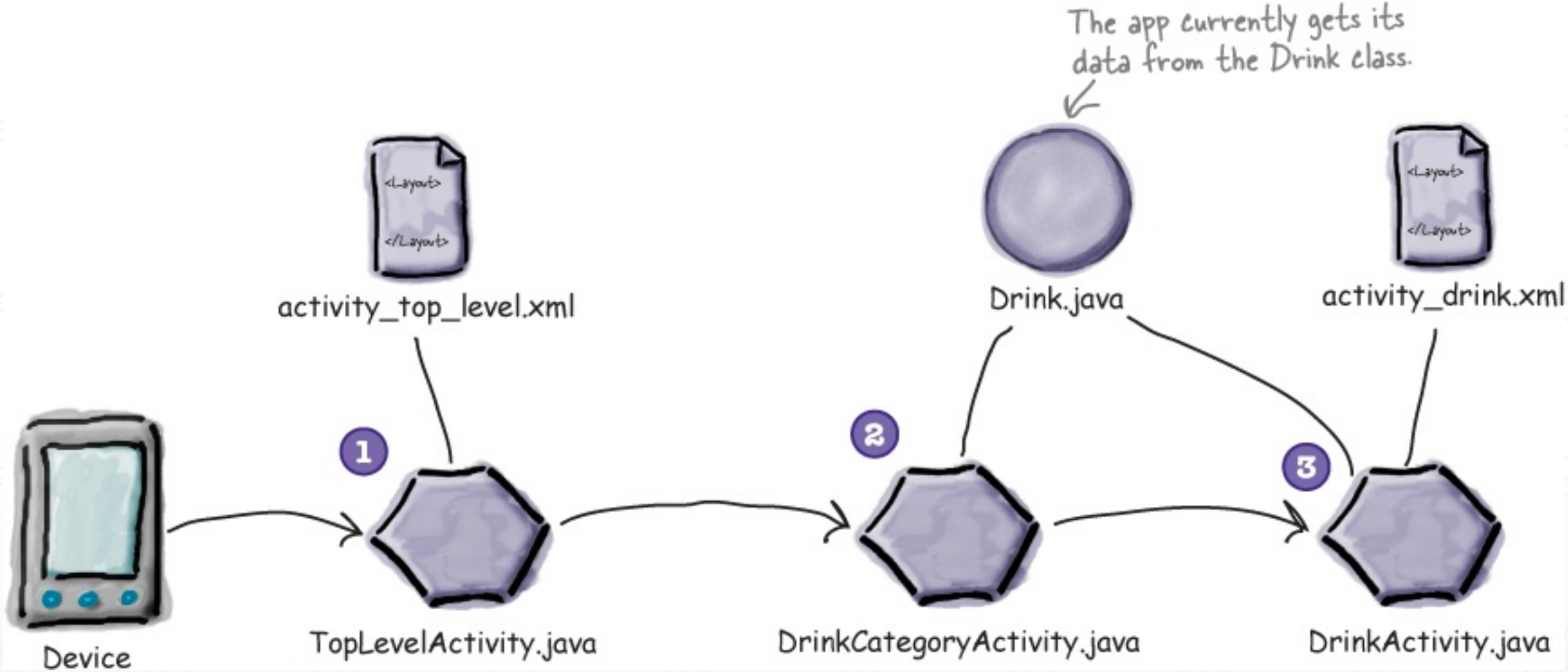
SQLite

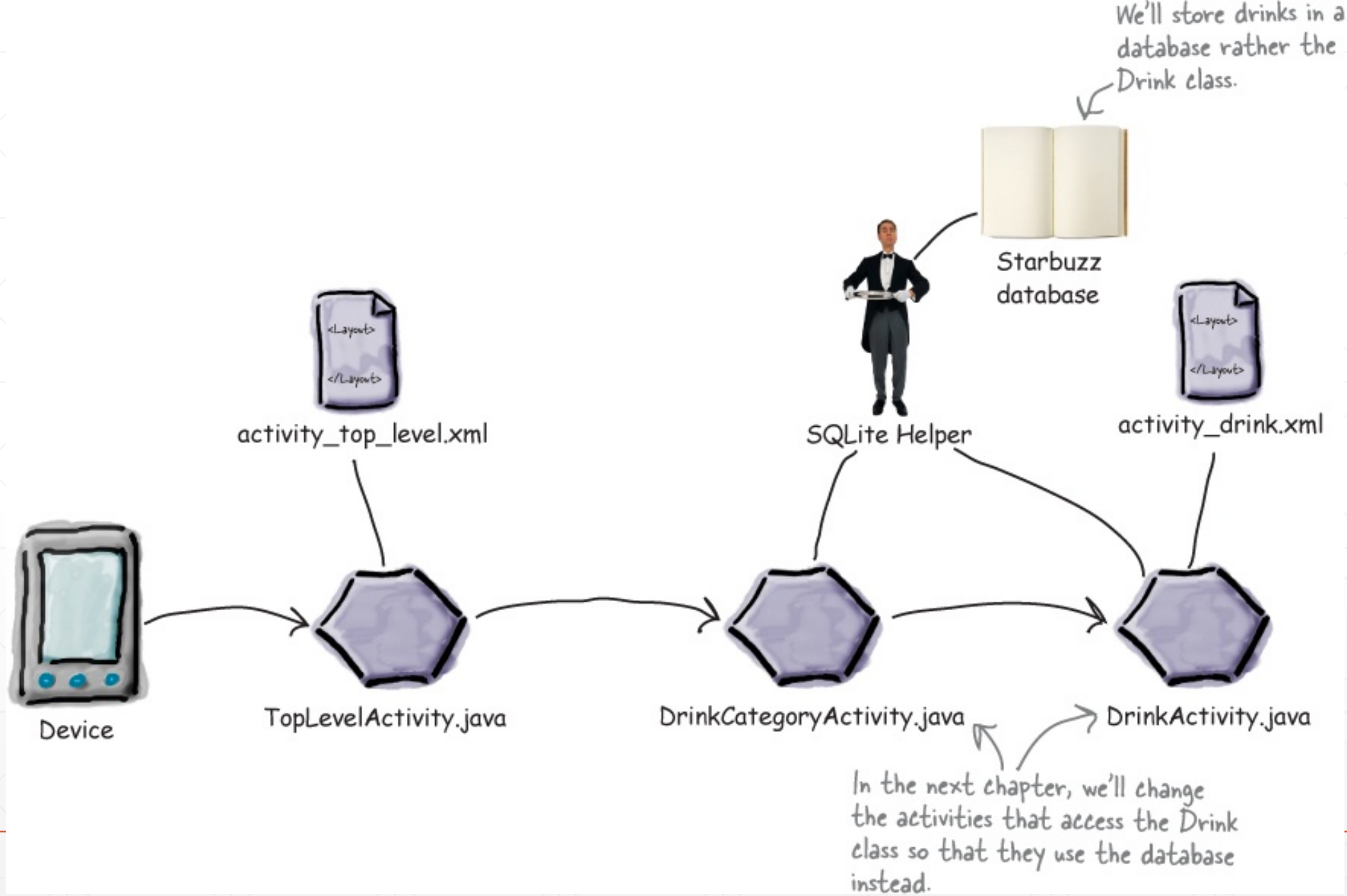
- Lightweight
- Optimized for single user
- Stable and fast


data
data
com.hfad.starbuzz
databases
This is the database file.
starbuzz
This is the journal file.
starbuzz-journal

SQLite

- SQLite Helper
- Cursors
- SQLite Databases


The app currently gets its data from the Drink class.
<Layout>
</Layout>
activity_top_level.xml
Drink.java
<Layout>
</Layout>
activity_drink.xml
1
2
3
Device
TopLevelActivity.java
DrinkCategoryActivity.java
DrinkActivity.java


We'll store drinks in a database rather the Drink class.
Starbuzz database
SQLite Helper
activity_top_level.xml
activity_drink.xml
Device
TopLevelActivity.java
DrinkCategoryActivity.java
DrinkActivity.java
In the next chapter, we'll change the activities that access the Drink class so that they use the database instead.

Tasks

- Create Database
- Create table

Creating the database

When you first install an app, the database file won't exist. The SQLite helper will make sure the database file is created with the correct name and with the correct table structures installed.

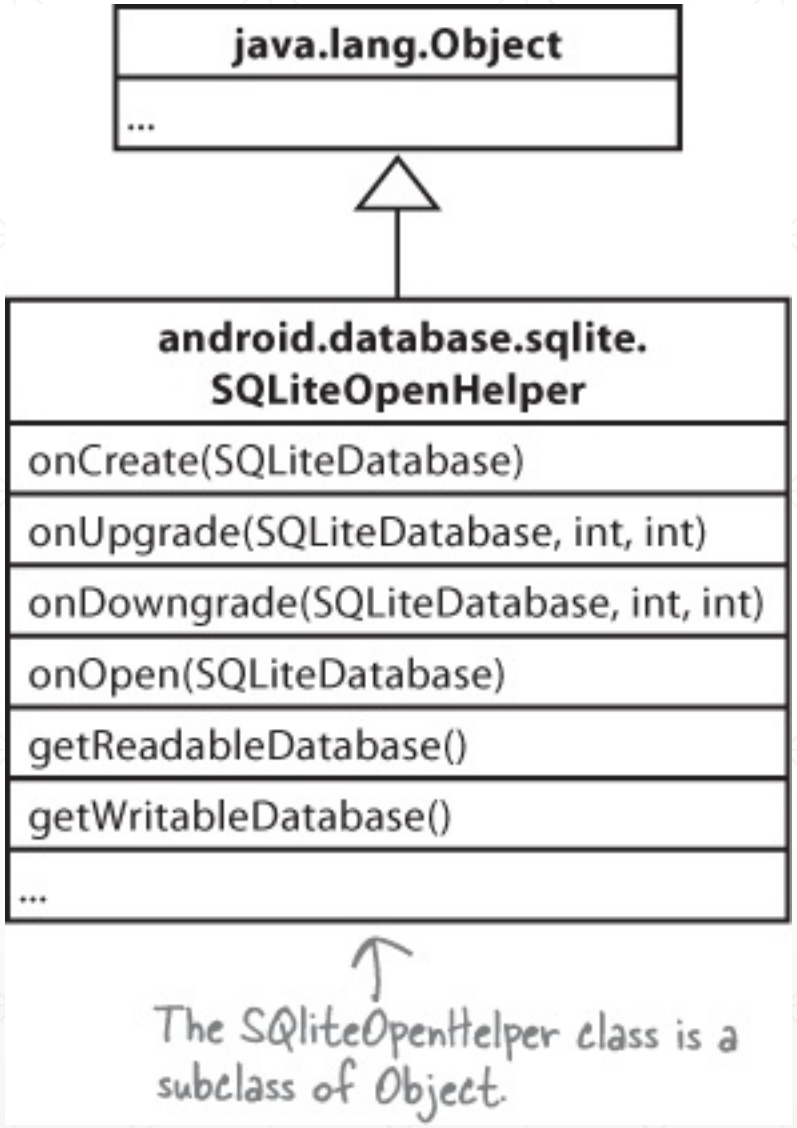
Getting access to the database

Our app shouldn't need to know all of the details about where the database file is, so the SQLite helper can serve us with an easy-to-use database object whenever we need it. At all hours, day or night.

The SQLite helper

Keeping the database shipshape

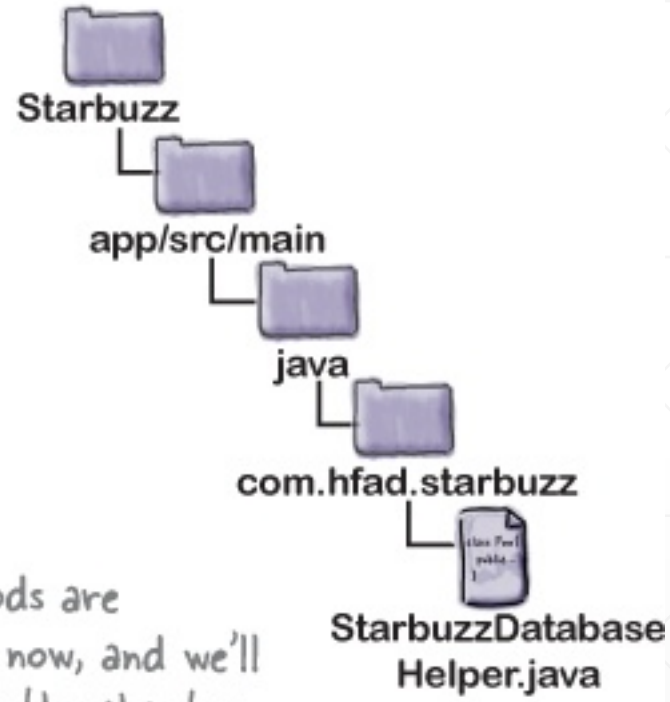
The structure of the database will probably change over time, and the SQLite helper can be relied upon to convert an old version of a database into a shiny, spiffy new version, with all the latest database structures it needs.


java.lang.Object
...
android.database.sqlite.
SQLiteOpenHelper
onCreate(SQLiteDatabase)
onUpgrade(SQLiteDatabase, int, int)
onDowngrade(SQLiteDatabase, int, int)
onOpen(SQLiteDatabase)
getReadableDatabase()
getWritableDatabase()
...
The SQLiteOpenHelper class is a subclass of Object.

```
package com.hfad.starbuzz;

import android.database.sqlite.SQLiteOpenHelper;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;

class StarbuzzDatabaseHelper extends SQLiteOpenHelper {

    StarbuzzDatabaseHelper(Context context) {
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    }
}
```

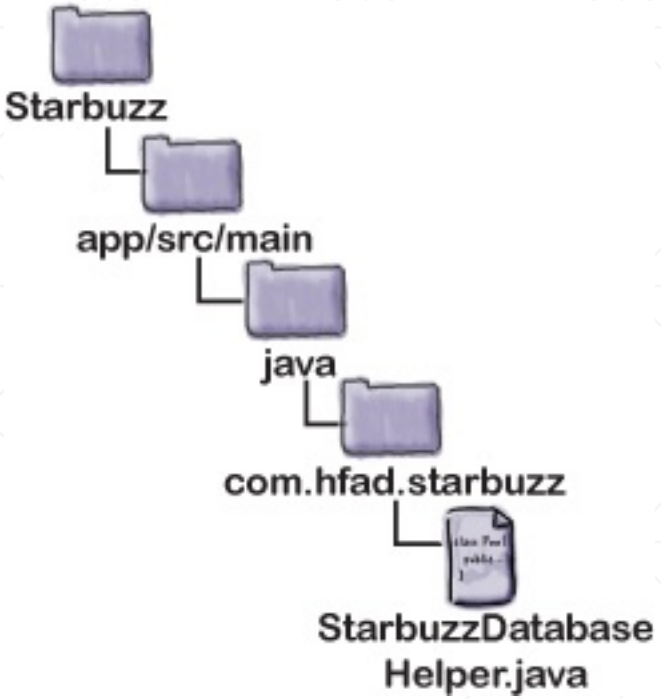
SQLite helpers must extend the SQLiteOpenHelper class.

The onCreate() and onUpgrade() methods are mandatory. We've left them empty for now, and we'll look at them in more detail throughout the chapter.



SQLite database

Name: "starbuzz"
Version: 1




```
...
class StarbuzzDatabaseHelper extends SQLiteOpenHelper {

    private static final String DB_NAME = "starbuzz"; // the name of our database
    private static final int DB_VERSION = 1; // the version of the database

    StarbuzzDatabaseHelper(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
    }
...
}
```

We're calling the constructor of the SQLiteOpenHelper superclass, and passing it the database name and version.

This parameter is an advanced feature relating to cursors. We're covering cursors in the next chapter.

We've done everything we need to get the database created when we need it.

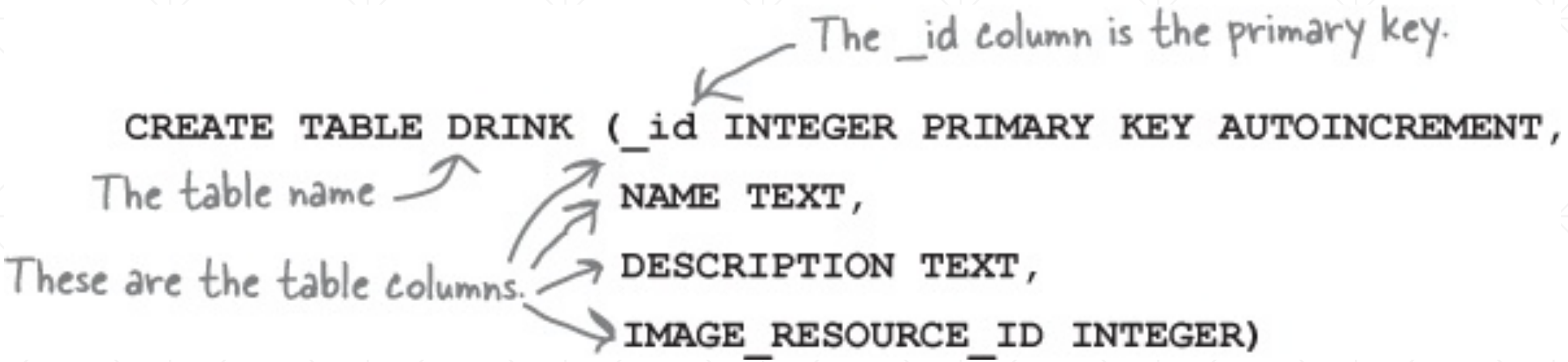
Create database

Create table

The columns in the table are _id, NAME, DESCRIPTION, and IMAGE_RESOURCE_ID. The Drink class contained similarly named attributes.

_id	NAME	DESCRIPTION	IMAGE_RESOURCE_ID
1	“Latte”	"Espresso and steamed milk"	54543543
2	“Cappuccino”	"Espresso, hot milk and steamed-milk foam"	654334453
3	“Filter”	"Our best drip coffee"	44324234

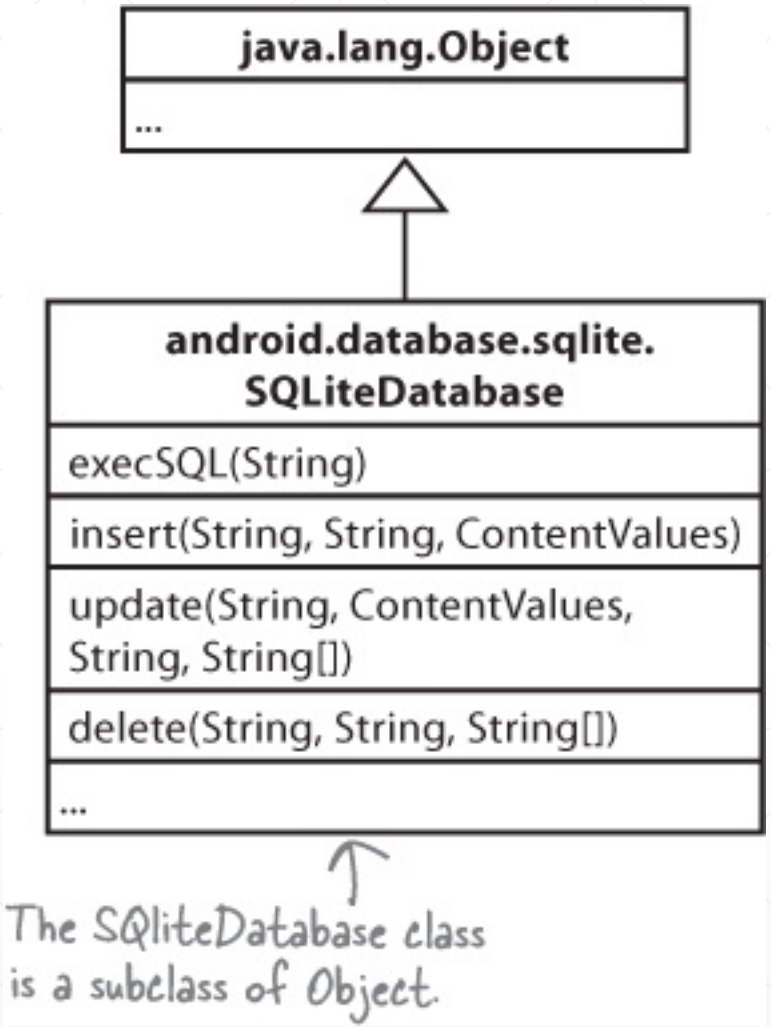
Type	Description
INTEGER	Any integer type
TEXT	Any character type
REAL	Any floating-point number
NUMERIC	Booleans, dates, and date-times
BLOB	Binary Large Object

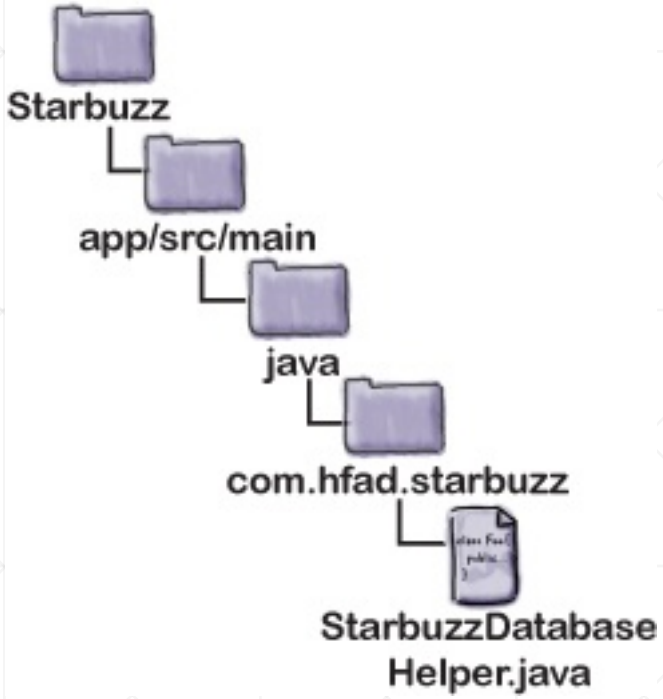

The _id column is the primary key.
CREATE TABLE DRINK (_id INTEGER PRIMARY KEY AUTOINCREMENT,
The table name
NAME TEXT,
These are the table columns.
DESCRIPTION TEXT,
IMAGE_RESOURCE_ID INTEGER)

```
SQLiteDatabase.execSQL(String sql);
```

Execute the SQL in the String on the database.

```
@Override
public void onCreate(SQLiteDatabase db){
    db.execSQL("CREATE TABLE DRINK ("
                + "_id INTEGER PRIMARY KEY AUTOINCREMENT, "
                + "NAME TEXT, "
                + "DESCRIPTION TEXT, "
                + "IMAGE_RESOURCE_ID INTEGER);");
}
```


java.lang.Object
...
android.database.sqlite.
SQLiteDatabase
execSQL(String)
insert(String, String, ContentValues)
update(String, ContentValues,
String, String[])
delete(String, String, String[])
...
The SQliteDatabase class
is a subclass of Object.




```
ContentValues drinkValues = new ContentValues();
drinkValues.put("NAME", "Latte");
drinkValues.put("DESCRIPTION", "Espresso and steamed milk");
drinkValues.put("IMAGE_RESOURCE_ID", R.drawable.latte);
```

This will put the value "Espresso and steamed milk" in the DESCRIPTION column.

You need a separate call to the put() method for each value you want to enter.

```
db.insert("DRINK", null, drinkValues);
```

_id	NAME	DESCRIPTION	IMAGE_RESOURCE_ID
1	“Latte”	"Espresso and steamed milk"	54543543

A shiny new record gets inserted into the table.

```
db.insert(String table, String nullColumnHack, ContentValues values);
```

This inserts a single row into the table. To insert multiple rows, you need to make repeated calls to the insert() method.

```
public int update (String table,
                   ContentValues values,
                   String whereClause,
                   String[] whereArgs)
```

```
ContentValues drinkValues = new ContentValues();
drinkValues.put("DESCRIPTION", "Tasty");
db.update("DRINK",
          drinkValues,
          "NAME = ?",
          new String[] {"Latte"});
```

This will put the value "Tasty" in the DESCRIPTION column.

Update the DESCRIPTION column to "Tasty" in the DRINK table where NAME = "Latte".

_id	NAME	DESCRIPTION	IMAGE_RESOURCE_ID
1	"Latte"	~~"Espresso and steamed milk"~~ "Tasty"	54543543

```
db.update("DRINK",
          drinkValues,
          "NAME = ? OR DESCRIPTION = ?",
          new String[] {"Latte", "Our best drip coffee"});
```

This means: Where NAME = "Latte" or DESCRIPTION = "Our best drip coffee".

```
db.update("DRINK",
          drinkValues,
          "_id = ?",
          new String[] {Integer.toString(1)});
```

Convert the int 1 to a String value.

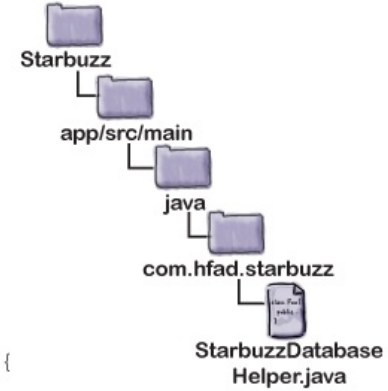
```
public int delete (String table,
                   String whereClause,
                   String[] whereArgs)
```

```
db.delete("DRINK",
          "NAME = ?",
          new String[] {"Latte"});
```

Can you see how similar this is to the update() method?

The entire row is deleted

_id	NAME	DESCRIPTION	IMAGE_RESOURCE_ID
1	"Latte"	"Espresso and steamed milk"	54543543




```
package com.hfad.starbuzz;

import android.content.ContentValues;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

class StarbuzzDatabaseHelper extends SQLiteOpenHelper{

    private static final String DB_NAME = "starbuzz"; // the name of our database
    private static final int DB_VERSION = 1; // the version of the database

    StarbuzzDatabaseHelper(Context context){
        super(context, DB_NAME, null, DB_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db){
        db.execSQL("CREATE TABLE DRINK (_id INTEGER PRIMARY KEY AUTOINCREMENT, "
                + "NAME TEXT, "
                + "DESCRIPTION TEXT, "
                + "IMAGE_RESOURCE_ID INTEGER);");
        insertDrink(db, "Latte", "Espresso and steamed milk", R.drawable.latte);
        insertDrink(db, "Cappuccino", "Espresso, hot milk and steamed-milk foam",
                R.drawable.cappuccino);
        insertDrink(db, "Filter", "Our best drip coffee", R.drawable.filter);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    }

    private static void insertDrink(SQLiteDatabase db, String name,
                                    String description, int resourceId) {
        ContentValues drinkValues = new ContentValues();
        drinkValues.put("NAME", name);
        drinkValues.put("DESCRIPTION", description);
        drinkValues.put("IMAGE_RESOURCE_ID", resourceId);
        db.insert("DRINK", null, drinkValues);
    }
}
```

Say what the database name and version is. It's the first version of the database, so the version should be 1.

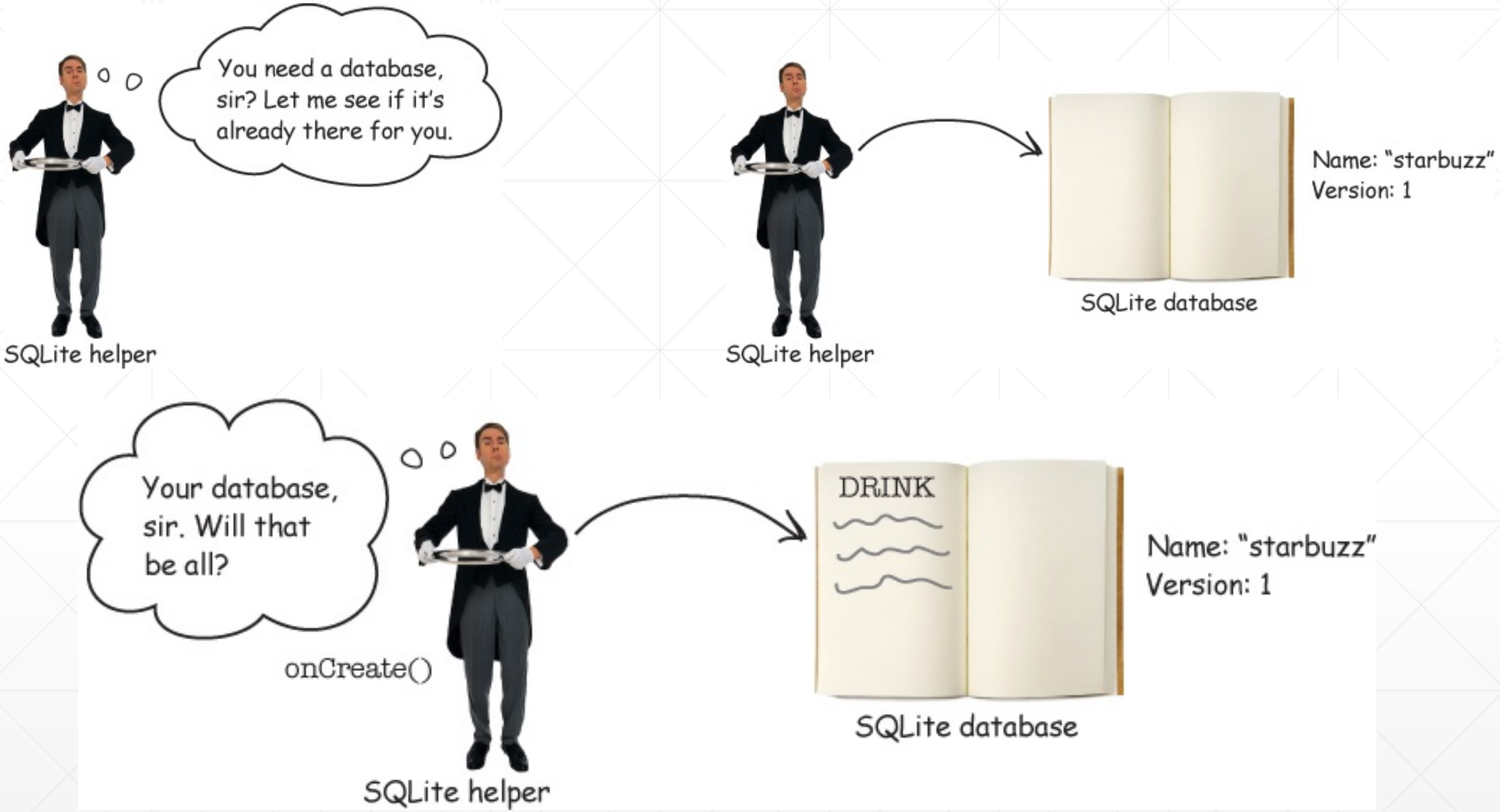
onCreate() gets called when the database first gets created, so we're using it to create the table and insert data.

Create the DRINK table.

Insert each drink in a separate row.

onUpgrade() gets called when the database needs to be upgraded. We'll look at this next.

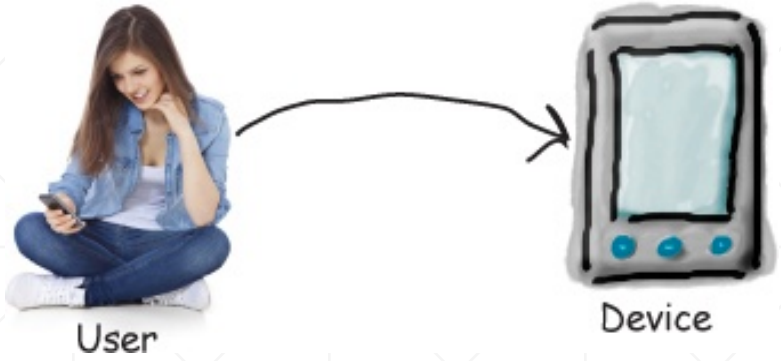
We need to insert several drinks, so we created a separate method to do this.

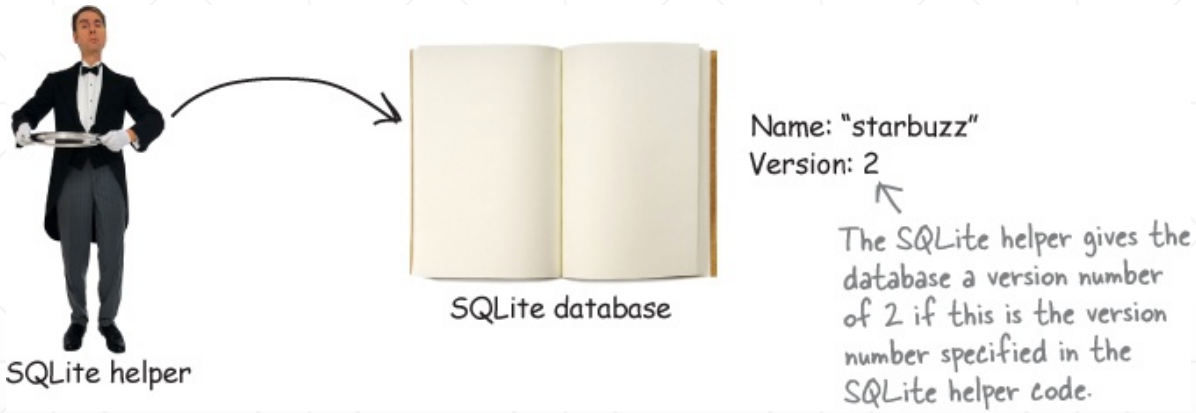

You need a database, sir? Let me see if it's already there for you.
SQLite helper
SQLite helper
Name: "starbuzz"
Version: 1
SQLite database
Your database, sir. Will that be all?
onCreate()
SQLite helper
DRINK
Name: "starbuzz"
Version: 1
SQLite database

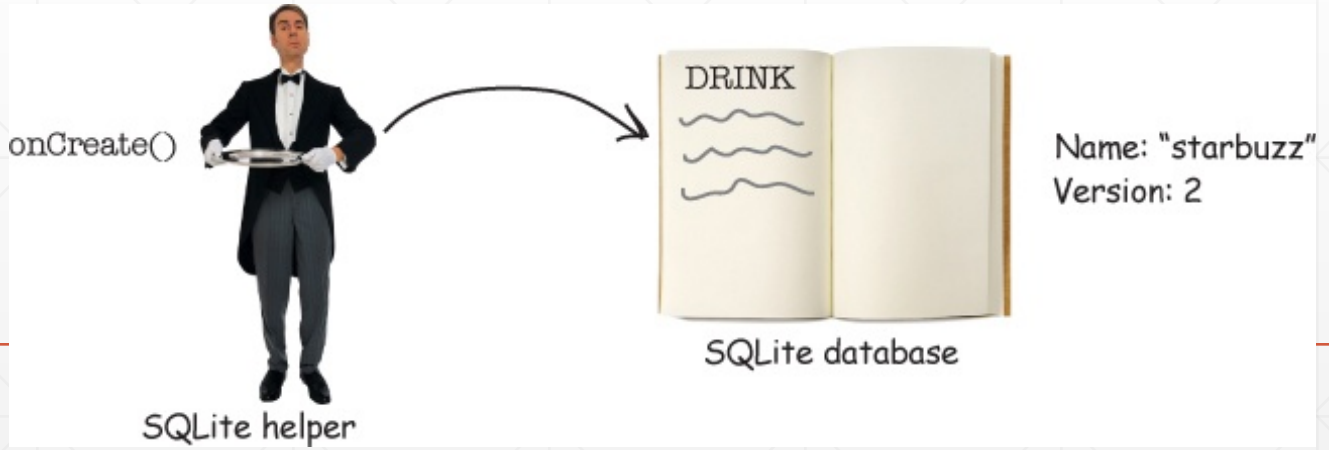
```
...
    private static final String DB_NAME = "starbuzz";
    private static final int DB_VERSION = 1;

    StarbuzzDatabaseHelper(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
}
...
```

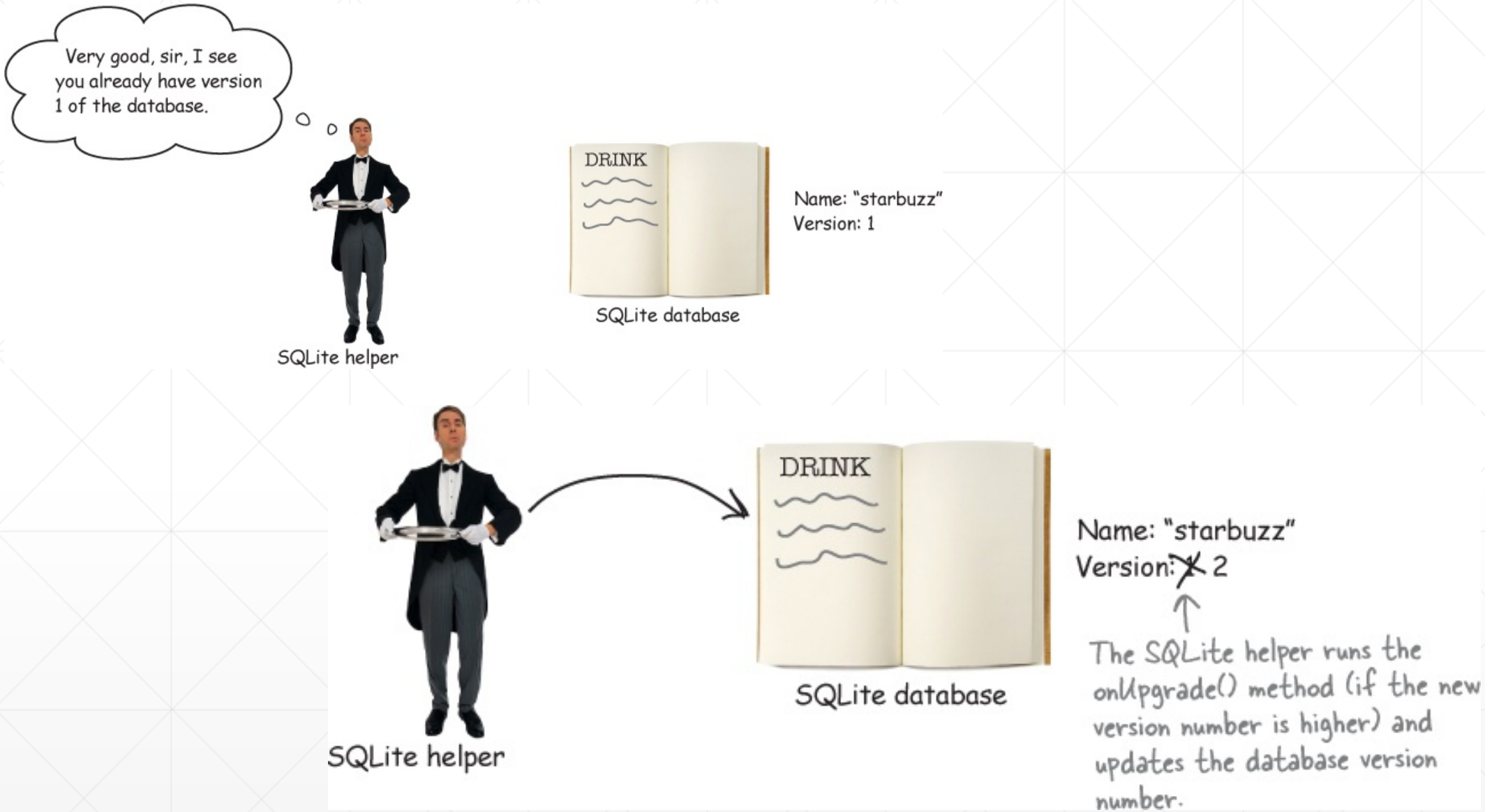
```
...
    private static final int DB_VERSION = 2;
...
```

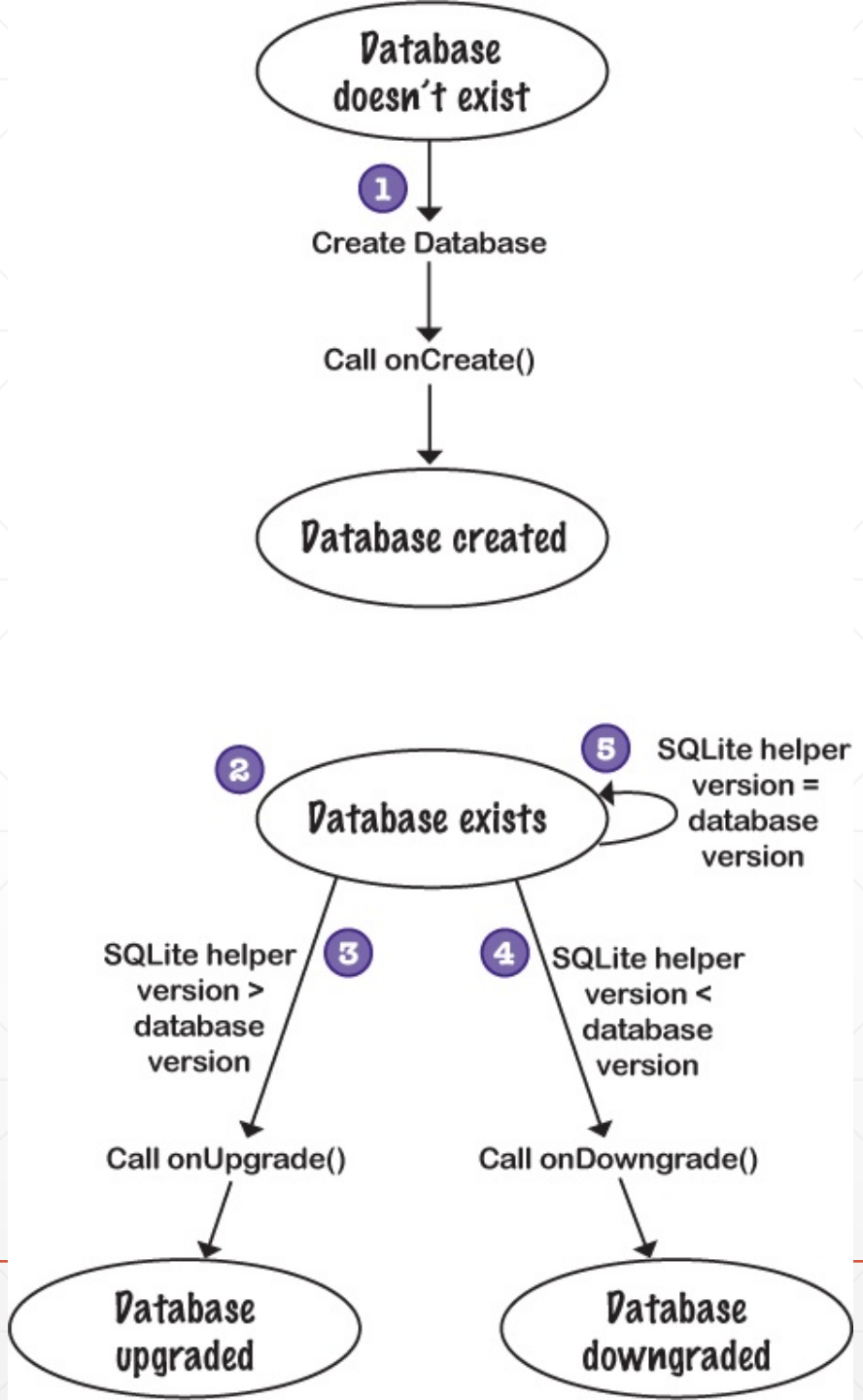
Here we're increasing the version number, so the database will get upgraded.


User
Device


SQLite helper
SQLite database
Name: "starbuzz"
Version: 2
The SQLite helper gives the database a version number of 2 if this is the version number specified in the SQLite helper code.


onCreate()
SQLite helper
DRINK
SQLite database
Name: "starbuzz"
Version: 2


Very good, sir, I see you already have version 1 of the database.
SQLite helper
DRINK
SQLite database
Name: "starbuzz"
Version: 1
SQLite helper
DRINK
SQLite database
Name: "starbuzz"
Version: ~~1~~ 2
The SQLite helper runs the onUpgrade() method (if the new version number is higher) and updates the database version number.


Database doesn't exist
1
Create Database
Call onCreate()
Database created
2
Database exists
5
SQLite helper version = database version
SQLite helper version > database version
3
Call onUpgrade()
Database upgraded
4
SQLite helper version < database version
Call onDowngrade()
Database downgraded

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    //Your code goes here
}
```

The current version of the database

The new version described in the SQlite helper code

Remember, to upgrade the database, the new version must be higher than the old version.

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    if (oldVersion == 1) {
        //Code to run if the database version is 1
    }
}
```

This code will only run if the user's database is at version 1.

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    if (oldVersion == 1) {
        //Code to run if the database version is 1
    }
    if (oldVersion < 3) {
        //Code to run if the database version is 1 or 2
    }
}
```

This code will only run if the user's database is at version 1.

This code will run if the user's database is at version 1 or 2. If the user has version 1 of the database, it will run both sets of code.

```
@Override
public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    //Your code goes here
}
```

To downgrade the database, the new version must be lower than the old version.

```
@Override
public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    if (oldVersion == 3) {
        //Code to run if the database version is 3
    }
}
```

This code will run if the user has version 3 of the database, but you want to downgrade it to a lower version.

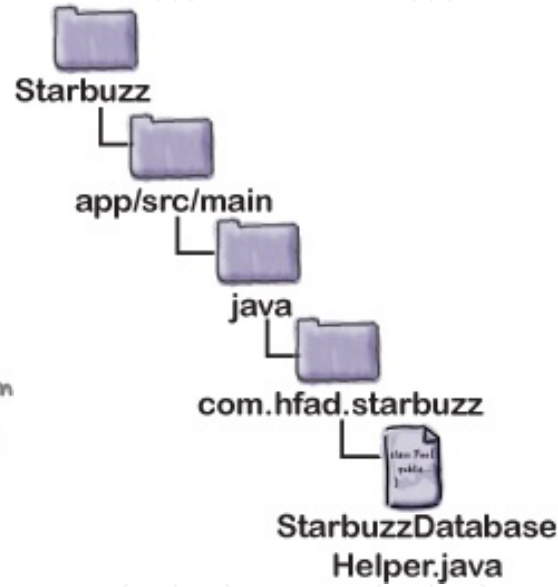
```
...
@Override
public void onCreate(SQLiteDatabase db){
    updateMyDatabase(db, 0, DB_VERSION);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    updateMyDatabase(db, oldVersion, newVersion);
}

private void updateMyDatabase(SQLiteDatabase db, int oldVersion, int newVersion) {
    if (oldVersion < 1) {
        db.execSQL("CREATE TABLE DRINK (_id INTEGER PRIMARY KEY AUTOINCREMENT, "
                + "NAME TEXT, "
                + "DESCRIPTION TEXT, "
                + "IMAGE_RESOURCE_ID INTEGER);");
        insertDrink(db, "Latte", "Espresso and steamed milk", R.drawable.latte);
        insertDrink(db, "Cappuccino", "Espresso, hot milk and steamed-milk foam",
                R.drawable.cappuccino);
        insertDrink(db, "Filter", "Our best drip coffee", R.drawable.filter);
    }
    if (oldVersion < 2) {
        //Code to add the extra column
    }
}
...
```

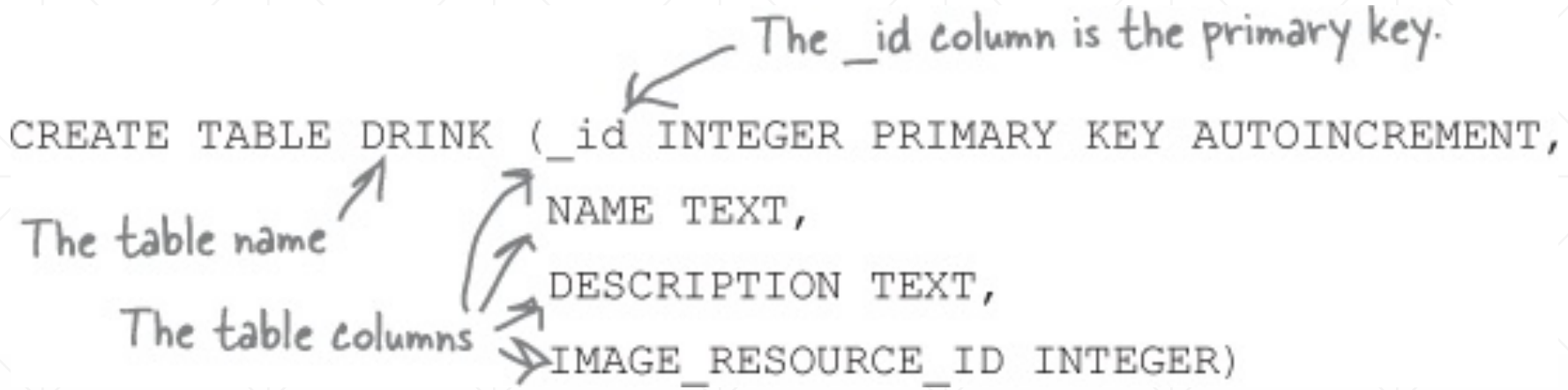
Rather than create the DRINK table here, we'll get our updateMyDatabase() method to do it.

Call the updateMyDatabase() method from onUpgrade(), passing along the parameters.

This is the code we had in our onCreate() method.

This code will run if the user already has version 1 of the database.




The _id column is the primary key.
CREATE TABLE DRINK (_id INTEGER PRIMARY KEY AUTOINCREMENT,
NAME TEXT,
DESCRIPTION TEXT,
IMAGE_RESOURCE_ID INTEGER)
The table name
The table columns

```
ALTER TABLE DRINK
ADD COLUMN FAVORITE NUMERIC
```

The table name

The column you want to add

```
ALTER TABLE DRINK
RENAME TO FOO
```

The current table name

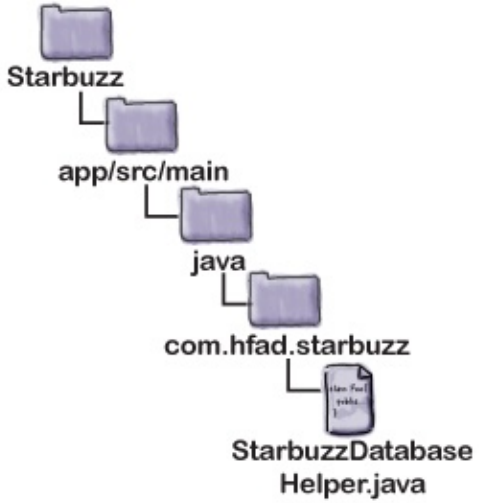
The new name of the table

```
DROP TABLE DRINK
```

The name of the table you want to remove

```
SQLiteDatabase.execSQL(String sql);
```

```
db.execSQL("ALTER TABLE DRINK ADD COLUMN FAVORITE NUMERIC;");
```




```
package com.hfad.starbuzz;

import android.content.ContentValues;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

class StarbuzzDatabaseHelper extends SQLiteOpenHelper{

    private static final String DB_NAME = "starbuzz"; // the name of our database
    private static final int DB_VERSION = 2; // the version of the database

    StarbuzzDatabaseHelper(Context context){
        super(context, DB_NAME, null, DB_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db){
        updateMyDatabase(db, 0, DB_VERSION);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        updateMyDatabase(db, oldVersion, newVersion);
    }
```

Changing the version number to a larger integer enables the SQLite helper to spot that you want to upgrade the database.

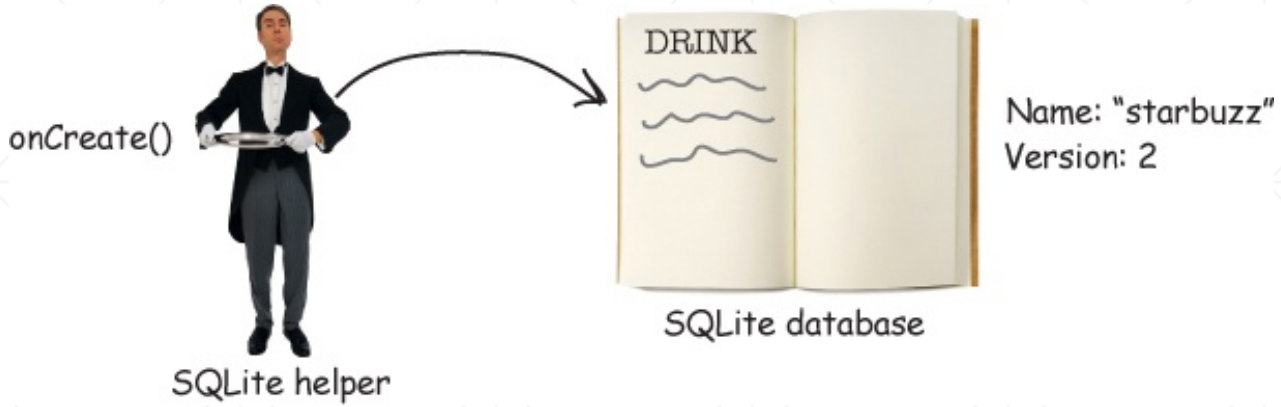
We'll put the code from the onCreate() method in the updateMyDatabase() method.

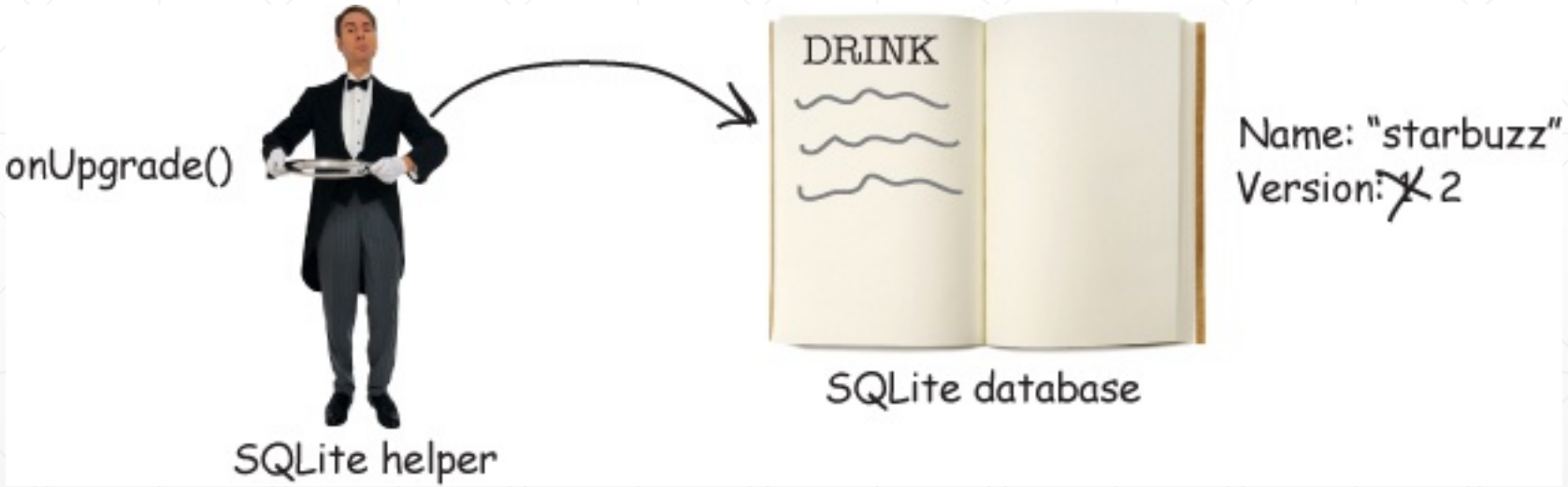
The code to upgrade the database is in our updateMyDatabase() method.

```
    private void updateMyDatabase(SQLiteDatabase db, int oldVersion, int newVersion) {
        if (oldVersion < 1) {
            db.execSQL("CREATE TABLE DRINK (_id INTEGER PRIMARY KEY AUTOINCREMENT, "
                       + "NAME TEXT, "
                       + "DESCRIPTION TEXT, "
                       + "IMAGE_RESOURCE_ID INTEGER);");
            insertDrink(db, "Latte", "Espresso and steamed milk", R.drawable.latte);
            insertDrink(db, "Cappuccino", "Espresso, hot milk and steamed-milk foam",
                        R.drawable.cappuccino);
            insertDrink(db, "Filter", "Our best drip coffee", R.drawable.filter);
        }
        if (oldVersion < 2) {
            db.execSQL("ALTER TABLE DRINK ADD COLUMN FAVORITE NUMERIC;");
        }
    }

    private static void insertDrink(SQLiteDatabase db, String name,
                                    String description, int resourceId) {
        ContentValues drinkValues = new ContentValues();
        drinkValues.put("NAME", name);
        drinkValues.put("DESCRIPTION", description);
        drinkValues.put("IMAGE_RESOURCE_ID", resourceId);
        db.insert("DRINK", null, drinkValues);
    }
}
```

Add a numeric FAVORITE column to the DRINK table.


You need a database, sir? Let me see if it's already there for you.
SQLite helper


onCreate()
DRINK
Name: "starbuzz"
Version: 2
SQLite database
SQLite helper


onUpgrade()
DRINK
Name: "starbuzz"
Version: 2
SQLite database
SQLite helper

Bullet Points

- Android uses SQLite as its backend database to persist data.
- The SQLiteDatabase class gives you access to the SQLite database.
- A SQLite helper lets you create and manage SQLite databases. You create a SQLite helper by extending the SQLiteOpenHelper class.
- You must implement the SQLiteOpenHelper onCreate() and onUpgrade() methods.
- The database gets created the first time it needs to be accessed. You need to give the database a name and version number, starting at 1. If you don’t give the database a name, it will just get created in memory.
- The onCreate() method gets called when the database first gets created.
- The onUpgrade() method gets called when the database needs to be upgraded.

Bullet Points

- Execute SQL using the SQLiteDatabase execSQL(String) method.
- Add records to tables using the insert() method.
- Update records using the update() method.
- Remove records from tables using the delete() method.

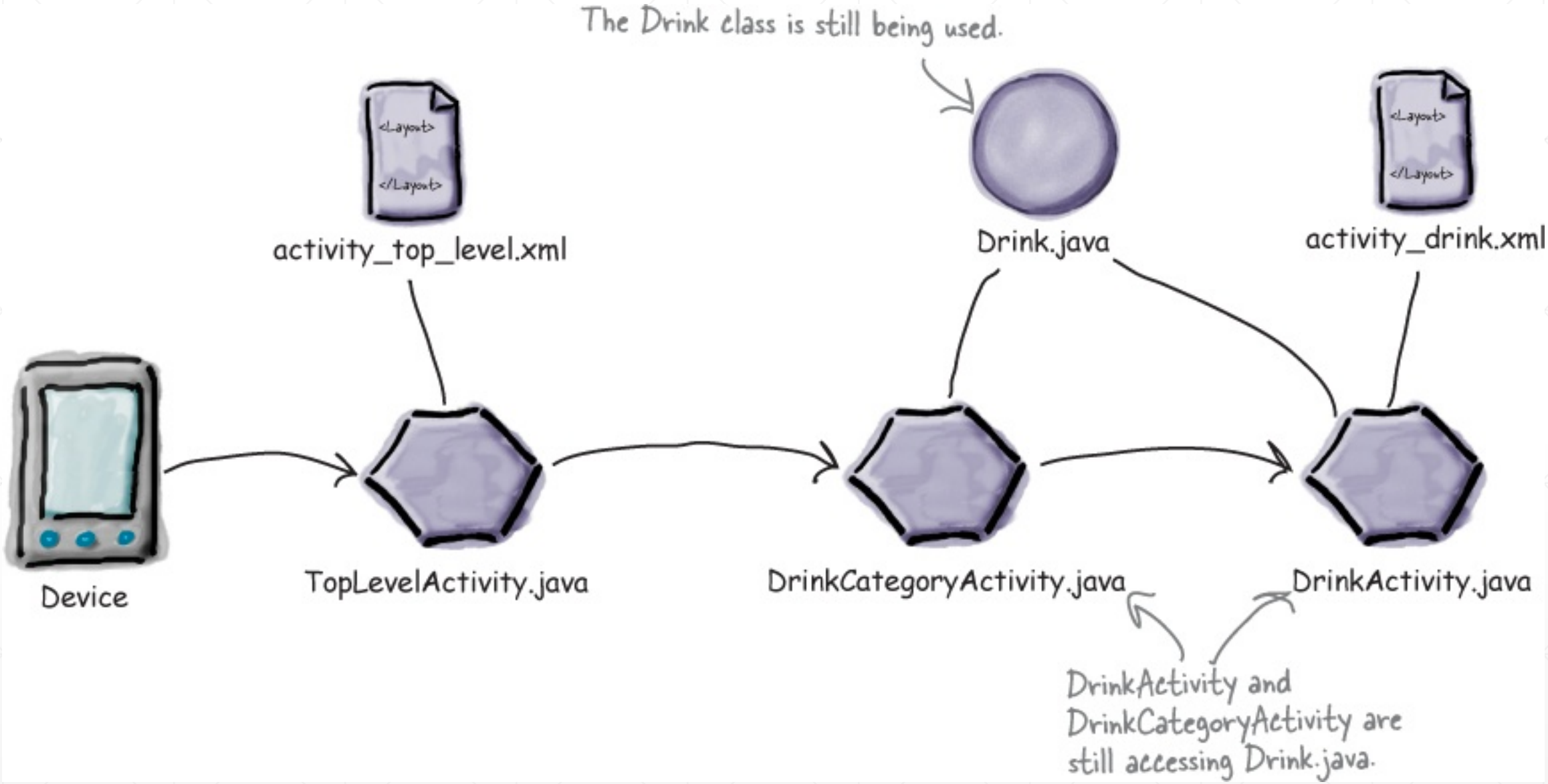
Bullet Points

- A Cursor lets you read from and write to the database.
- You create a cursor by calling the SQLiteDatabase query() method. Behind the scenes, this builds a SQL SELECT statement.
- The getWritableDatabase() method returns a SQLiteDatabase object that allows you to read from and write to the database.
- The getReadableDatabase() returns a SQLiteDatabase object. This gives you read-only access to the database. It may also allow you to read from and write to the database, but this isn't guaranteed.
- Navigate through a cursor using the moveTo*() methods.
- Get values from a cursor using the get*() methods.

Bullet Points

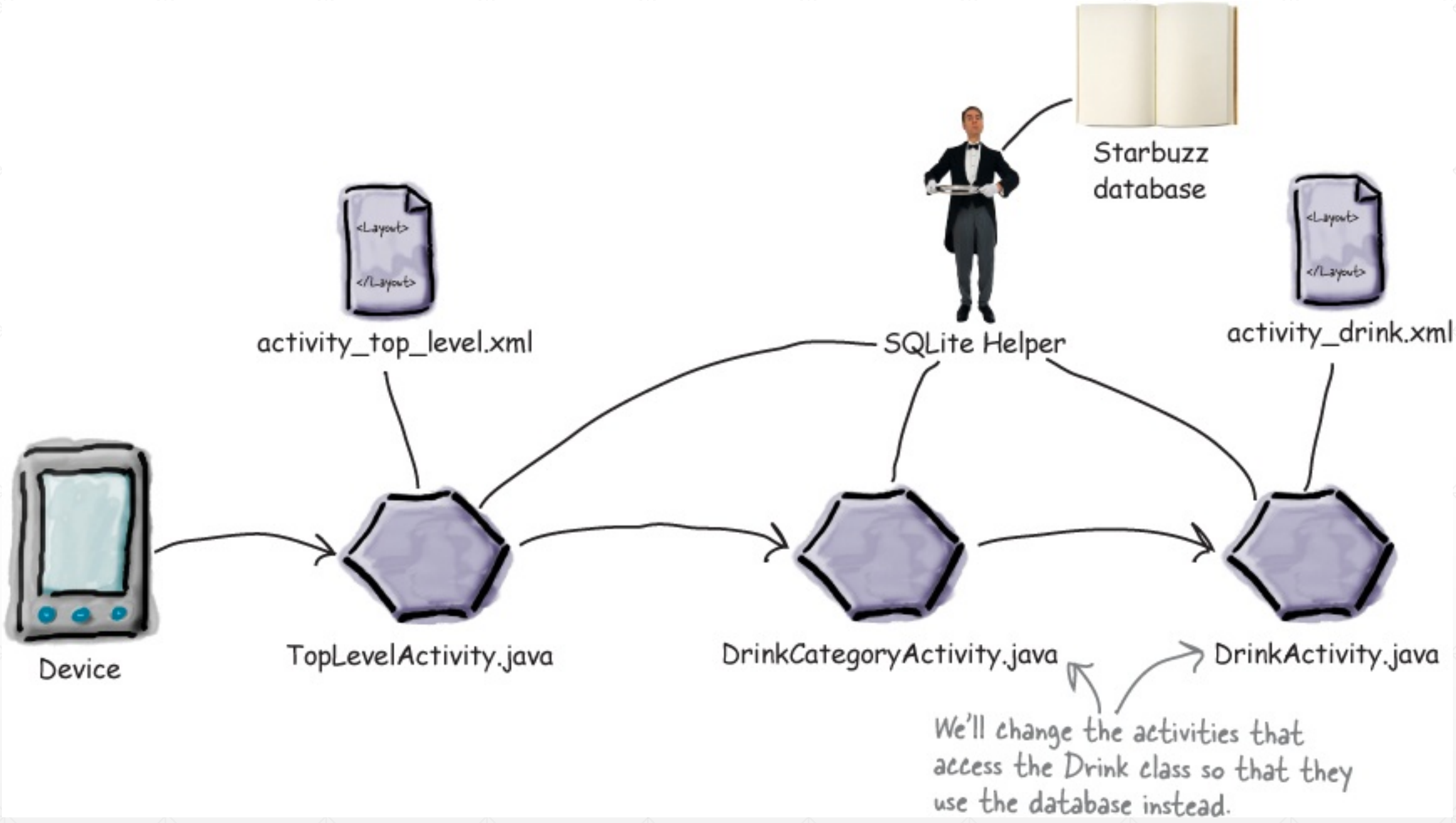
- Close cursors and database connections after you've finished with them.
- A CursorAdapter is an adapter that works with cursors.
  Use SimpleCursorAdapter to populate a ListView with the values returned by a cursor.
- Design your app so that you put useful content in your top-level activity.
- The CursorAdapter changeCursor() method replaces the cursor currently used by a cursor adapter to a new cursor that you provide. It then closes the old cursor.
- Run your database code in a background thread using AsyncTask.

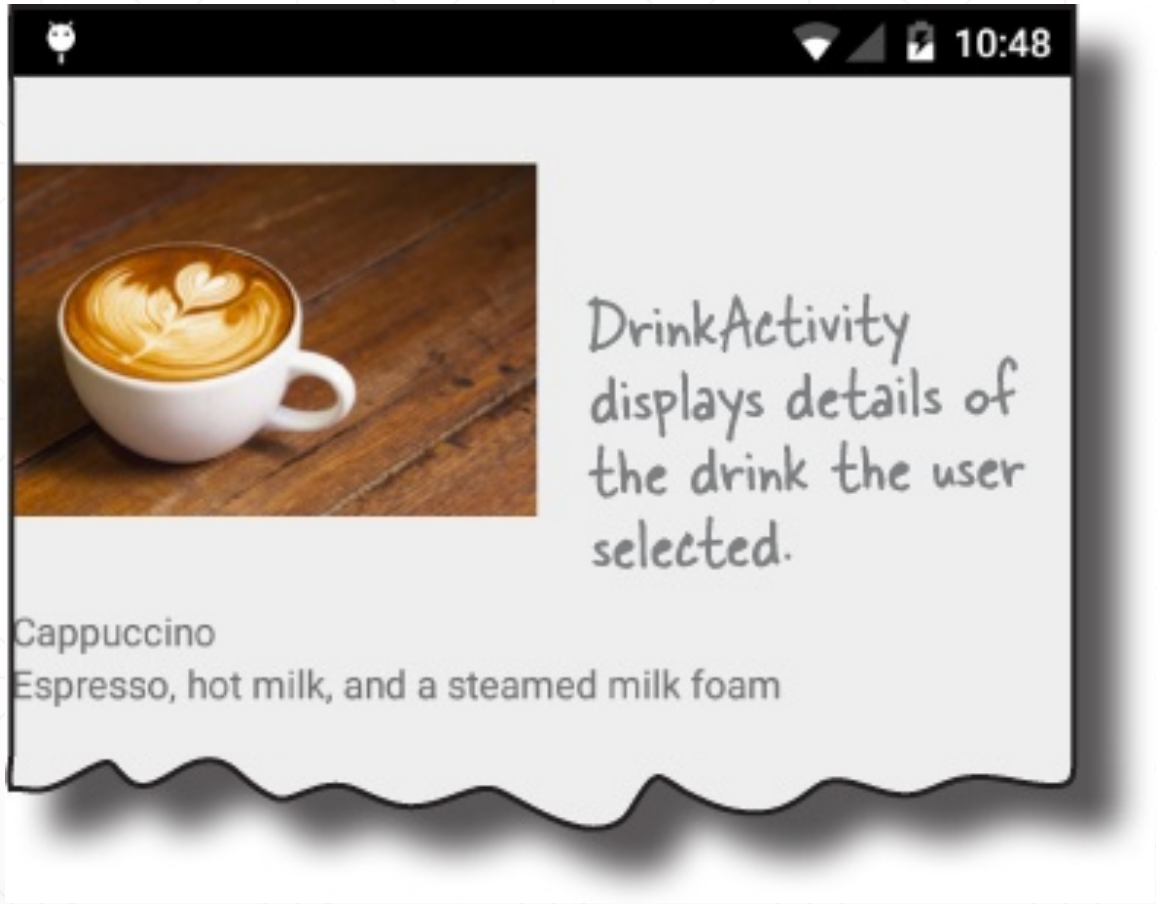
Chap 12 – Connecting to Databases


The Drink class is still being used.
<Layout>
</Layout>
activity_top_level.xml
Drink.java
<Layout>
</Layout>
activity_drink.xml
Device
TopLevelActivity.java
DrinkCategoryActivity.java
DrinkActivity.java
DrinkActivity and
DrinkCategoryActivity are
still accessing Drink.java.

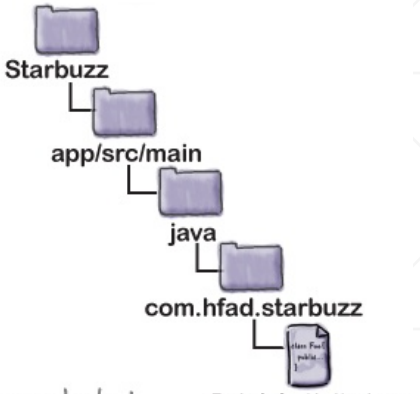
App changes

- Update the Drink code in DrinkActivity.
- Update the Drink code in DrinkCategoryActivity.
- Let users choose their favorite drinks.


Starbuzz
database
<Layout>
</Layout>
activity_top_level.xml
SQLite Helper
<Layout>
</Layout>
activity_drink.xml
Device
TopLevelActivity.java
DrinkCategoryActivity.java
DrinkActivity.java
We'll change the activities that
access the Drink class so that they
use the database instead.


10:48
DrinkActivity displays details of the drink the user selected.
Cappuccino
Espresso, hot milk, and a steamed milk foam

```
package com.hfad.starbuzz;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ImageView;
import android.widget.TextView;

public class DrinkActivity extends Activity {

    public static final String EXTRA_DRINKNO = "drinkNo";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_drink);

        //Get the drink from the intent
        int drinkNo = (Integer)getIntent().getExtras().get(EXTRA_DRINKNO);
        Drink drink = Drink.drinks[drinkNo];

        //Populate the drink image
        ImageView photo = (ImageView)findViewById(R.id.photo);
        photo.setImageResource(drink.getImageResourceId());
        photo.setContentDescription(drink.getName());

        //Populate the drink name
        TextView name = (TextView)findViewById(R.id.name);
        name.setText(drink.getName());

        //Populate the drink description
        TextView description = (TextView)findViewById(R.id.description);
        description.setText(drink.getDescription());
    }
}
```



This is the drink the user selected.

Use the drink number from the intent to get the drink from the Drink class. We'll need to change this so the drink comes from the database.

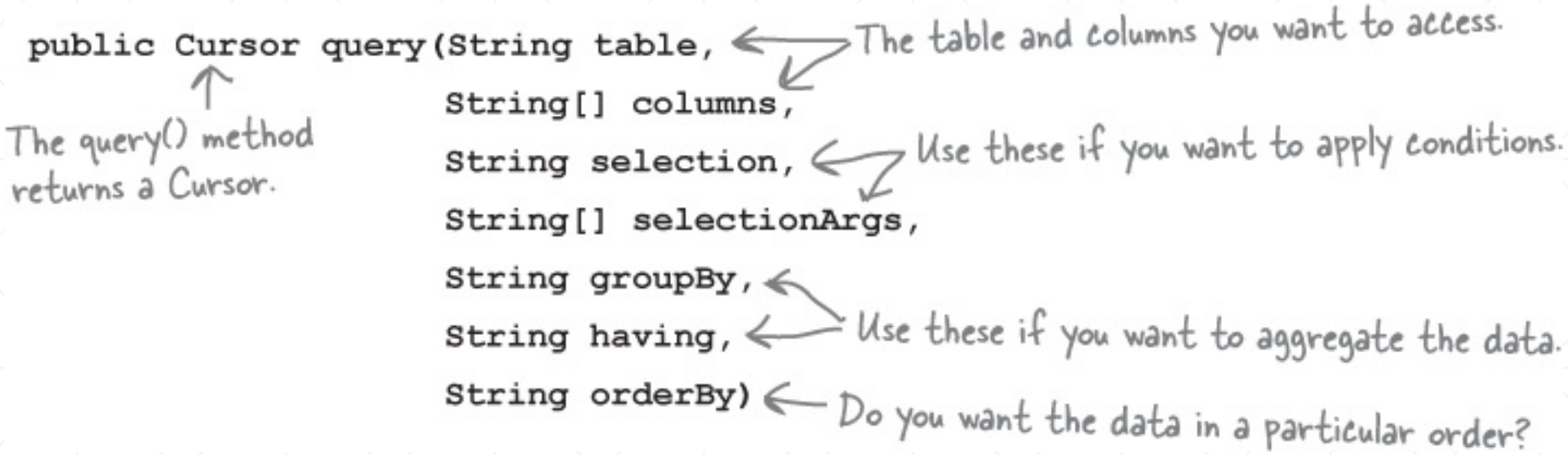
We need to populate the views in the layout with values from the database, not from the Drink class.

Return the data from the NAME and DESCRIPTION columns in the DRINK table.

_id	NAME	DESCRIPTION	IMAGE_RESOURCE_ID	FAVORITE
1	“Latte”	"Espresso and steamed milk"	54543543	0
2	“Cappuccino”	"Espresso, hot milk and steamed-milk foam"	654334453	0
3	“Filter”	"Our best drip coffee"	44324234	0

Return data where _id is 1.

_id	NAME	DESCRIPTION	IMAGE_RESOURCE_ID	FAVORITE
1	“Latte”	"Espresso and steamed milk"	54543543	0
2	“Cappuccino”	"Espresso, hot milk and steamed-milk foam"	654334453	0
3	“Filter”	"Our best drip coffee"	44324234	0




```
Select
        NAME, DESCRIPTION, IMAGE_RESOURCE_ID
From
        DRINK
Where
        _id = 1;
```

Put each column you want back as a separate value in a String array.

```
Cursor cursor = db.query("DRINK",
                         new String[] {"NAME", "DESCRIPTION"},
                         null, null, null, null, null);
```

This query only uses the first two parameters, hence the null values.

The query returns all the data from the NAME and DESCRIPTION columns in the DRINK table.

NAME	DESCRIPTION
“Latte”	"Espresso and steamed milk"
“Cappuccino”	"Espresso, hot milk and steamed-milk foam"
“Filter”	"Our best drip coffee"

```
Select
        NAME, DESCRIPTION
From
        DRINK;
```

```
Cursor cursor = db.query("DRINK",
                         new String[] {"NAME", "DESCRIPTION"},
                         "NAME = ?",
                         new String[] {"Latte"},
                         null, null, null);
```

This means "where the contents of the NAME column is 'Latte'".

The query returns all the data from the NAME and DESCRIPTION columns in the DRINK table where the value of the NAME column is "Latte".

NAME	DESCRIPTION
"Latte"	"Espresso and steamed milk"

Select NAME, DESCRIPTION From DRINK Where NAME = ”Latte”;

```
Cursor cursor = db.query("DRINK",
                         new String[] {"NAME", "DESCRIPTION"},
                         "NAME = ? OR DESCRIPTION = ?",
                         new String[] {"Latte", "Our best drip coffee"},
                         null, null, null);
```

This means "where NAME is 'Latte' or DESCRIPTION is "'Our best drip coffee'".

The query returns all the data from the NAME and DESCRIPTION columns in the DRINK table where the value of the NAME column is "Latte" or the value of the DESCRIPTION column is "Our best drip coffee".

NAME	DESCRIPTION
"Latte"	"Espresso and steamed milk"
"Filter"	"Our best drip coffee"

Select NAME, DESCRIPTION From DRINK
Where NAME = ”Latte” or DESCRIPTION = “Our best drip coffee”

```
Cursor cursor = db.query("Drink",
                         new String[] {"NAME", "DESCRIPTION"},
                         "_id = ?",
                         new String[] {Integer.toString(1)},
                         null, null, null);
```

Convert the int 1 to a String value.

The query returns all the data from the NAME and DESCRIPTION columns in the DRINK table where the value of the _id column is 1.

_id	NAME	DESCRIPTION
1	"Latte"	"Espresso and steamed milk"

```
Select
        NAME, DESCRIPTION
From
        DRINK
Where
        _id = 1;
```

```
Cursor cursor = db.query("DRINK",
        new String[] {"_id", "NAME", "FAVORITE"},
        null, null, null, null,
        "NAME ASC");
```

Order by NAME in ascending order.

NAME	FAVORITE
"Cappuccino"	0
"Filter"	0
"Latte"	1

```
Select
        NAME, FAVORITE
From
        DRINK
Order By
        NAME ASC;
```

```
Cursor cursor = db.query("DRINK",
        new String[] {"_id", "NAME", "FAVORITE"},
        null, null, null, null,
        "FAVORITE DESC, NAME");
```

Order by FAVORITE in descending order, then NAME in ascending order.

NAME	FAVORITE
"Latte"	1
"Cappuccino"	0
"Filter"	0

```
Select
        NAME, FAVORITE
From
        DRINK
Order By
        FAVORITE DESC, NAME ASC;
```

SQL FUNCTION	DESCRIPTION
AVG()	The average value
COUNT()	The number of rows
SUM()	The sum
MAX()	The largest value
MIN()	The smallest value

This returns the number of drinks in a column labeled "count"

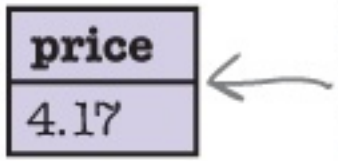
```
Cursor cursor = db.query("DRINK",
                         new String[] {"COUNT(_id) AS count"},
                         null, null, null, null, null);
```

The query returns the number of rows in the DRINK table.

count
3

Select COUNT(_id) as count From DRINK;

```
Cursor cursor = db.query("DRINK",
                         new String[] {"AVG(PRICE) AS price"},
                         null, null, null, null, null);
```



Our DRINK table doesn't contain a PRICE column, but if it did, we could use it to get the average drink price.

```
Cursor cursor = db.query("DRINK",
                         new String[] {"FAVORITE", "COUNT(_id) AS count"},
                         null, null,
                         "FAVORITE",
                         null, null);
```

Group by the FAVORITE column.

Return the FAVORITE column and the number of drinks.

```
Select FAVORITE, count(_id) as count From DRINK Group by FAVORITE;
```

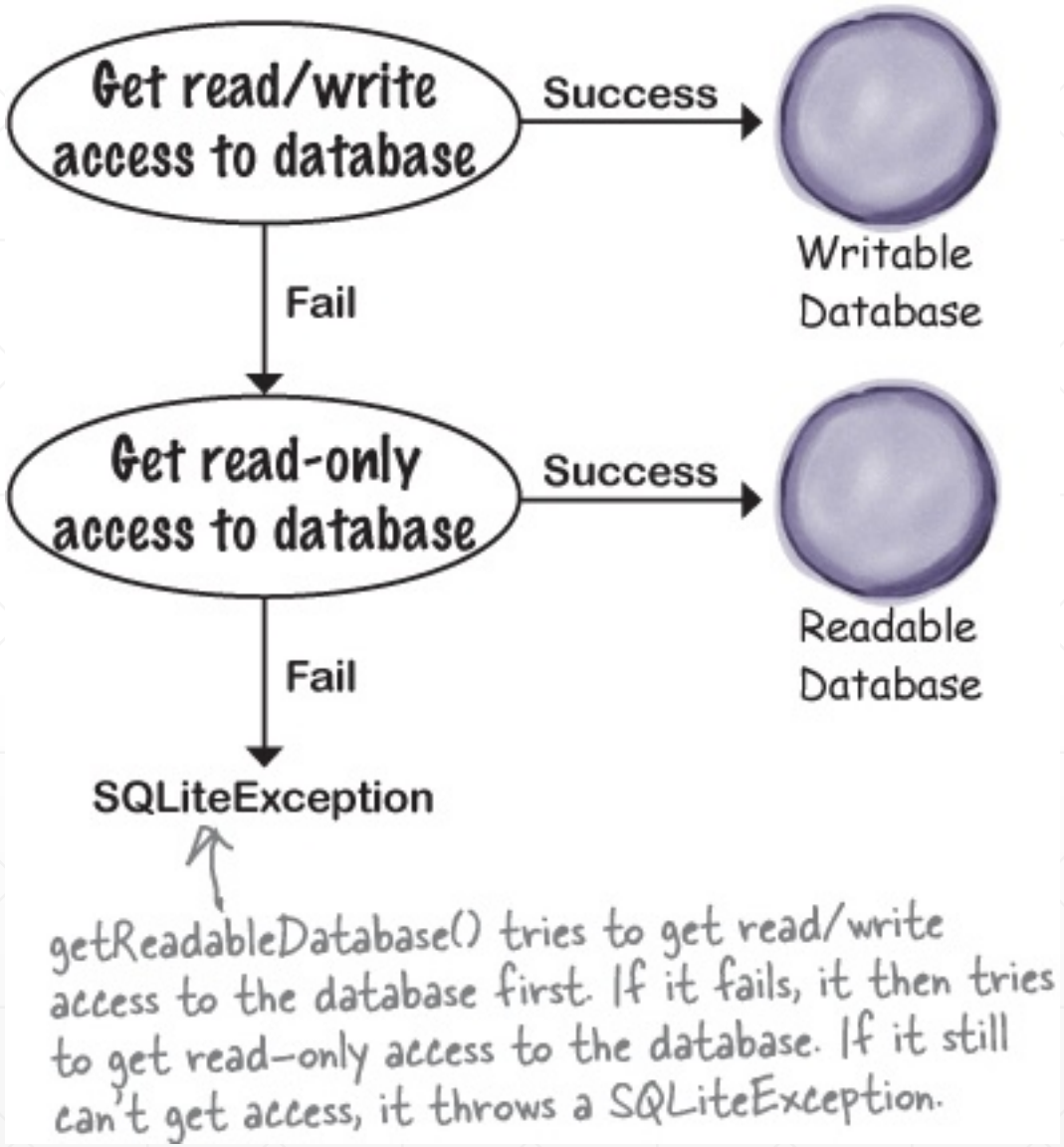
_id	NAME	DESCRIPTION	IMAGE_RESOURCE_ID	FAVORITE
1	“Latte”	"Espresso and steamed milk"	54543543	1
2	“Cappuccino”	"Espresso, hot milk and steamed-milk foam"	654334453	0
3	“Filter”	"Our best drip coffee"	44324234	0

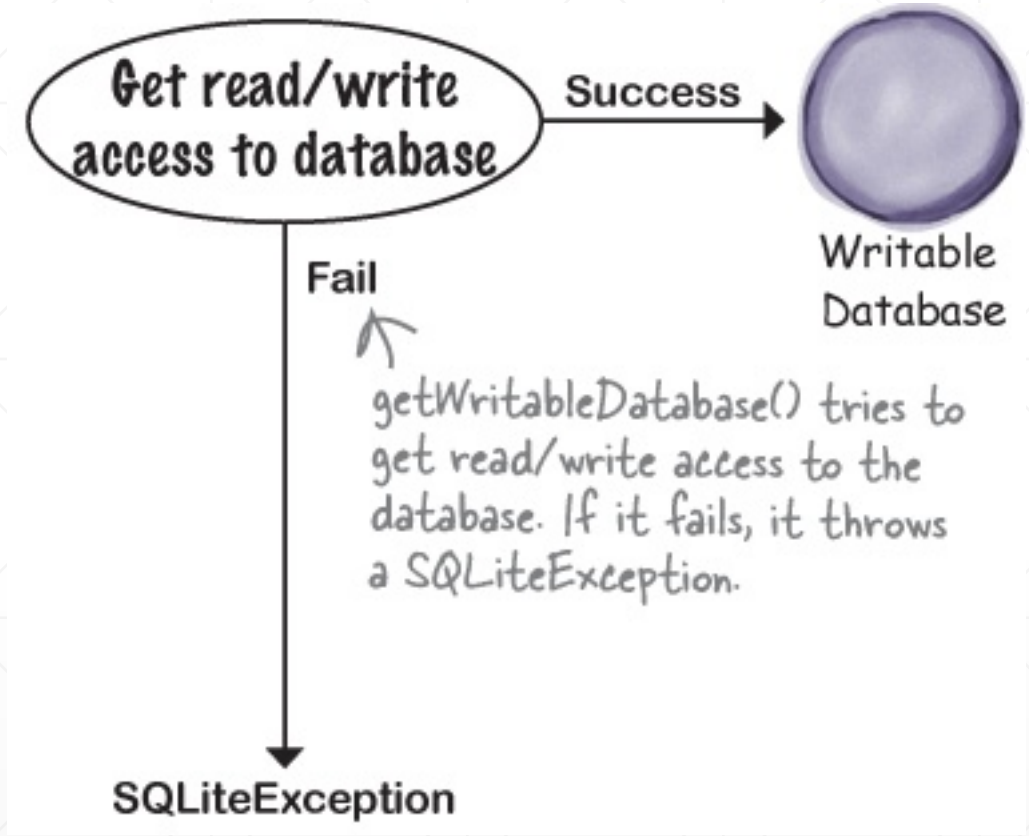
There's 1 drink where FAVORITE has a value of 1, and 2 drinks where FAVORITE has a value of 0.

FAVORITE	count
1	1
0	2

```
SQLiteOpenHelper starbuzzDatabaseHelper = new StarbuzzDatabaseHelper(this);
SQLiteDatabase db = starbuzzDatabaseHelper.getReadableDatabase();
```

```
SQLiteOpenHelper starbuzzDatabaseHelper = new StarbuzzDatabaseHelper(this);
SQLiteDatabase db = starbuzzDatabaseHelper.getWritableDatabase();
```


Get read/write access to database
Success
Writable Database
Fail
Get read-only access to database
Success
Readable Database
Fail
SQLiteException
getReadableDatabase() tries to get read/write access to the database first. If it fails, it then tries to get read-only access to the database. If it still can't get access, it throws a SQLiteException.


Get read/write access to database
Success
Writable Database
Fail
getWritableDatabase() tries to get read/write access to the database. If it fails, it throws a SQLiteException.
SQLiteException

```
try {
    SQLiteOpenHelper starbuzzDatabaseHelper = new StarbuzzDatabaseHelper(this);
    SQLiteDatabase db = starbuzzDatabaseHelper.getReadableDatabase();
    Cursor cursor = db.query("DRINK",
                             new String[] {"NAME", "DESCRIPTION", "IMAGE_RESOURCE_ID"},
                             "_id = ?",
                             new String[] {Integer.toString(drinkNo)},
                             null, null,null);

    //Code to do something with the cursor

} catch(SQLiteException e) {
    Toast toast = Toast.makeText(this, "Database unavailable", Toast.LENGTH_SHORT);
    toast.show();
}
```

We don't need to write to the database so we're using getReadableDatabase().

The cursor contains a single record as the _id column contains unique records.

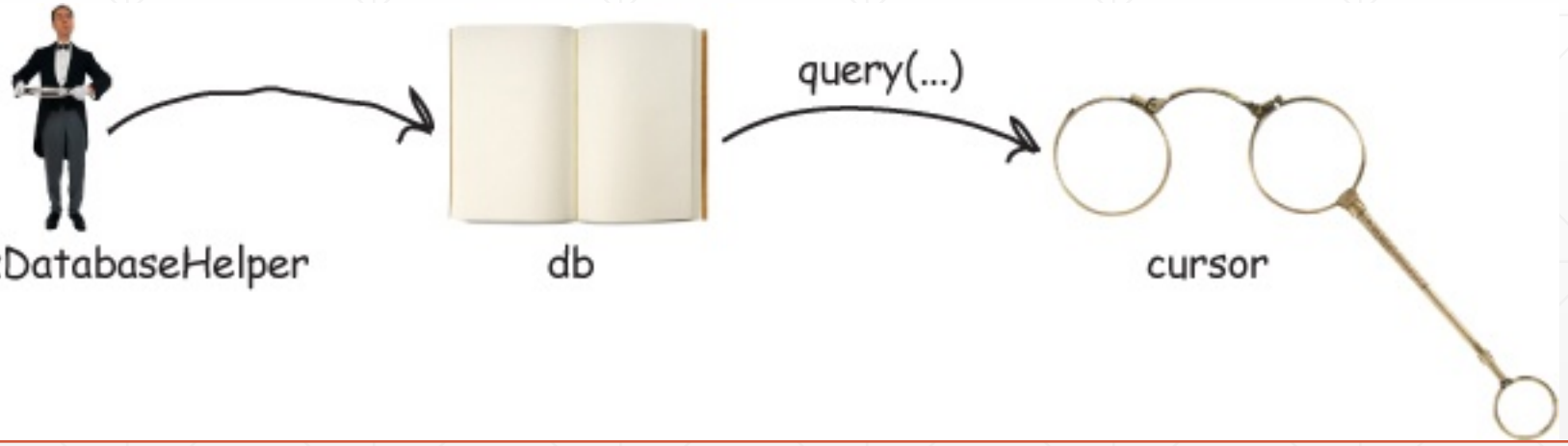
We have the cursor, but we still need to do something with it.

Display a message if the database is unavailable.

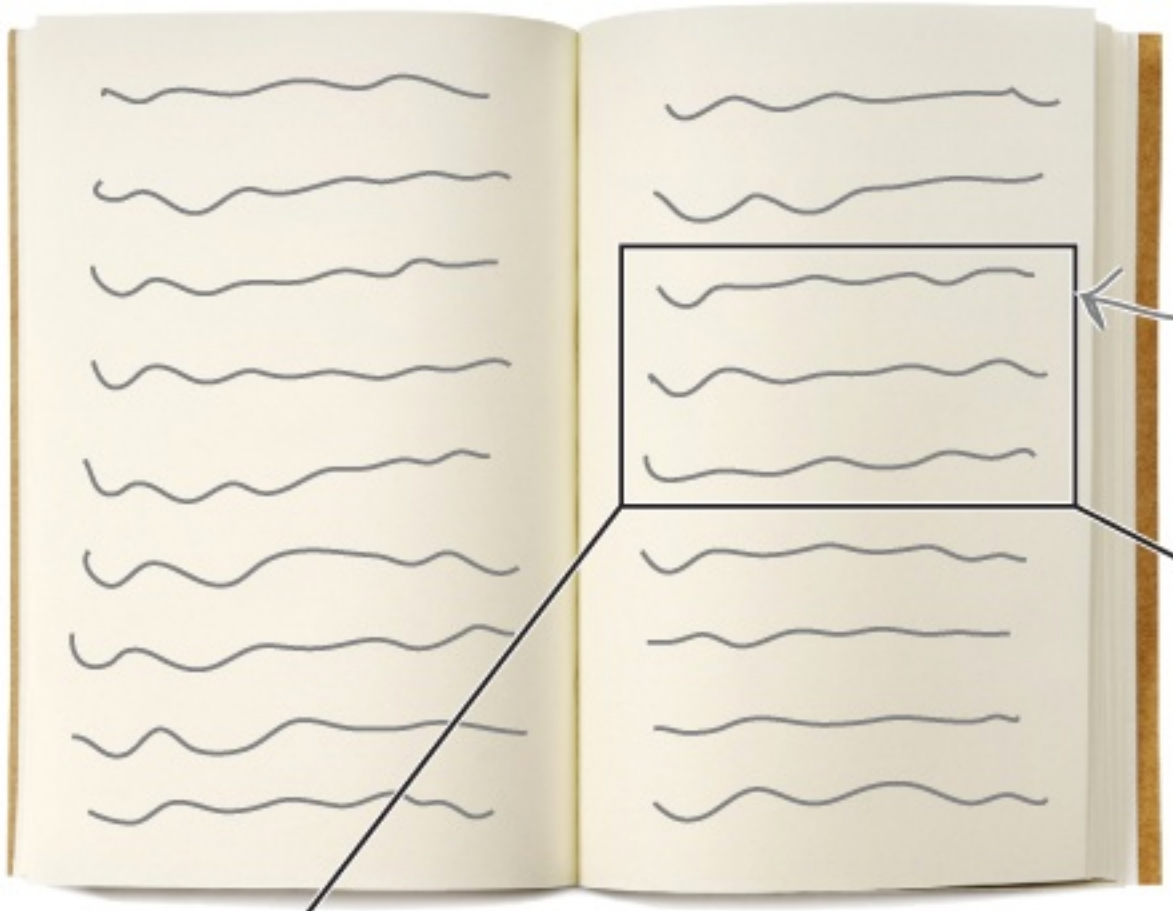
Select NAME, DESCRIPTION, IMAGE_RESOURCE_ID From Name Where _id = drinkNo

starbuzzDatabaseHelper

starbuzzDatabaseHelper db



starbuzzDatabaseHelper db cursor



You specify what records you want by creating a query on the database.

The cursor contains the records described by the query.

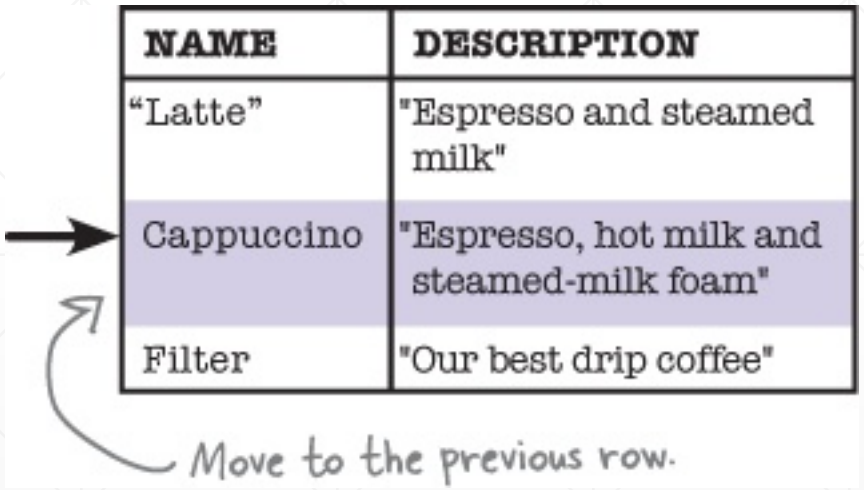
_id	NAME	DESCRIPTION	IMAGE_RESOURCE_ID
1	"Latte"	"Espresso and steamed milk"	54543543
2	"Cappuccino"	"Espresso, hot milk and steamed-milk foam"	654334453
3	"Filter"	"Our best drip coffee"	44324234

We need to move to a record in the cursor to read values from it.

```
if (cursor.moveToFirst()) {
    //Do something
};
```

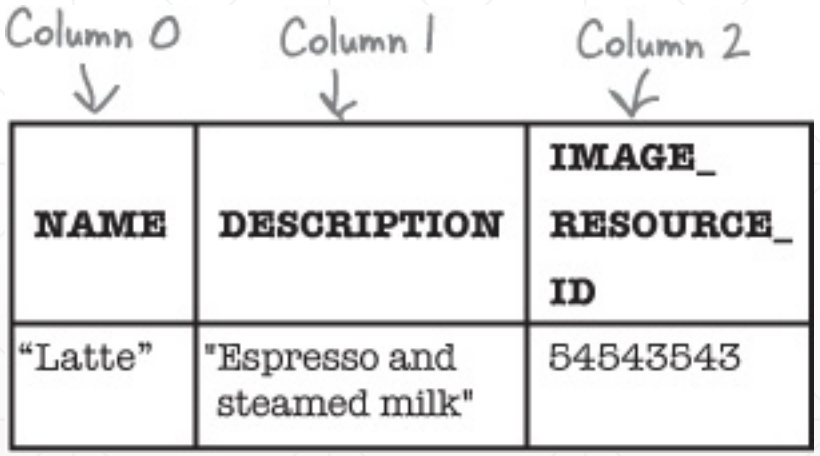
```
if (cursor.moveToLast()) {
    //Do something
};
```

```
if (cursor.moveToPrevious()) {
    //Do something
};
```



```
if (cursor.moveToNext()) {
    //Do something
};
```

```
Cursor cursor = db.query ("Drink",
                          new String[] {"NAME", "DESCRIPTION", "IMAGE_RESOURCE_ID"},
                          "_id = ?",
                          new String[] {Integer.toString(1)},
                          null, null,null);
```

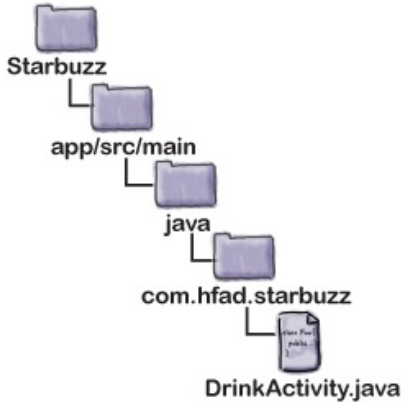


```
String name = cursor.getString(0);
```
← This is the first column in the cursor.

```
cursor.close();
db.close();
```

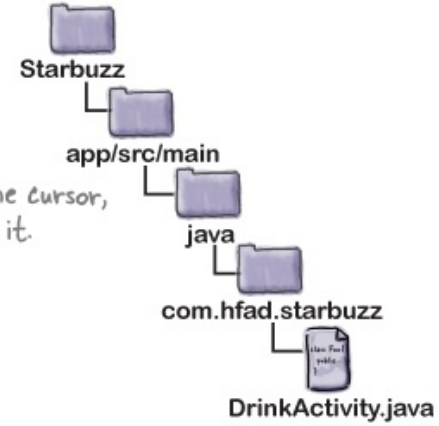
```
package com.hfad.starbuzz;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import android.database.sqlite.SQLiteOpenHelper;

public class DrinkActivity extends Activity {

    public static final String EXTRA_DRINKNO = "drinkNo";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_drink);

        //Get the drink from the intent
        int drinkNo = (Integer)getIntent().getExtras().get(EXTRA_DRINKNO);

        //Create a cursor
        try {
            SQLiteOpenHelper starbuzzDatabaseHelper = new StarbuzzDatabaseHelper(this);
            SQLiteDatabase db = starbuzzDatabaseHelper.getReadableDatabase();
            Cursor cursor = db.query ("DRINK",
                    new String[] {"NAME", "DESCRIPTION", "IMAGE_RESOURCE_ID"},
                    "_id = ?",
                    new String[] {Integer.toString(drinkNo)},
                    null, null,null);
```

We're using these extra classes in the code.



This is the ID of the drink the user chose.

Create a cursor that gets the NAME, DESCRIPTION, and IMAGE_RESOURCE_ID data from the DRINK table where _id matches drinkNo.




```
        //Move to the first record in the Cursor
        if (cursor.moveToFirst()) {

            //Get the drink details from the cursor
            String nameText = cursor.getString(0);
            String descriptionText = cursor.getString(1);
            int photoId = cursor.getInt(2);

            //Populate the drink name
            TextView name = (TextView)findViewById(R.id.name);
            name.setText(nameText);

            //Populate the drink description
            TextView description = (TextView)findViewById(R.id.description);
            description.setText(descriptionText);

            //Populate the drink image
            ImageView photo = (ImageView)findViewById(R.id.photo);
            photo.setImageResource(photoId);
            photo.setContentDescription(nameText);
        }
        cursor.close();
        db.close();
    } catch(SQLiteException e) {
        Toast toast = Toast.makeText(this, "Database unavailable", Toast.LENGTH_SHORT);
        toast.show();
    }
}
```

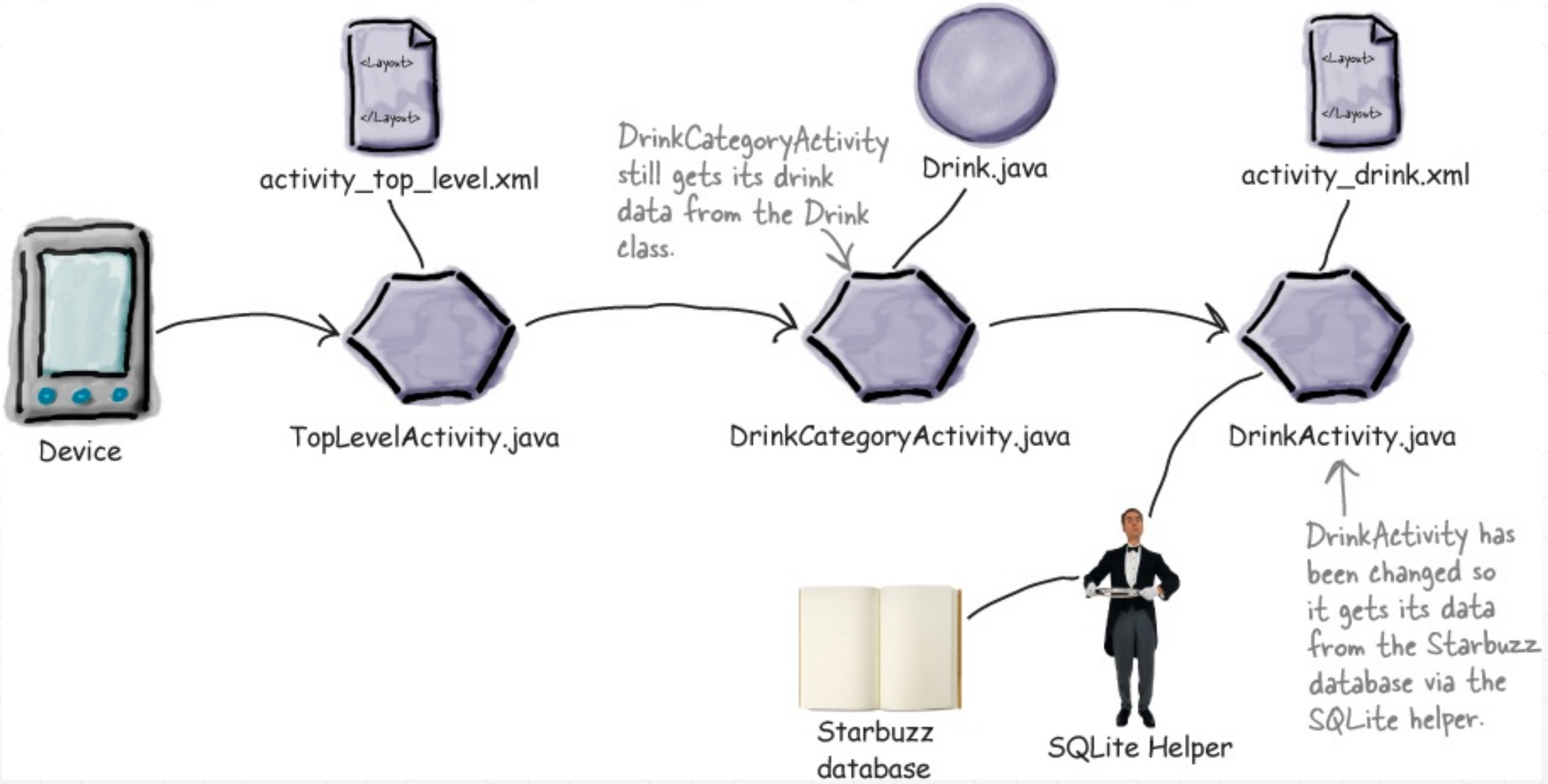
There's only one record in the cursor, but we still need to move to it.

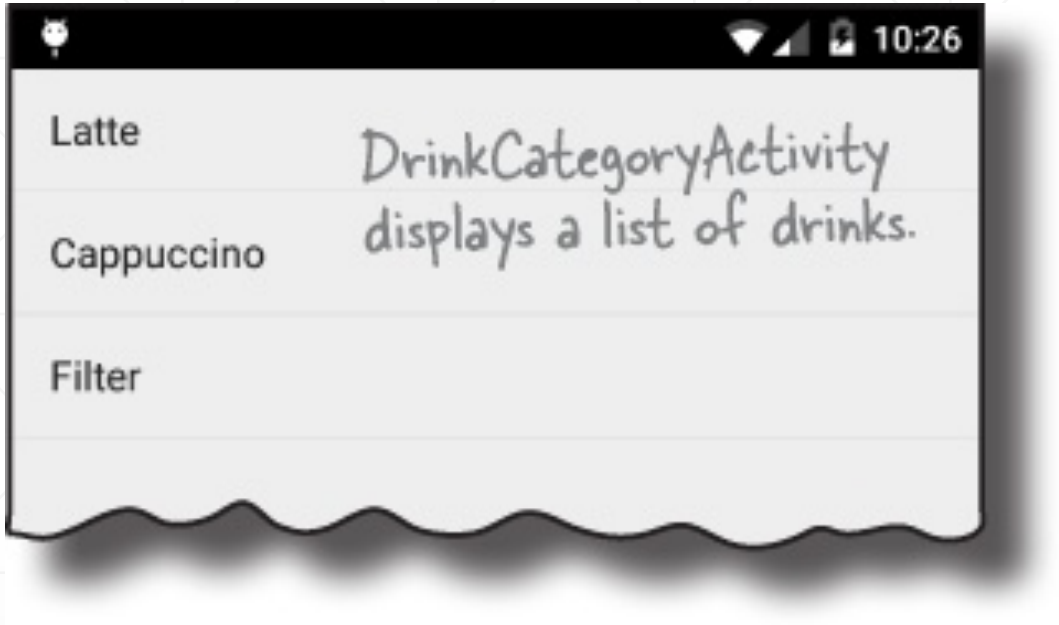
The name of the drink is the first item in the cursor, the description is the second column, and the image resource ID is the third. That's because we told the cursor to use the NAME, DESCRIPTION, and IMAGE_RESOURCE_ID columns from the database in that order.

Use the data from the cursor to populate the views.

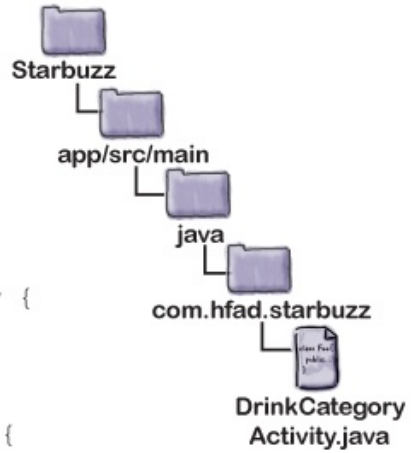
Close the cursor and database.

If a SQLiteException is thrown, this means there's a problem with the database. In this case, we'll use a Toast to display a message to the user.

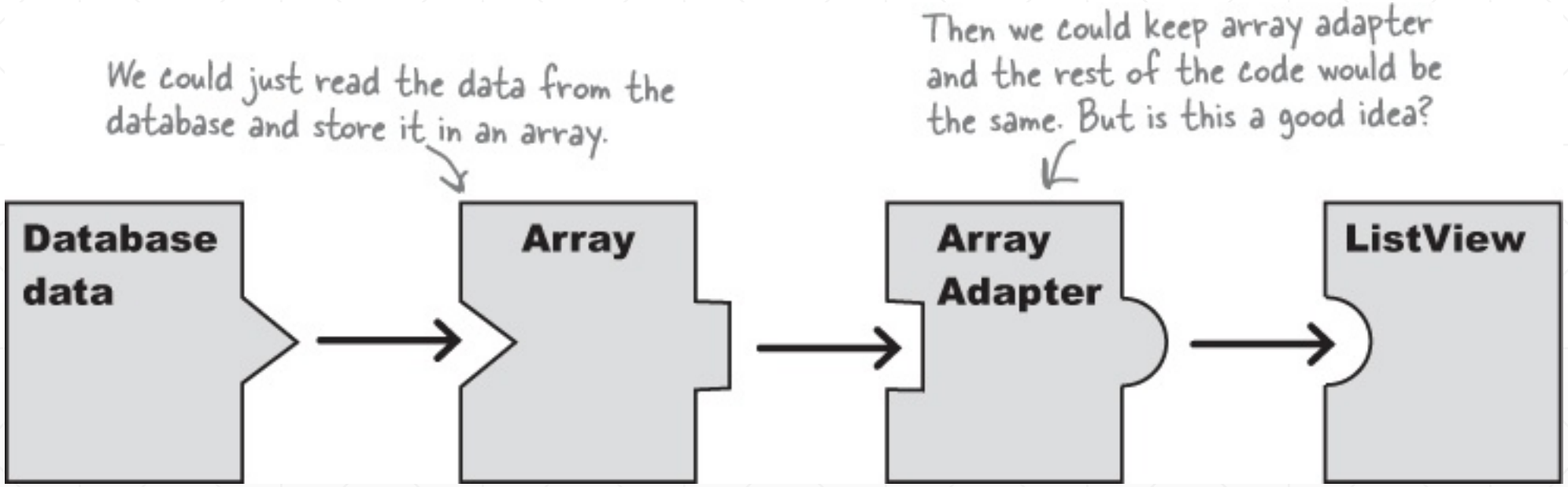

<Layout>
</Layout>
activity_top_level.xml
DrinkCategoryActivity still gets its drink data from the Drink class.
Drink.java
<Layout>
</Layout>
activity_drink.xml
Device
TopLevelActivity.java
DrinkCategoryActivity.java
DrinkActivity.java
DrinkActivity has been changed so it gets its data from the Starbuzz database via the SQLite helper.
Starbuzz database
SQLite Helper

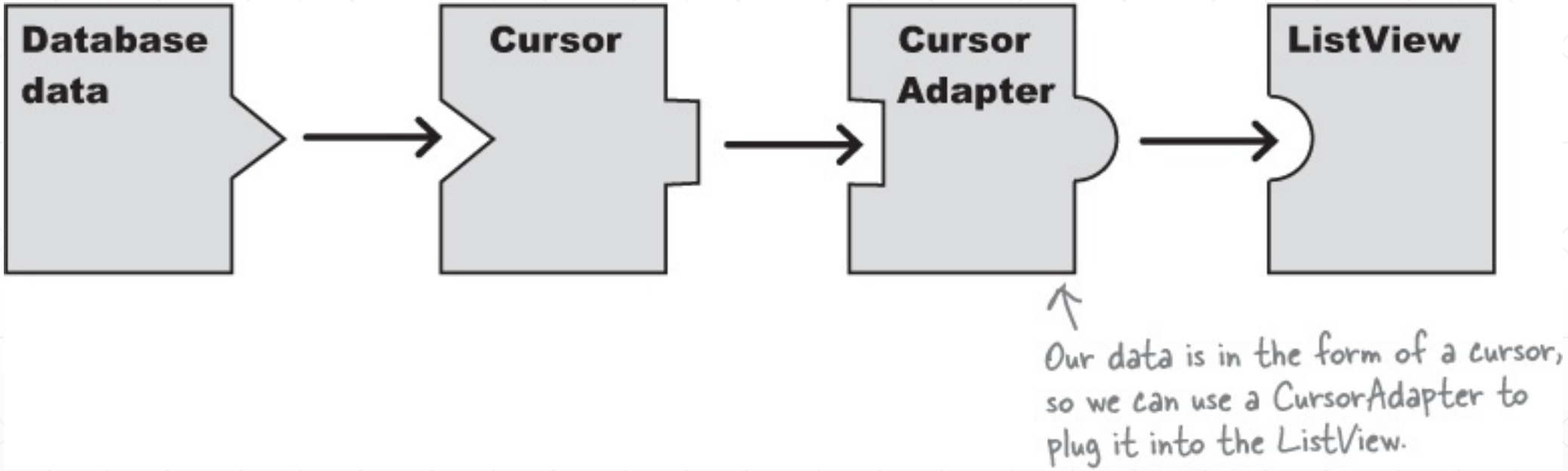

10:26
Latte
Cappuccino
Filter
DrinkCategoryActivity
displays a list of drinks.

```
package com.hfad.starbuzz;

import android.app.ListActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.view.View;
import android.content.Intent;

public class DrinkCategoryActivity extends ListActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ListView listDrinks = getListView();
        ArrayAdapter<Drink> listAdapter = new ArrayAdapter<Drink>(
                this,
                android.R.layout.simple_list_item_1,
                Drink.drinks);
        listDrinks.setAdapter(listAdapter);
    }

    @Override
    public void onListItemClick(ListView listView,
                                View itemView,
                                int position,
                                long id) {
        Intent intent = new Intent(DrinkCategoryActivity.this, DrinkActivity.class);
        intent.putExtra(DrinkActivity.EXTRA_DRINKNO, (int)id);
        startActivity(intent);
    }
}
```

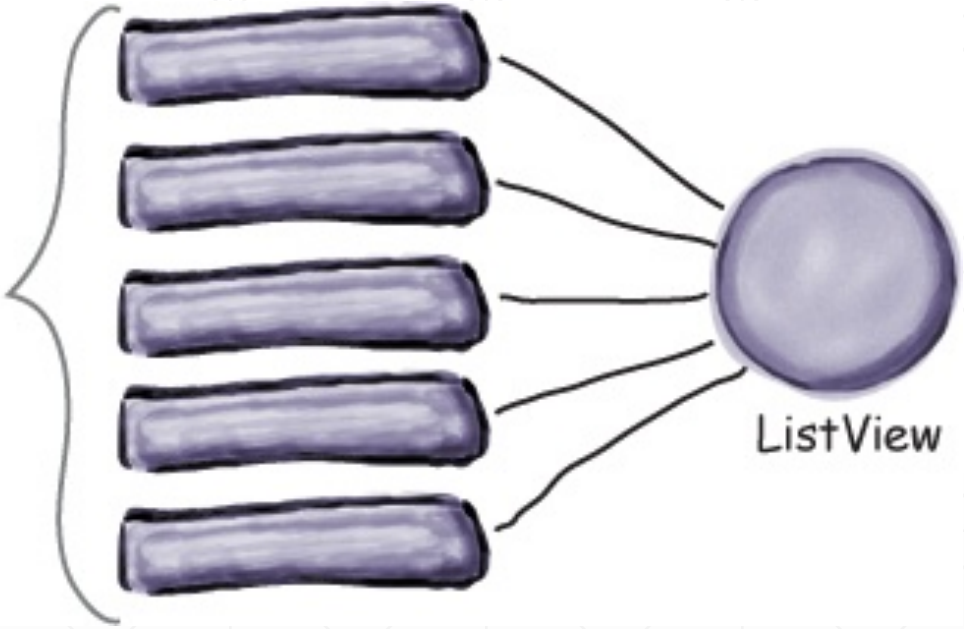


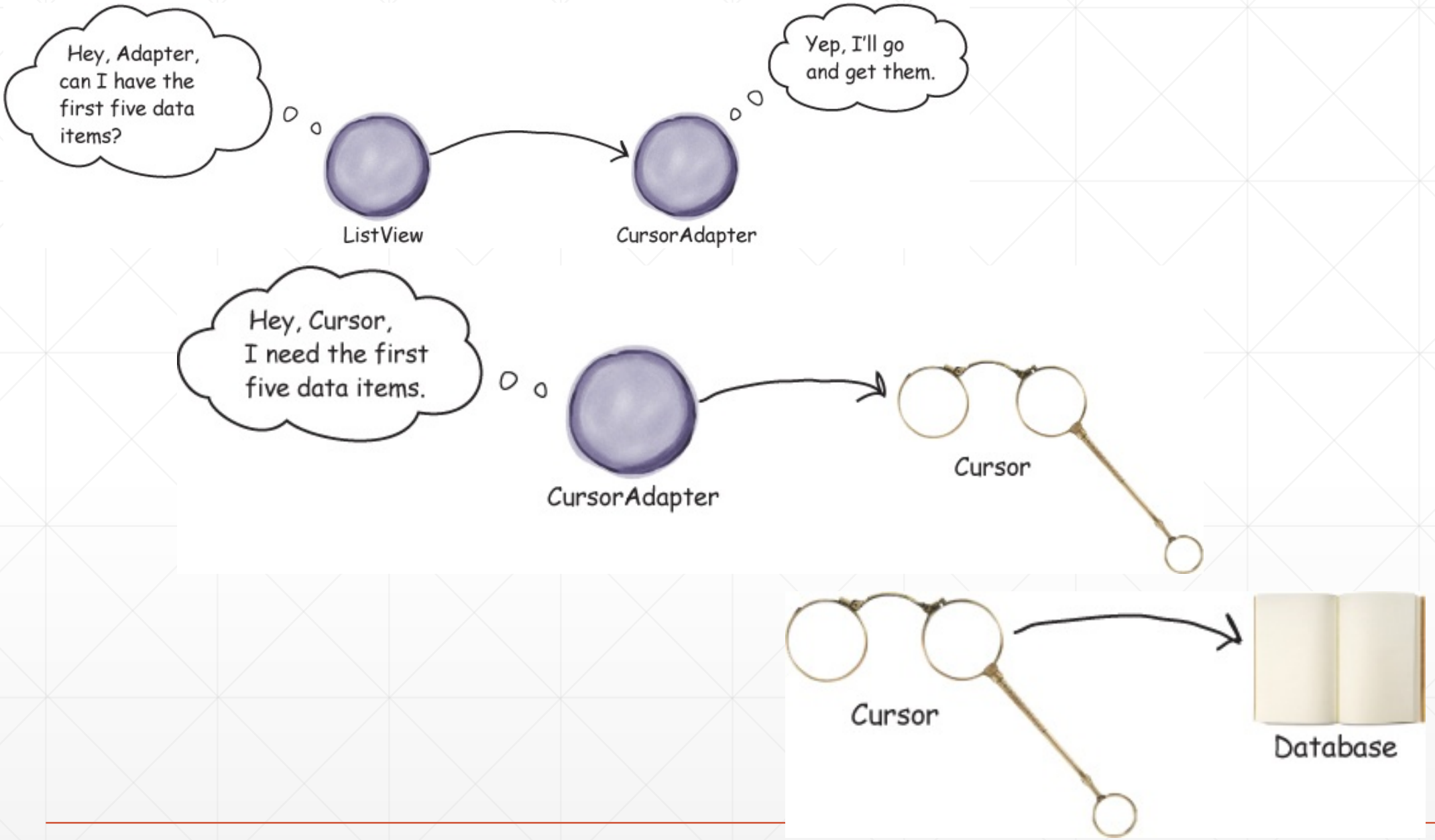
At the moment, we're using an ArrayAdapter to bind an array to the ListView. We need to replace this code so that the data comes from a database instead.

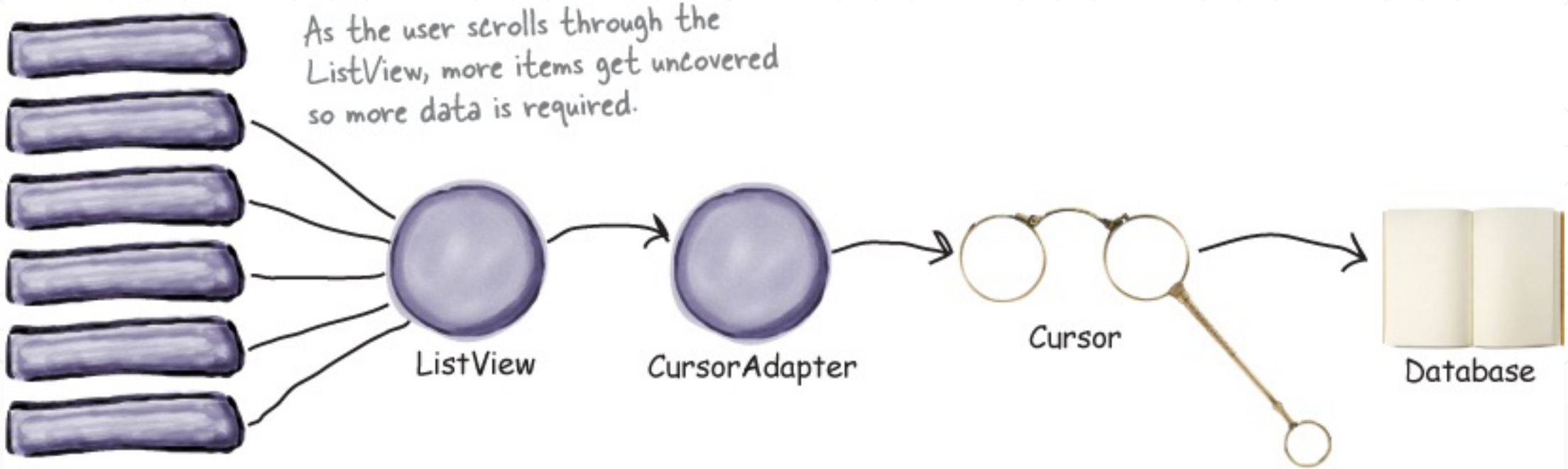

We could just read the data from the database and store it in an array.
Then we could keep array adapter and the rest of the code would be the same. But is this a good idea?
Database data
Array
Array Adapter
ListView

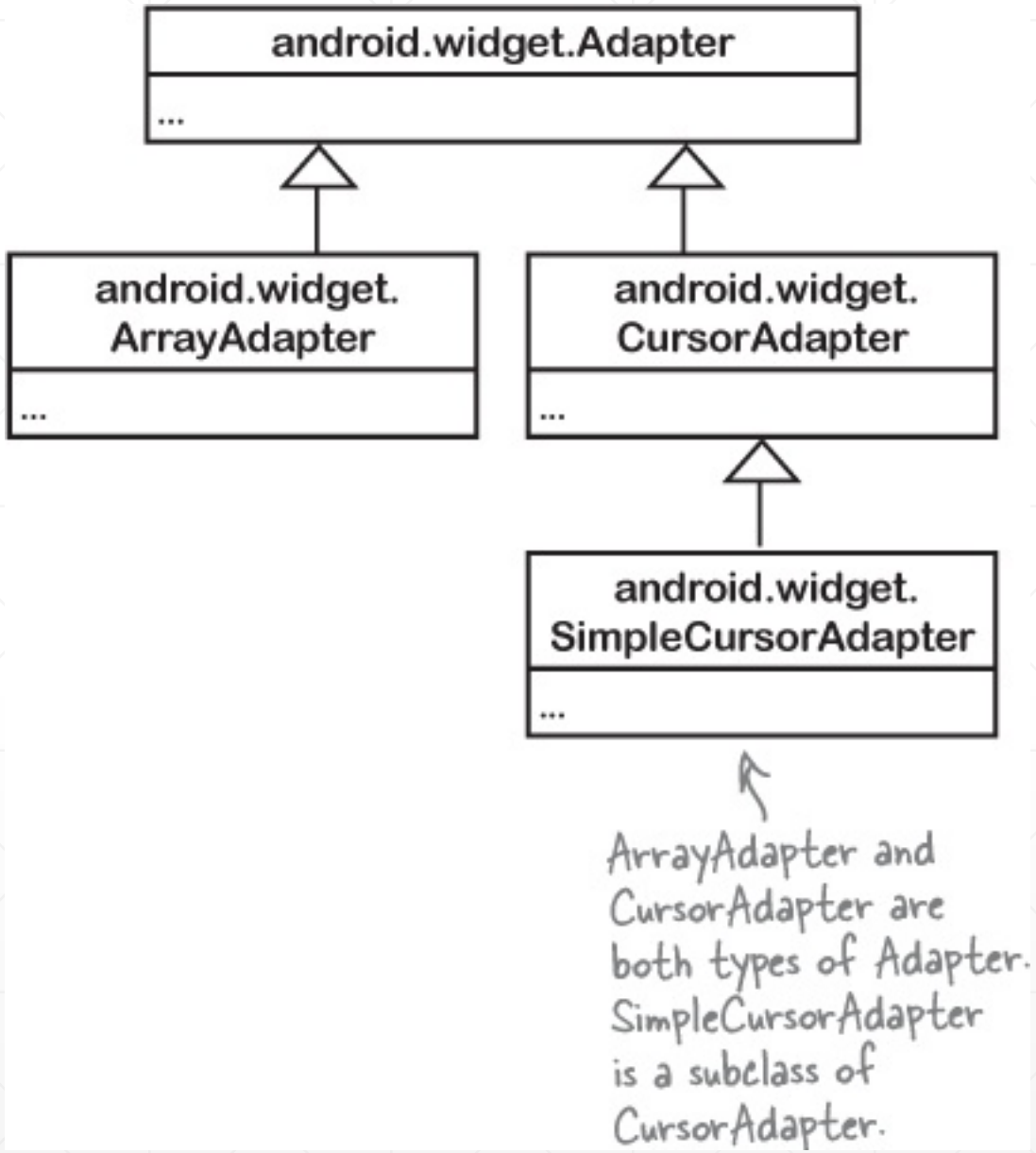

Database data
Cursor
Cursor Adapter
ListView
Our data is in the form of a cursor, so we can use a CursorAdapter to plug it into the ListView.

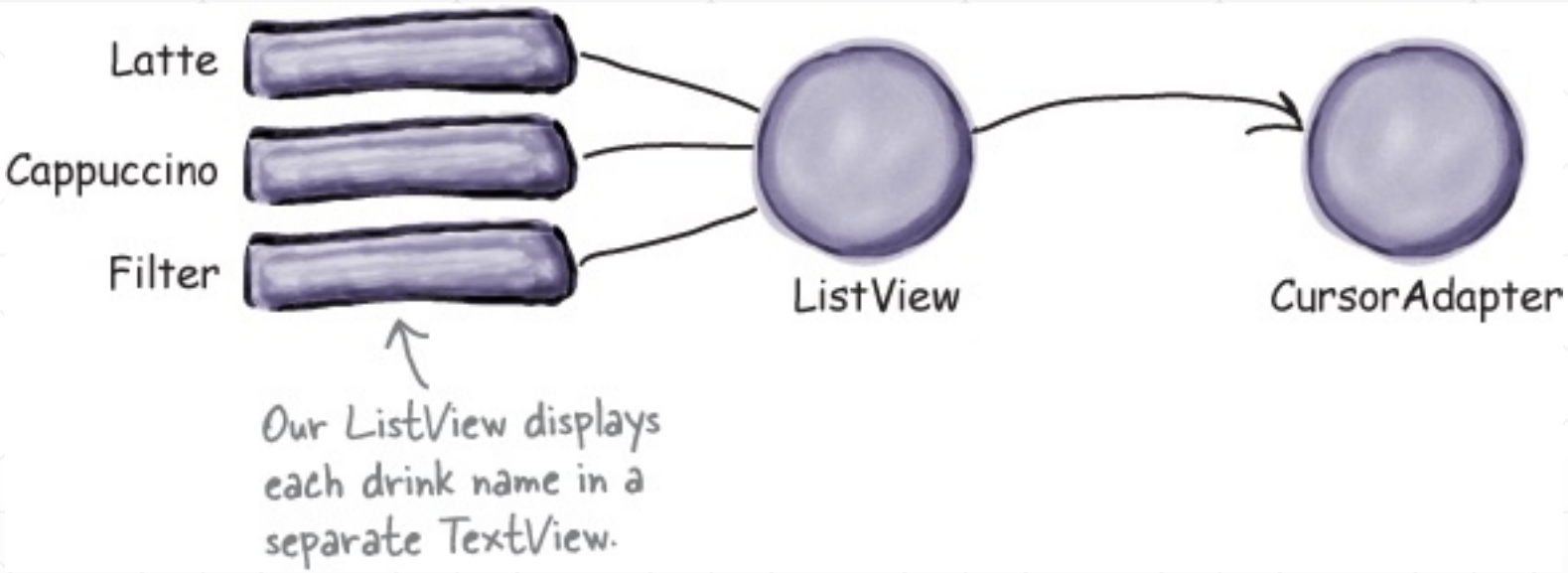
These are the items the ListView has space to display. We're using five to keep things simple, but in practice it's likely to be more.




Hey, Adapter, can I have the first five data items?
ListView
CursorAdapter
Yep, I'll go and get them.
Hey, Cursor, I need the first five data items.
CursorAdapter
Cursor
Cursor
Database


As the user scrolls through the ListView, more items get uncovered so more data is required.
ListView
CursorAdapter
Cursor
Database


android.widget.Adapter
...
android.widget.
ArrayAdapter
...
android.widget.
CursorAdapter
...
android.widget.
SimpleCursorAdapter
...
ArrayAdapter and
CursorAdapter are
both types of Adapter.
SimpleCursorAdapter
is a subclass of
CursorAdapter.


Latte
Cappuccino
Filter
ListView
CursorAdapter
Our ListView displays each drink name in a separate TextView.

```
cursor = db.query("DRINK", new String[]{"_id", "NAME"},
                  null, null, null, null, null);
```

We must include the _id column, even though we're not displaying its data.

This is the cursor from the previous page.

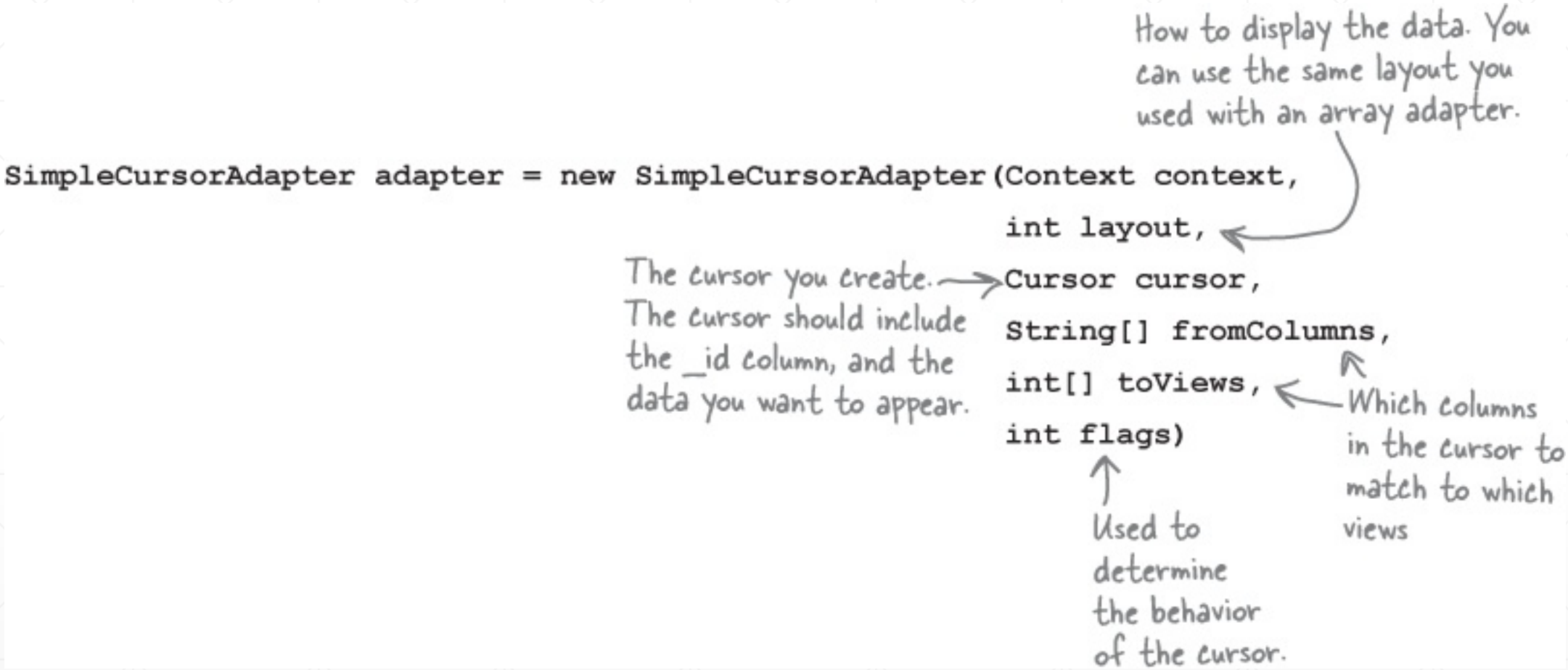
```
cursor = db.query("DRINK", new String[]{"_id", "NAME"},
                  null, null, null, null, null);
CursorAdapter listAdapter = new SimpleCursorAdapter(this,
                  android.R.layout.simple_list_item_1,
                  cursor,
                  new String[]{"NAME"},
                  new int[]{android.R.id.text1},
                  0);
listDrinks.setAdapter(listAdapter);
```

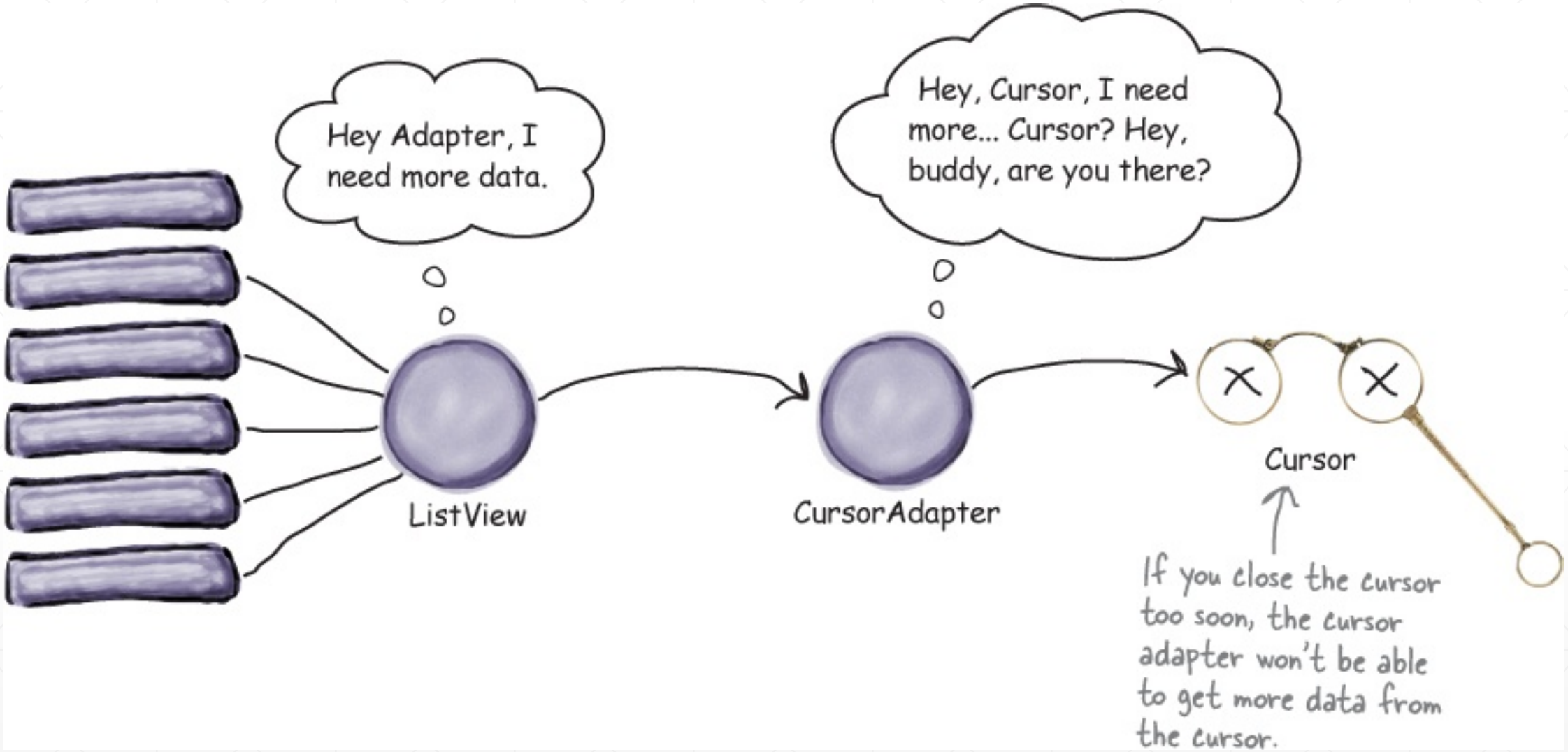
This is the same layout we used with the array adapter. It displays a single value for each row in the list view.

This is the cursor.

Display the contents of the NAME column in the ListView text views.

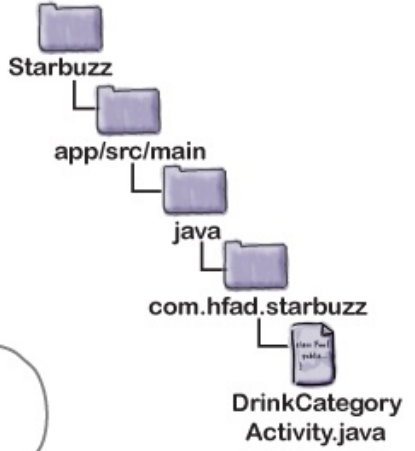
Use setAdapter() to connect the adapter to the list view.


How to display the data. You can use the same layout you used with an array adapter.
SimpleCursorAdapter adapter = new SimpleCursorAdapter(Context context,
int layout,
The cursor you create. The cursor should include the _id column, and the data you want to appear.
Cursor cursor,
String[] fromColumns,
int[] toViews,
Which columns in the cursor to match to which views
int flags)
Used to determine the behavior of the cursor.


Hey Adapter, I need more data.
Hey, Cursor, I need more... Cursor? Hey, buddy, are you there?
ListView
CursorAdapter
Cursor
If you close the cursor too soon, the cursor adapter won't be able to get more data from the cursor.

```
public void onDestroy(){
    super.onDestroy();
    cursor.close();
    db.close();
}
```

Close the cursor and database when the activity is destroyed.




```
package com.hfad.starbuzz;

import android.app.ListActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ListView;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import android.database.sqlite.SQLiteOpenHelper;
import android.widget.CursorAdapter;
import android.widget.SimpleCursorAdapter;
import android.widget.Toast;
import android.widget.SimpleCursorAdapter;

public class DrinkCategoryActivity extends ListActivity {
    private SQLiteDatabase db;
    private Cursor cursor;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ListView listDrinks = getListView();

        try {
            SQLiteOpenHelper starbuzzDatabaseHelper = new StarbuzzDatabaseHelper(this);
            db = starbuzzDatabaseHelper.getReadableDatabase();

            cursor = db.query("DRINK",
                              new String[]{"_id", "NAME"},
                              null, null, null, null, null);
```

We're using these extra classes, so you need to import them.

We're adding these as private variables so we can close the database and cursor in our onDestroy() method.

Get a reference to the database.

Create the cursor.

```
        CursorAdapter listAdapter = new SimpleCursorAdapter(this,
                                        android.R.layout.simple_list_item_1,
                                        cursor,
                                        new String[]{"NAME"},
                                        new int[]{android.R.id.text1},
                                        0);
        listDrinks.setAdapter(listAdapter);
    } catch(SQLiteException e) {
        Toast toast = Toast.makeText(this, "Database unavailable", Toast.LENGTH_SHORT);
        toast.show();
    }
}

@Override
public void onDestroy(){
    super.onDestroy();
    cursor.close();
    db.close();
}

@Override
public void onListItemClick(ListView listView,
                            View itemView,
                            int position,
                            long id) {
    Intent intent = new Intent(DrinkCategoryActivity.this, DrinkActivity.class);
    intent.putExtra(DrinkActivity.EXTRA_DRINKNO, (int)id);
    startActivity(intent);
}
}
```

Create the cursor adapter.

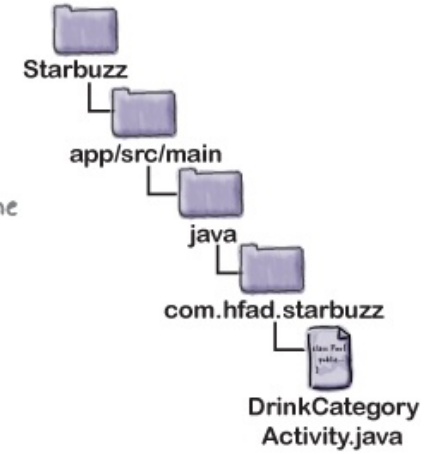
Map the contents of the NAME column to the text in the ListView.

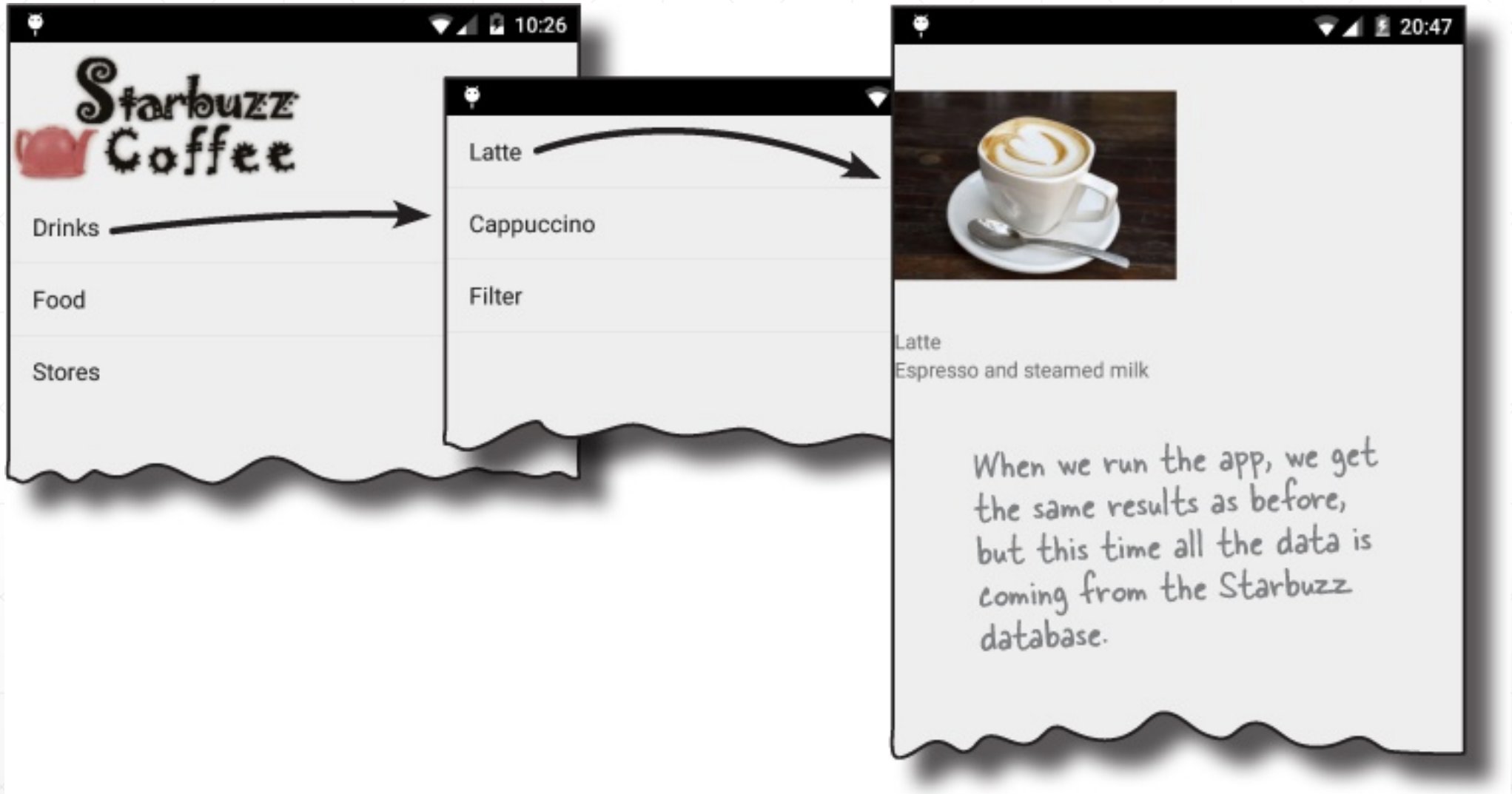
We're still using an adapter, but this time it's a cursor adapter.

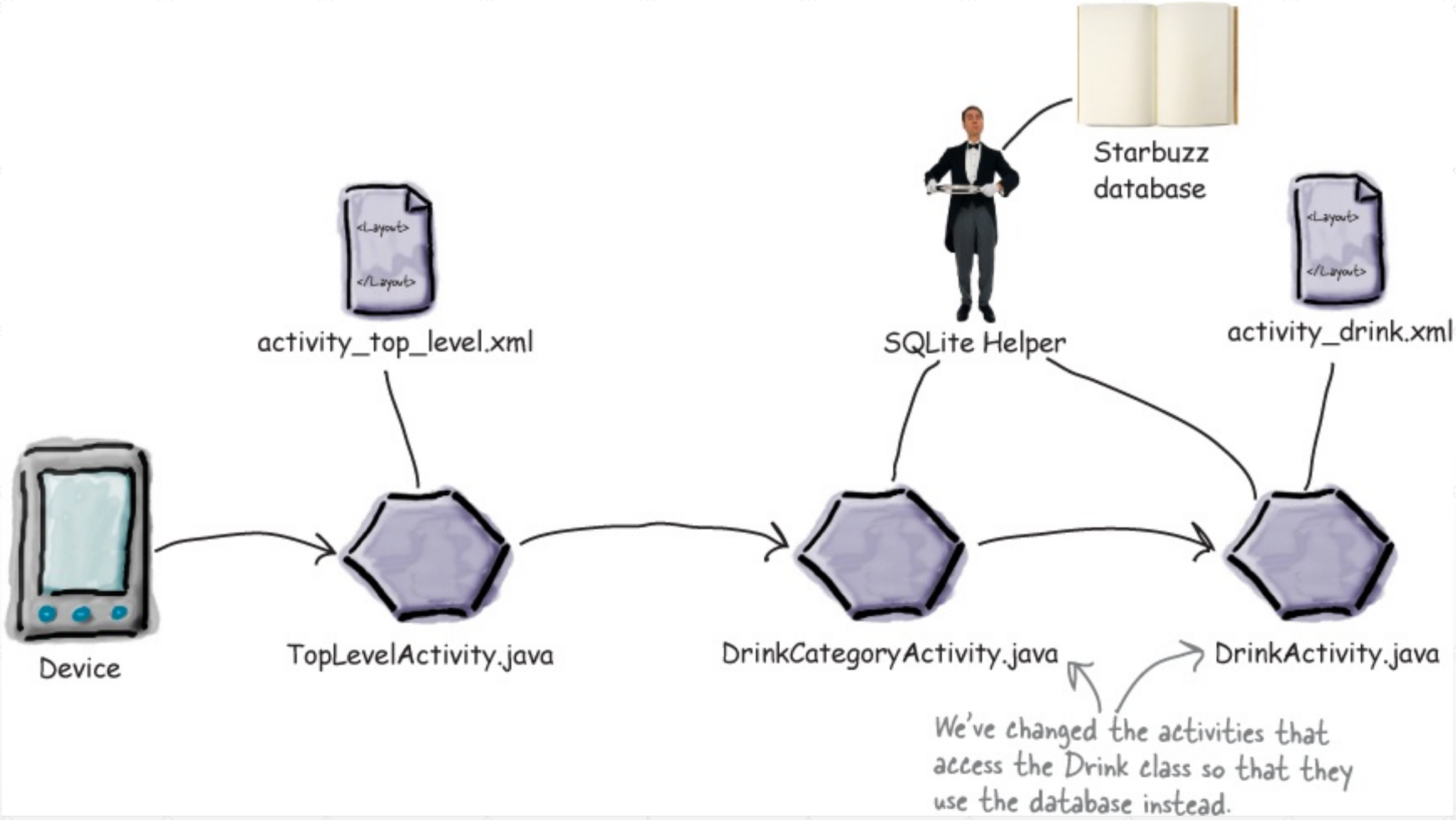
Display a message to the user if a SQLiteException gets thrown.

We're closing the database and cursor in the activity's onDestroy() method. The cursor will stay open until the cursor adapter no longer needs it.

We didn't need to change this method.




10:26
Starbuzz Coffee
Drinks
Food
Stores
Latte
Cappuccino
Filter
20:47
Latte
Espresso and steamed milk
When we run the app, we get the same results as before, but this time all the data is coming from the Starbuzz database.


<Layout>
</Layout>
activity_top_level.xml
Starbuzz
database
SQLite Helper
<Layout>
</Layout>
activity_drink.xml
Device
TopLevelActivity.java
DrinkCategoryActivity.java
DrinkActivity.java
We've changed the activities that
access the Drink class so that they
use the database instead.

These items ideally belong in a navigation drawer. We're not going to add a navigation drawer to the Starbuzz app, as navigation drawer code is a bit involved and we want you to focus on databases.

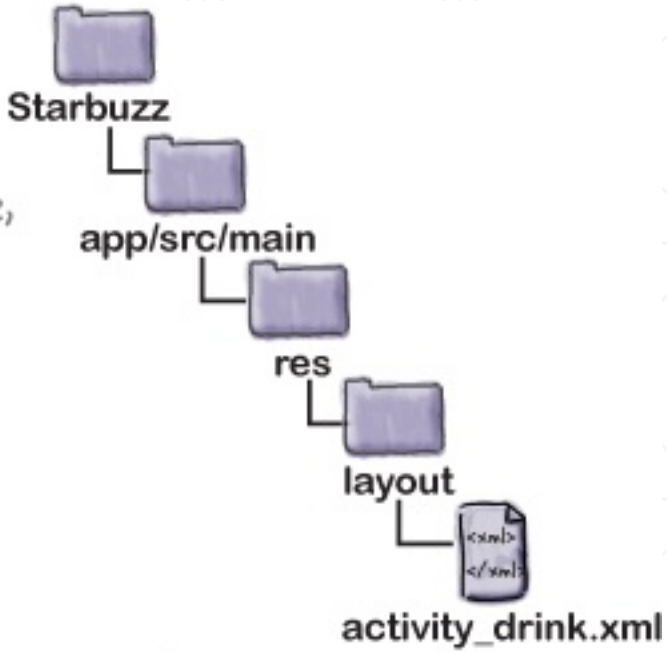
We'll add a ListView to TopLevelActivity, which contains the user's favorite drinks.

When the user clicks on a drink, it takes them through to that drink's details in DrinkActivity.

Users can say which drinks are their favorites by ticking a checkbox. We need to add this to DrinkActivity and get it to update the database.

```
<LinearLayout ...>
    <ImageView android:id="@+id/photo"
        ... />

    <TextView android:id="@+id/name"
        ... />

    <TextView android:id="@+id/description"
        ... />

    <CheckBox android:id="@+id/favorite"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/favorite"
        android:onClick="onFavoriteClicked"/>
</LinearLayout>
```

These are the photo, name, and description views we added when we first created the activity.



The checkbox has an ID of favorite.

We need to give the checkbox a label.

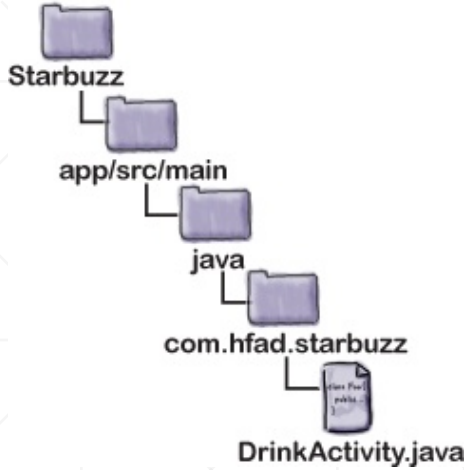
When the checkbox is clicked, the onFavoriteClicked() method will get called.

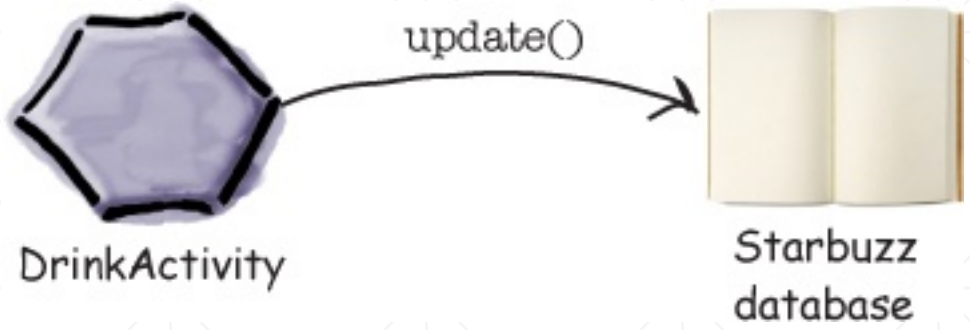
```
protected void onCreate(Bundle savedInstanceState) {
    ...
    SQLiteOpenHelper starbuzzDatabaseHelper = new StarbuzzDatabaseHelper(this);
    SQLiteDatabase db = starbuzzDatabaseHelper.getReadableDatabase();
    Cursor cursor = db.query ("DRINK",
                new String[] {"NAME", "DESCRIPTION", "IMAGE_RESOURCE_ID", "FAVORITE"},
                "_id = ?",
                new String[] {Integer.toString(drinkNo)},
                null, null,null);

    //Move to the first record in the Cursor
    if (cursor.moveToFirst()) {
        //Get the drink details from the cursor
        String nameText = cursor.getString(0);
        String descriptionText = cursor.getString(1);
        int photoId = cursor.getInt(2);
        boolean isFavorite = (cursor.getInt(3) == 1);
    ...
        //Populate the favorite checkbox
        CheckBox favorite = (CheckBox)findViewById(R.id.favorite);
        favorite.setChecked(isFavorite);
    ...
}
```

Add the FAVORITE column to the cursor.

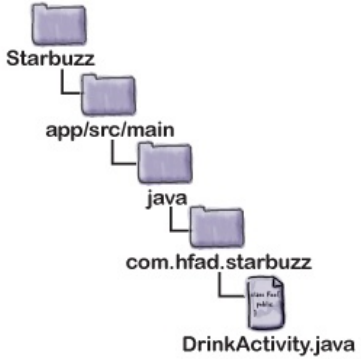
Get the value of the FAVORITE column. It's stored in the database as 1 for true, 0 for false.

Set the value of the favorite checkbox.






```
database.update(String table,
                ContentValues values,
                String whereClause,
                String[] whereArgs);
```




```
package com.hfad.starbuzz;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import android.database.sqlite.SQLiteOpenHelper;
import android.view.View;
import android.widget.CheckBox;
import android.content.ContentValues;

public class DrinkActivity extends Activity {

    public static final String EXTRA_DRINKNO = "drinkNo";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_drink);

        //Get the drink from the intent
        int drinkNo = (Integer)getIntent().getExtras().get(EXTRA_DRINKNO);

        //Create a cursor
        try {
            SQLiteOpenHelper starbuzzDatabaseHelper = new StarbuzzDatabaseHelper(this);
            SQLiteDatabase db = starbuzzDatabaseHelper.getWritableDatabase();
            Cursor cursor = db.query ("DRINK",
                    new String[] {"NAME", "DESCRIPTION", "IMAGE_RESOURCE_ID", "FAVORITE"},
                    "_id = ?",
                    new String[] {Integer.toString(drinkNo)},
                    null, null,null);

            //Move to the first record in the Cursor
            if (cursor.moveToFirst()) {

                //Get the drink details from the cursor
                String nameText = cursor.getString(0);
                String descriptionText = cursor.getString(1);
                int photoId = cursor.getInt(2);
                boolean isFavorite = (cursor.getInt(3) == 1);
```

We're using these extra classes.

You need read/write access to the database to update it.

Add the FAVORITE column to the cursor.

Get the value of the FAVORITE column.

```
            //Populate the drink name
            TextView name = (TextView)findViewById(R.id.name);
            name.setText(nameText);

            //Populate the drink description
            TextView description = (TextView)findViewById(R.id.description);
            description.setText(descriptionText);

            //Populate the drink image
            ImageView photo = (ImageView)findViewById(R.id.photo);
            photo.setImageResource(photoId);
            photo.setContentDescription(nameText);

            //Populate the favorite checkbox
            CheckBox favorite = (CheckBox)findViewById(R.id.favorite);
            favorite.setChecked(isFavorite);
        };
        cursor.close();
        db.close();
    } catch(SQLiteException e) {
        Toast toast = Toast.makeText(this, "Database unavailable", Toast.LENGTH_SHORT);
        toast.show();
    }
}

//Update the database when the checkbox is clicked
public void onFavoriteClicked(View view){
    int drinkNo = (Integer)getIntent().getExtras().get("drinkNo");
    CheckBox favorite = (CheckBox)findViewById(R.id.favorite);
    ContentValues drinkValues = new ContentValues();
    drinkValues.put("FAVORITE", favorite.isChecked());
    SQLiteOpenHelper starbuzzDatabaseHelper =
                            new StarbuzzDatabaseHelper(DrinkActivity.this);
    try {
        SQLiteDatabase db = starbuzzDatabaseHelper.getWritableDatabase();
        db.update("DRINK", drinkValues,
                            "_id = ?", new String[] {Integer.toString(drinkNo)});
        db.close();
    } catch(SQLiteException e) {
        Toast toast = Toast.makeText(this, "Database unavailable", Toast.LENGTH_SHORT);
        toast.show();
    }
}
}
```

Populate the checkbox.

Add the value of the favorite checkbox to the drinkValues ContentValues object.

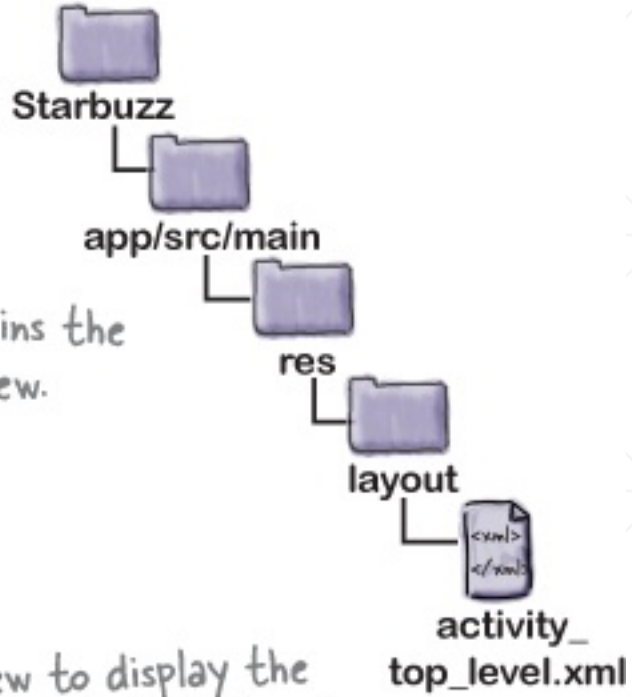
Update the FAVORITE column to the value of the checkbox.

Display a message if there's a problem with the database.


16:27
Starbuzz Coffee
Drinks
Food
Stores
Your favorite drinks:
Latte
16:28
Latte
Espresso and steamed milk
Favorite
The favorites ListView will get its data from the database using a cursor.
Cursor
Starbuzz database
The items in the favorites ListView need to respond to clicks.

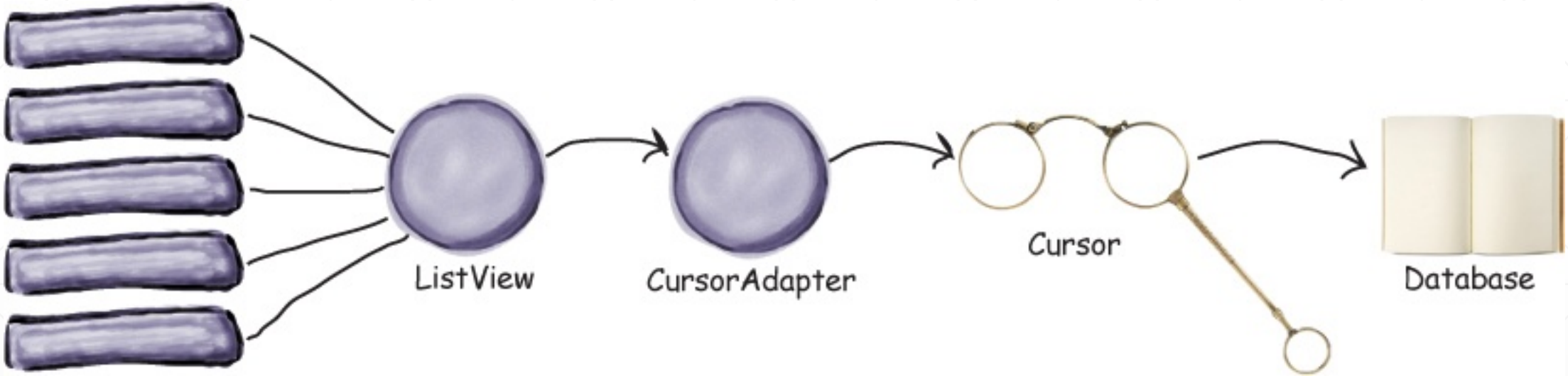
```
<LinearLayout ... >
    <ImageView
        android:layout_width="200dp"
        android:layout_height="100dp"
        android:src="@drawable/starbuzz_logo"
        android:contentDescription="@string/starbuzz_logo" />

    <ListView
        android:id="@+id/list_options"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:entries="@array/options" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="@string/favorites" />

    <ListView
        android:id="@+id/list_favorites"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

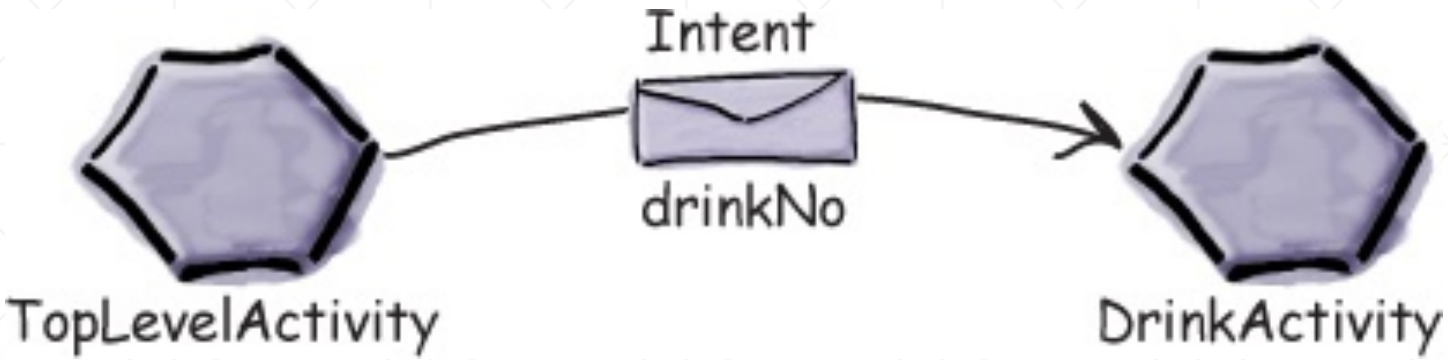
The layout already contains the Starbuzz logo and list view.

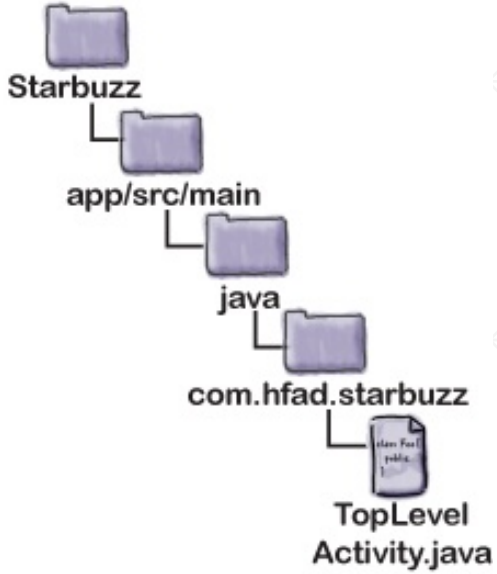


We'll add a text view to display the text "Your favorite drinks". We'll put this in a string called favorites.

The list_favorites ListView will display the user's favorite drinks.


ListView
CursorAdapter
Cursor
Database

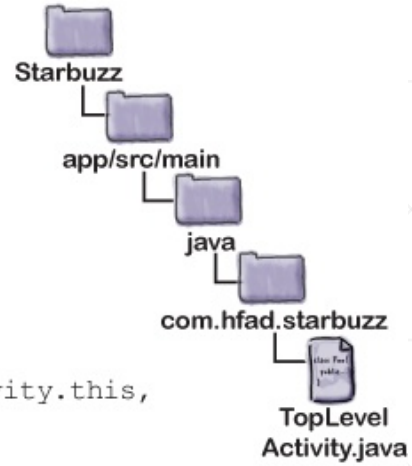

Intent
drinkNo
TopLevelActivity
DrinkActivity




```
package com.hfad.starbuzz;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.AdapterView;
import android.widget.ListView;
import android.view.View;
import android.database.Cursor;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteException;
import android.database.sqlite.SQLiteDatabase;
import android.widget.SimpleCursorAdapter;
import android.widget.CursorAdapter;
import android.widget.Toast;

public class TopLevelActivity extends Activity {

    private SQLiteDatabase db;
    private Cursor favoritesCursor;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_top_level);
```

We're using all these extra classes.

We're adding these as private variables so that we have access to them in the onDestroy() method.




```
//Create an OnItemClickListener for the Options ListView
AdapterView.OnItemClickListener itemClickListener =
        new AdapterView.OnItemClickListener(){
            public void onItemClick(AdapterView<?> listView,
                                    View v,
                                    int position,
                                    long id) {
                if (position == 0) {
                    Intent intent = new Intent(TopLevelActivity.this,
                            DrinkCategoryActivity.class);
                    startActivity(intent);
                }
            }
        };

//Add the listener to the Options ListView
ListView listView = (ListView) findViewById(R.id.list_options);
listView.setOnItemClickListener(itemClickListener);

//Populate the list_favorites ListView from a cursor
ListView listFavorites = (ListView)findViewById(R.id.list_favorites);
try{
    SQLiteOpenHelper starbuzzDatabaseHelper = new StarbuzzDatabaseHelper(this);
    db = starbuzzDatabaseHelper.getReadableDatabase();
    favoritesCursor = db.query("DRINK",
            new String[] { "_id", "NAME"},
            "FAVORITE = 1",
            null, null, null, null);
```

This is code we originally had in our onCreate() method. It populates the options list view and gets the list view to respond to clicks. We still need this code.

Get the favorites list view.

Create a cursor that gets the values of the _id and NAME columns where FAVORITE=1.

Get the names of the user's favorite drinks.

```
        CursorAdapter favoriteAdapter =
                new SimpleCursorAdapter(TopLevelActivity.this,
                        android.R.layout.simple_list_item_1,
                        favoritesCursor,
                        new String[]{"NAME"},
                        new int[]{android.R.id.text1}, 0);
        listFavorites.setAdapter(favoriteAdapter);
    } catch(SQLiteException e) {
        Toast toast = Toast.makeText(this, "Database unavailable", Toast.LENGTH_SHORT);
        toast.show();
    }

    //Navigate to DrinkActivity if a drink is clicked
    listFavorites.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> listView, View v, int position, long id)
{
            Intent intent = new Intent(TopLevelActivity.this, DrinkActivity.class);
            intent.putExtra(DrinkActivity.EXTRA_DRINKNO, (int)id);
            startActivity(intent);
        }
    });
}

//Close the cursor and database in the onDestroy() method
@Override
public void onDestroy(){
    super.onDestroy();
    favoritesCursor.close();
    db.close();
}
}
```

Use the cursor in the cursor adapter.

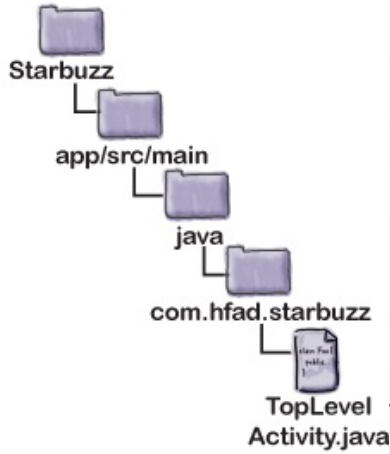
Display the names of the drinks in the ListView.

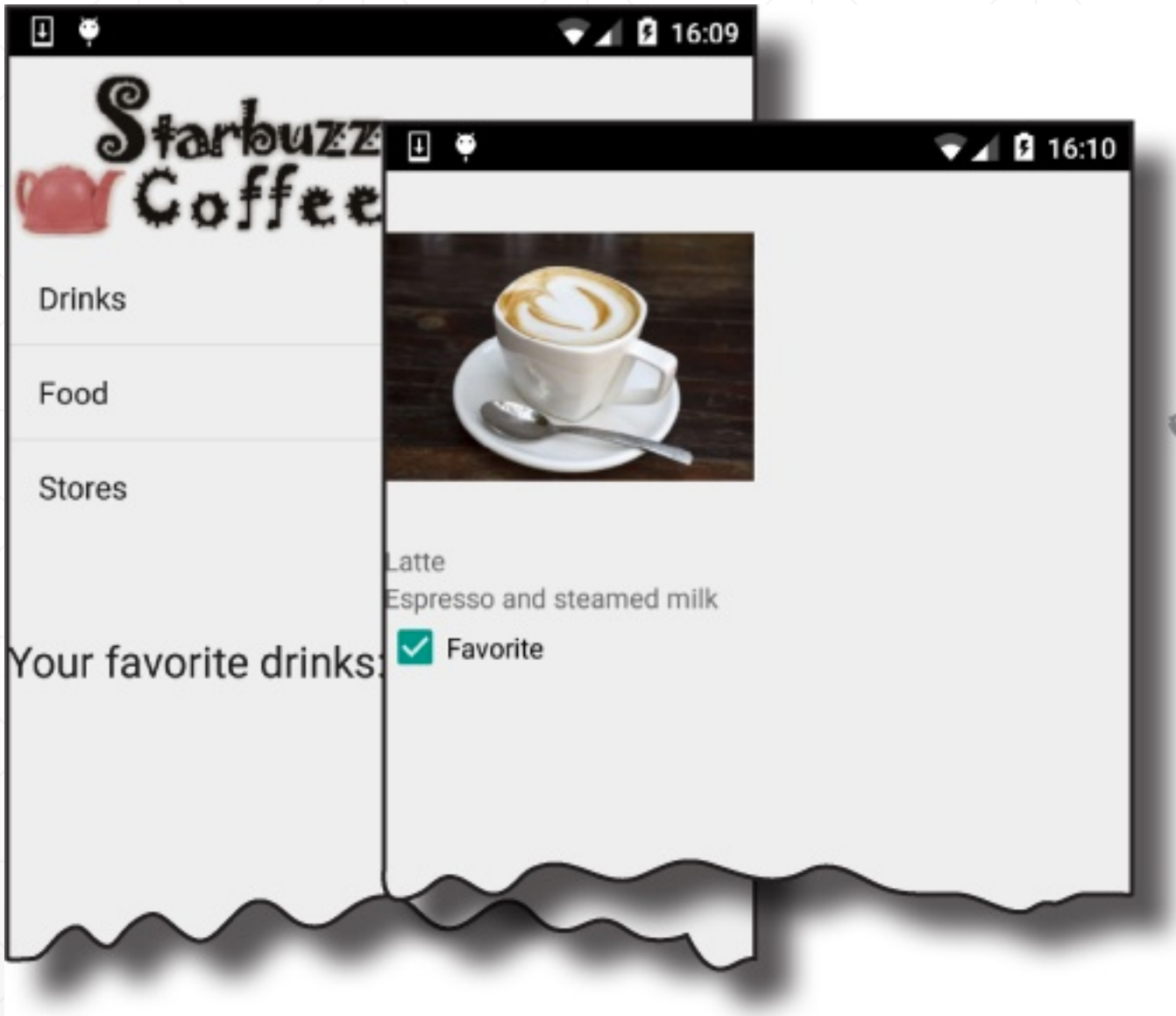
Display a message if there's a problem with the database

This will get called if an item in the list view is clicked.

If the user clicks on one of the items in the favorites ListView, create an intent to start DrinkActivity passing along the ID of the drink.

Close the cursor and database when the activity is destroyed.





When you start a second activity, the second activity is stacked on top of the first. The first activity isn't destroyed. Instead, it's paused then stopped, as it loses the focus and stops being visible to the user.

If you update the data in the database...

_id	NAME	DESCRIPTION	IMAGE_RESOURCE_ID	FAVORITE
1	"Latte"	"Espresso and steamed milk"	54543543	1
2	"Cappuccino"	"Espresso, hot milk and	654334453	0
3	"Filter"			

...the cursor won't see it if the cursor's already been created.

_id	NAME	DESCRIPTION	IMAGE_RESOURCE_ID	FAVORITE
1	"Latte"	"Espresso and steamed milk"	54543543	0
2	"Cappuccino"	"Espresso, hot milk and steamed-milk foam"	654334453	0
3	"Filter"	"Our best drip coffee"	44324234	0

```
public void changeCursor(Cursor newCursor)
```

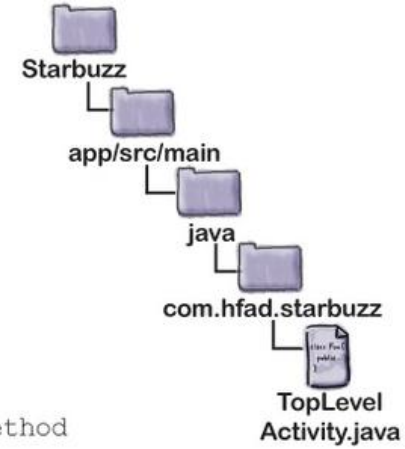
This is the new cursor you want the cursor adapter to use.

```
//Create the new cursor
StarbuzzDatabaseHelper starbuzzDatabaseHelper = new StarbuzzDatabaseHelper(this);
SQLiteDatabase db = starbuzzDatabaseHelper.getReadableDatabase();
Cursor cursor = db.query("DRINK",
                new String[] { "_id", "NAME"},
                "FAVORITE = 1",
                null, null, null, null);

//Get the CursorAdapter used by the ListView
ListView listFavorites = (ListView)findViewById(R.id.list_favorites);
CursorAdapter adapter = (CursorAdapter) listFavorites.getAdapter();

//Change the cursor used by the CursorAdapter to the new one we just created
adapter.changeCursor(cursor);
```

You create a new cursor in exactly the same way you did before.

You get the ListView's adapter using the getAdapter() method.

Change the cursor used by the cursor adapter to the new one.




```
package com.hfad.starbuzz;
...

public class TopLevelActivity extends Activity {
    ...
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
    }

    //Close the cursor and database in the onDestroy() method
    @Override
    public void onDestroy(){
        ...
    }

    public void onRestart() {
        super.onRestart();
        try{
            StarbuzzDatabaseHelper starbuzzDatabaseHelper = new StarbuzzDatabaseHelper(this);
            db = starbuzzDatabaseHelper.getReadableDatabase();
            Cursor newCursor = db.query("DRINK",
                    new String[] { "_id", "NAME"},
                    "FAVORITE = 1",
                    null, null, null, null);
            ListView listFavorites = (ListView)findViewById(R.id.list_favorites);
            CursorAdapter adapter = (CursorAdapter) listFavorites.getAdapter();
            adapter.changeCursor(newCursor);
            favoritesCursor = newCursor;
        } catch(SQLiteException e) {
            Toast toast = Toast.makeText(this, "Database unavailable", Toast.LENGTH_SHORT);
            toast.show();
        }
    }
}
```

These methods haven't changed.

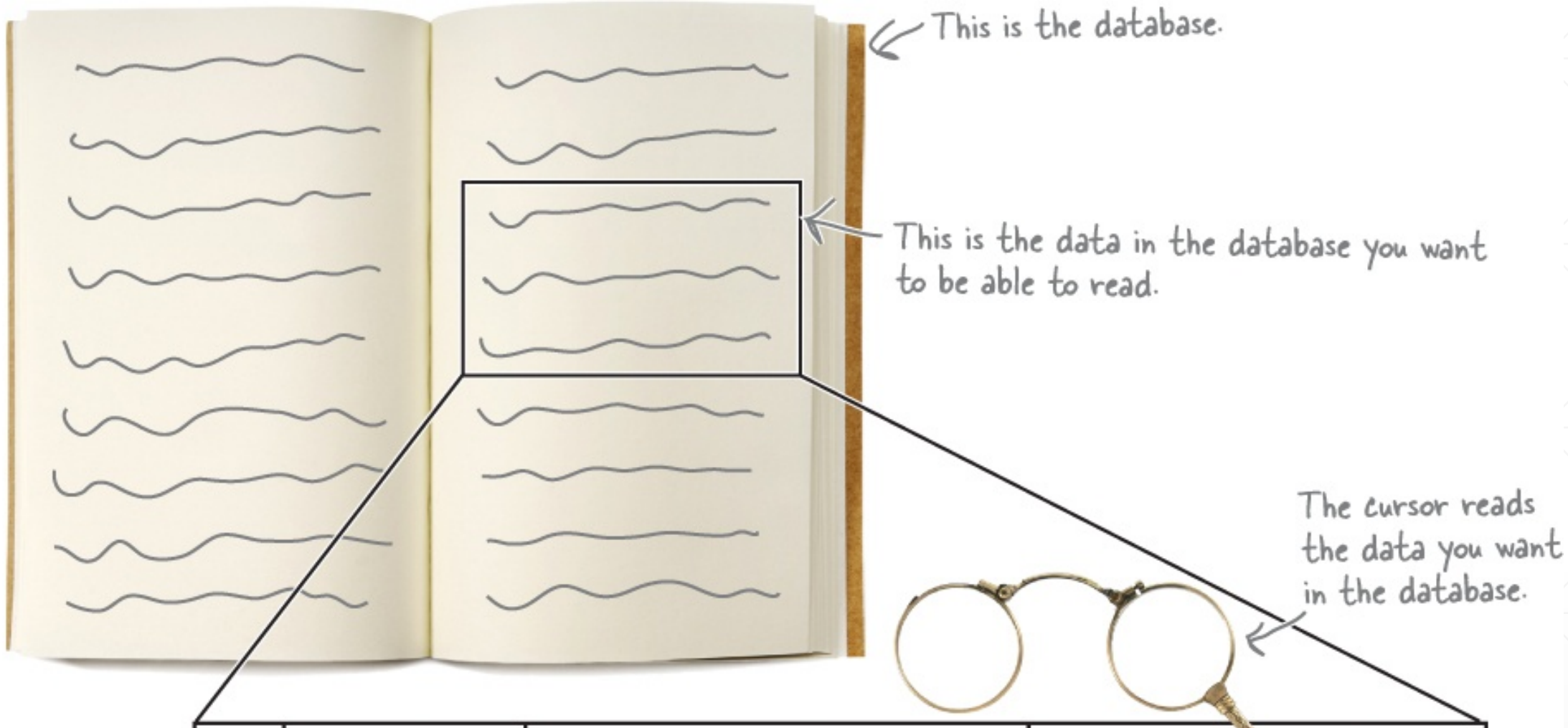
This gets called when the user returns to TopLevelActivity.

You create a new cursor in exactly the same way you did before.

Change the cursor used by the cursor adapter to the new one.

Get the list view's adapter.

Display a message if there's a problem with the database.



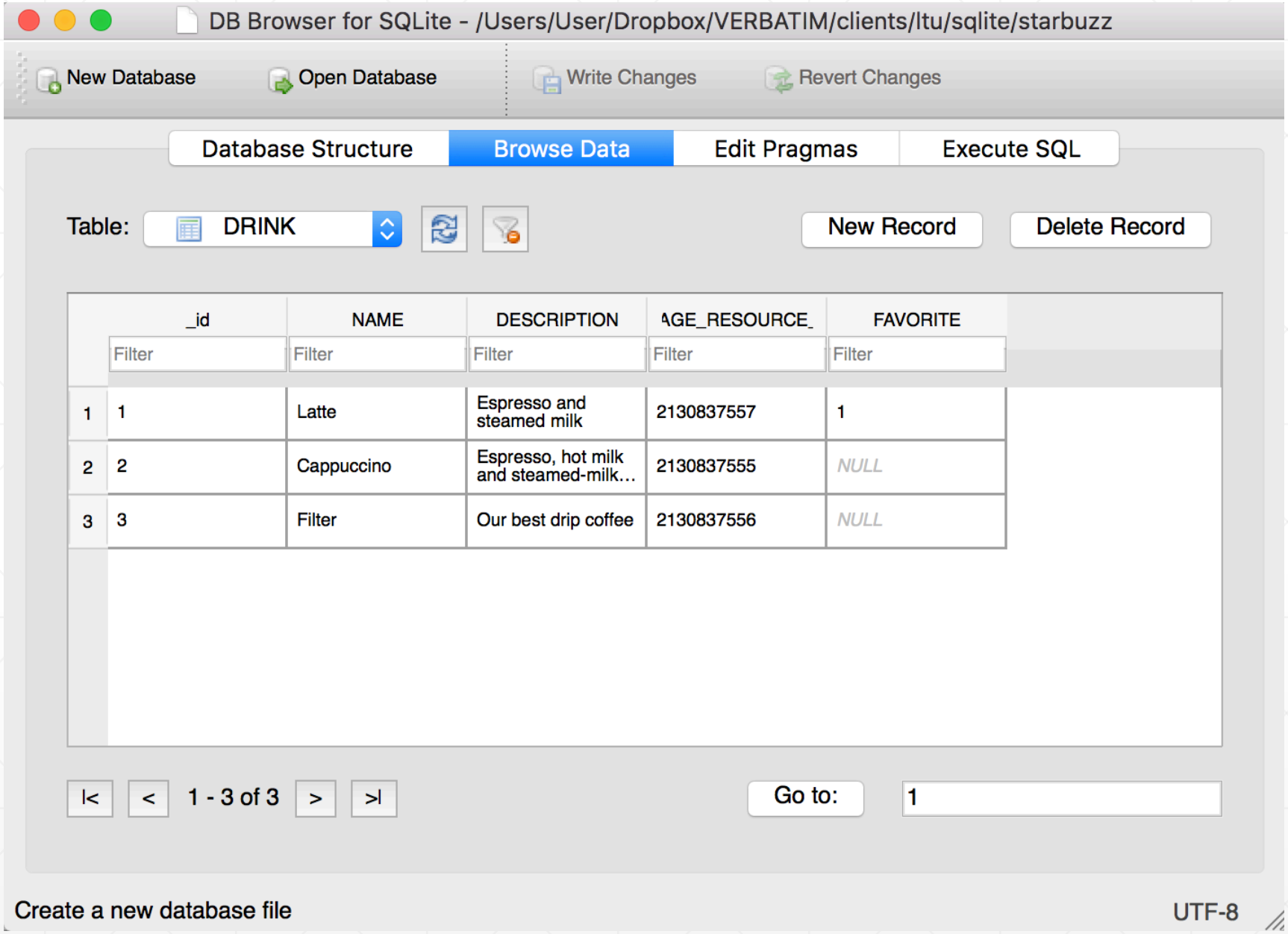
_id	NAME	DESCRIPTION	IMAGE_RESOURCE_ID
1	“Latte”	"Espresso and steamed milk"	54543543
2	“Cappuccino”	"Espresso, hot milk and steamed-milk foam"	654334453
3	“Filter”	"Our best drip coffee"	44324234

You navigate through the records supplied by the cursor and read their values.

Extra Credit

- SQLCipher
- SQLiteBrowser
- AsyncTasks

```
package com.demo.sqlcipher;

import java.io.File;
import net.sqlcipher.database.SQLiteDatabase;
import android.app.Activity;
import android.os.Bundle;

public class HelloSQLCipherActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        InitializeSQLCipher();
    }

    private void InitializeSQLCipher() {
        SQLiteDatabase.loadLibs(this);
        File databaseFile = getDatabasePath("demo.db");
        databaseFile.mkdirs();
        databaseFile.delete();
        SQLiteDatabase database = SQLiteDatabase.openOrCreateDatabase(databaseFile, "test123", null);
        database.execSQL("create table t1(a, b)");
        database.execSQL("insert into t1(a, b) values(?, ?)", new Object[]{"one for the money",
                                                                "two for the show"});
    }
}
```


DB Browser for SQLite - /Users/User/Dropbox/VERBATIM/clients/ltu/sqlite/starbuzz
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Table:
DRINK
New Record
Delete Record
_id
NAME
DESCRIPTION
AGE_RESOURCE_
FAVORITE
Filter
1 1 Latte Espresso and steamed milk 2130837557 1
2 2 Cappuccino Espresso, hot milk and steamed-milk… 2130837555 NULL
3 3 Filter Our best drip coffee 2130837556 NULL
1 - 3 of 3
Go to:
1
Create a new database file
UTF-8

Bullet Points

- A Cursor lets you read from and write to the database.
- You create a cursor by calling the SQLiteDatabase query() method. Behind the scenes, this builds a SQL SELECT statement.
- The getWritableDatabase() method returns a SQLiteDatabase object that allows you to read from and write to the database.
- The getReadableDatabase() returns a SQLiteDatabase object. This gives you read-only access to the database. It may also allow you to read from and write to the database, but this isn't guaranteed.
- Navigate through a cursor using the moveTo*() methods.
- Get values from a cursor using the get*() methods.

Bullet Points

- Close cursors and database connections after you've finished with them.
- A CursorAdapter is an adapter that works with cursors.
  Use SimpleCursorAdapter to populate a ListView with the values returned by a cursor.
- Design your app so that you put useful content in your top-level activity.
- The CursorAdapter changeCursor() method replaces the cursor currently used by a cursor adapter to a new cursor that you provide. It then closes the old cursor.
- Run your database code in a background thread using AsyncTask.