Build X: Android

Lecture Five:
Preferences and Databases

Notes: http://android.kcl.tech/

Recap

Recap: KCL Tech Todo

- We're going to build a basic todo list app, **KCL Tech Todo**.

- If you want a preview, it is **available on Google Play**.
  - Go to http://tiny.cc/kcl-tech-todo
  - Scan the QR code

KCL Tech Todo

Mark Ormesher Productivity

PEGI 3

This app is compatible with your device.

Recap: ListViews

- **ListViews** are...

Recap: ListViews

- **ListViews** are the best way to display data in any kind of list format.
    - Phone contacts, emails, tasks, documents, songs, anything!

- ListViews require an adapter to connect them to your data.
    - These have `getCount()`, `getItem()` and `getView()`.

Storing Data

Storing Data

- There are **many reasons** why you'd want to store data on a user's device.
  - Preferences, app content, high scores, game progress, usage logs, etc.

- This data is **persistent**, which means it still exists if your app closes or the device reboots.

- There are three main ways to achieve this...

Storing Data: Plain Files

- Your app can **write and read any kind of file** to a private area, accessible only to your app.

- This is useful if you have your own **proprietary file format**, or you have **large BLOBs** of data to store.

- We won’t be using or studying this method.

Storing Data: Shared Preferences

- **Shared preferences** is a utility built into Android that is primitive, but **very easy to use**.

- You can use shared preferences to store **simple key/value information**.

Storing Data: Databases

- Your app can **create and use** as many databases as it wants.

- Databases are great for storing **structured data** that needs to be organised, searchable, etc.

- Databases are by far the most **complicated**, but also the most **powerful**.

Shared Preferences

Storing Data: Shared Preferences

- **Shared preferences** are designed to store a user’s preferences and settings, like whether they have notifications turned on.

- However… you can use them for **any kind of simple key/value information**.

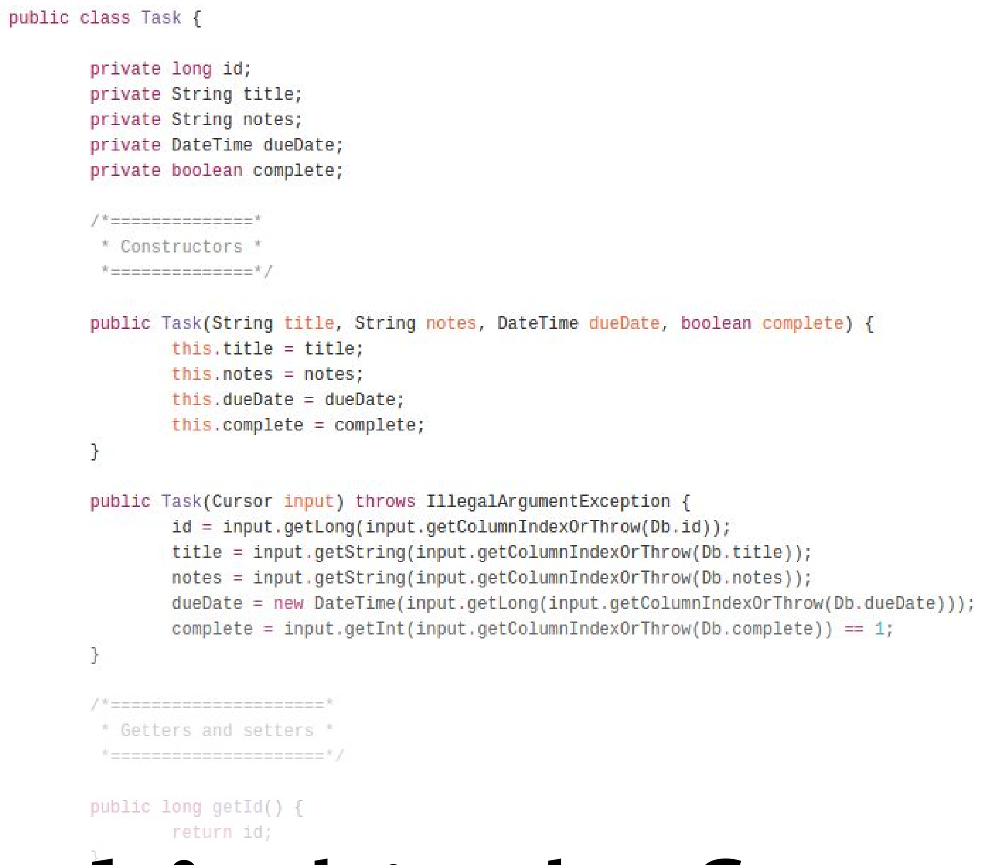
Storing Data: Shared Preferences

Key	Value
`user_highscore`	`42`
`welcome_tour_finished`	`true`
`notification_alert_sound`	`"bells"`

Coding Time

Side Note: Models

Not this kind of model.



```java
public class Task {

        private long id;
        private String title;
        private String notes;
        private DateTime dueDate;
        private boolean complete;

        /*==============*
         * Constructors *
         *==============*/

        public Task(String title, String notes, DateTime dueDate, boolean complete) {
                this.title = title;
                this.notes = notes;
                this.dueDate = dueDate;
                this.complete = complete;
        }

        public Task(Cursor input) throws IllegalArgumentException {
                id = input.getLong(input.getColumnIndexOrThrow(Db.id));
                title = input.getString(input.getColumnIndexOrThrow(Db.title));
                notes = input.getString(input.getColumnIndexOrThrow(Db.notes));
                dueDate = new DateTime(input.getLong(input.getColumnIndexOrThrow(Db.dueDate)));
                complete = input.getInt(input.getColumnIndexOrThrow(Db.complete)) == 1;
        }

        /*=====================*
         * Getters and setters *
         *=====================*/

        public long getId() {
                return id;
        }
```

This kind of model!

Models

- A model is a **representation** of some item or entity.

- In **Java (and therefore Android)**, a model generally means **a class** that represents one of the items or entities your software deals with.

Models

- We need a model that **represents a task**.

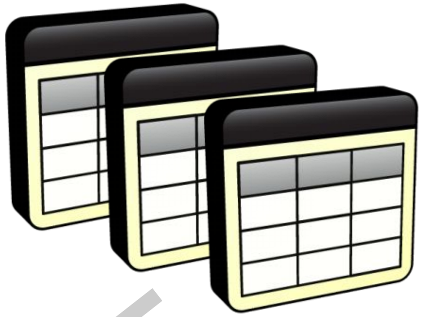
Coding Time

Break! (10 mins)

Databases

Databases: 60-Second Recap

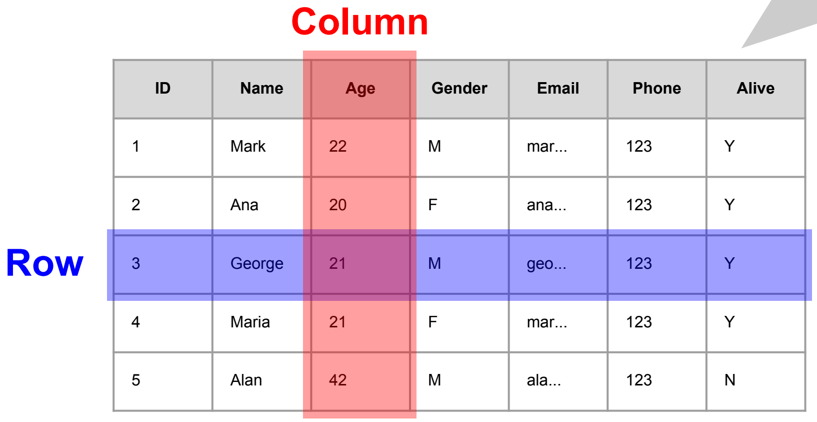
A database contains many tables



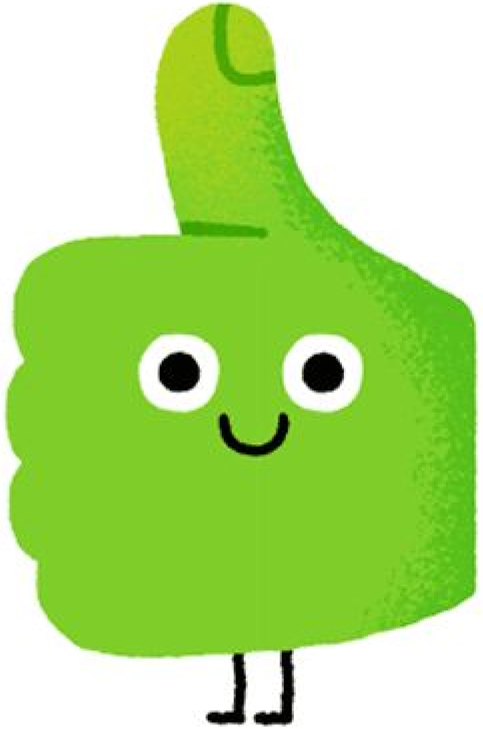
A table looks like this

Row:
A single object, item, record or entity of your data.

Column:
A property or attribute about your data.



ID	Name	Age	Gender	Email	Phone	Alive
1	Mark	22	M	mar...	123	Y
2	Ana	20	F	ana...	123	Y
3	George	21	M	geo...	123	Y
4	Maria	21	F	mar...	123	Y
5	Alan	42	M	ala...	123	N



Databases: SQL and SQLite

- Databases often use **SQL**, which stands for **Structured Query Language**.
    - A set of questions and commands you can send to a database.

- Android uses **SQLite**, which is a lightweight version of SQL.
    - It is a bit less powerful, but also better at running on mobile devices.

- Example queries:

```
SELECT Name, Age FROM Users;
```

```
SELECT * FROM Users WHERE alive = "Y";
```

```
SELECT * FROM Users ORDER BY age DESC;
```

Databases

- We’re going to cover four parts of using databases on Android:
    - Creating a database handler/helper
    - Creating a table
    - Inserting a task
    - Reading tasks

Databases:
Creating a Helper

Databases: Creating a Helper

- A **helper/handler** is a tool that makes working with SQLite databases easier.

- Remember how we made an adapter by extending `BaseAdapter`?

- To make our own helper, we extend `SQLiteOpenHelper`.

Coding Time

Databases:
Creating a Table

Databases: Creating a Table

- We will use an SQL command to create the table.

```
CREATE TABLE Tasks (
    id INTEGER PRIMARY KEY,
    title TEXT,
    notes TEXT,
    due_date INTEGER,
    is_complete INTEGER
);
```

pastebin.com/h61DrLjk

Coding Time

Databases:
Inserting a Task

Databases: Inserting a Task

- We **could** use SQL, but there’s a **better way**.

- If we can convert our task into a `ContentValues` object, we can insert that directly.

- We’re going to handle **create and update** in the same method by telling the database to **replace** any task that has the same ID.

Coding Time

Databases: Reading Tasks

Databases: Reading Tasks

- This time we will use SQL - we'll use two different queries.

- To get all uncomplete tasks:

```
SELECT * FROM Tasks WHERE is_complete = 0 ORDER BY due_date ASC;
```

- To get a single, specific task:

```
SELECT * FROM Tasks WHERE id = ?;
```

Databases: Reading Tasks / Cursors

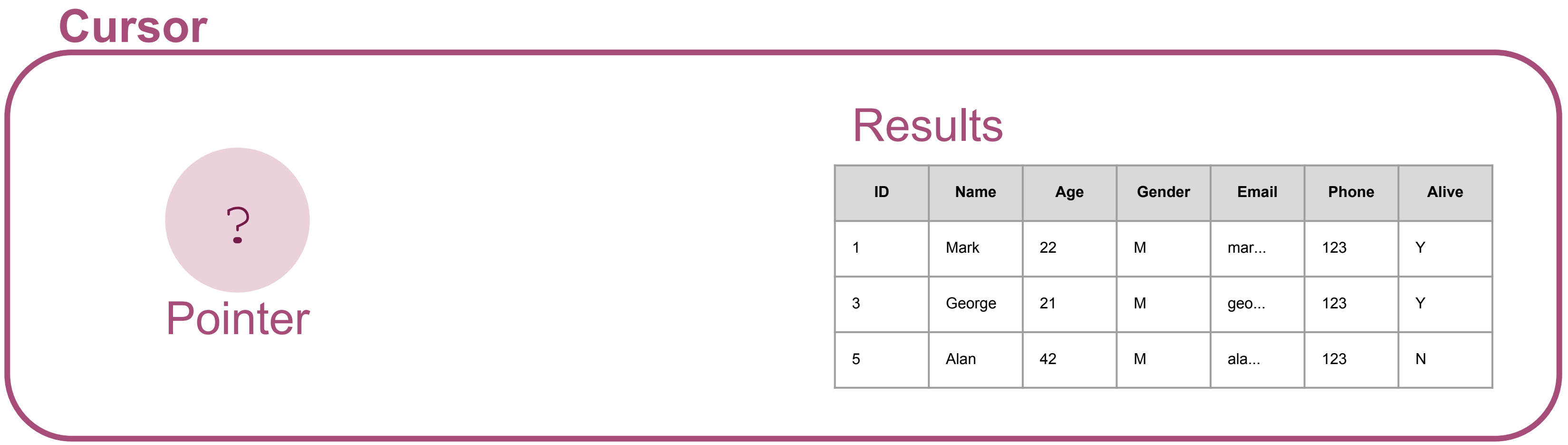
- Both of those queries return a `Cursor`. Cursors are special…

- A cursor is **one object** that can represent **any number of rows** from your database.

- It does this by only “looking at” **one record at a time** and using a **pointer** to remember which one it is currently looking at.

Databases: Reading Tasks / Cursors

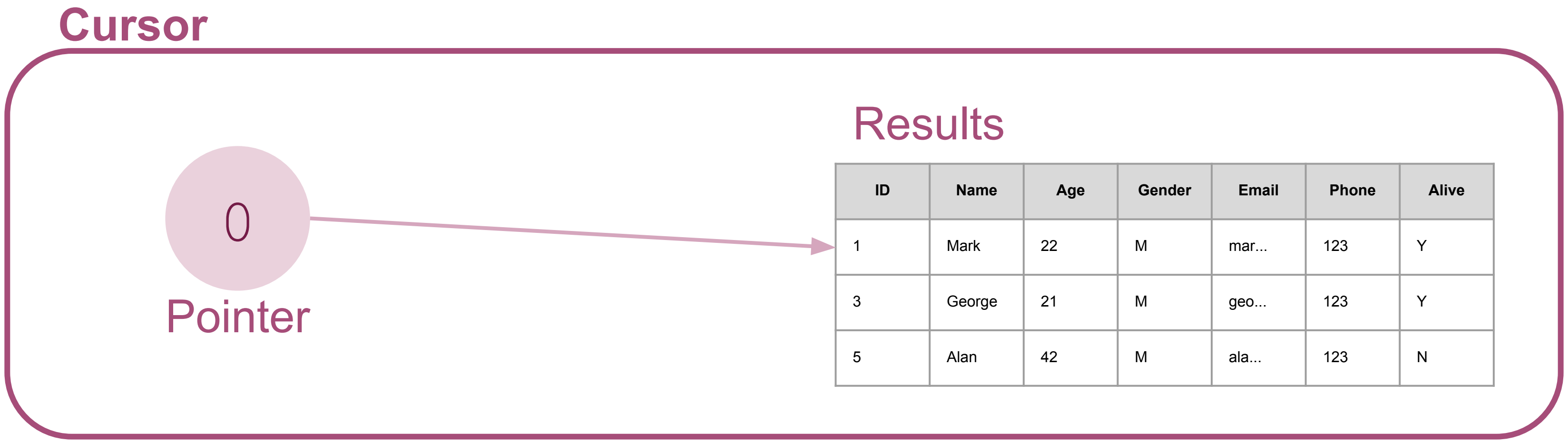
- Here is an **example result set**, and our **pointer**. To start with, it doesn't point at anything.



ID	Name	Age	Gender	Email	Phone	Alive
1	Mark	22	M	mar...	123	Y
3	George	21	M	geo...	123	Y
5	Alan	42	M	ala...	123	N

Databases: Reading Tasks / Cursors

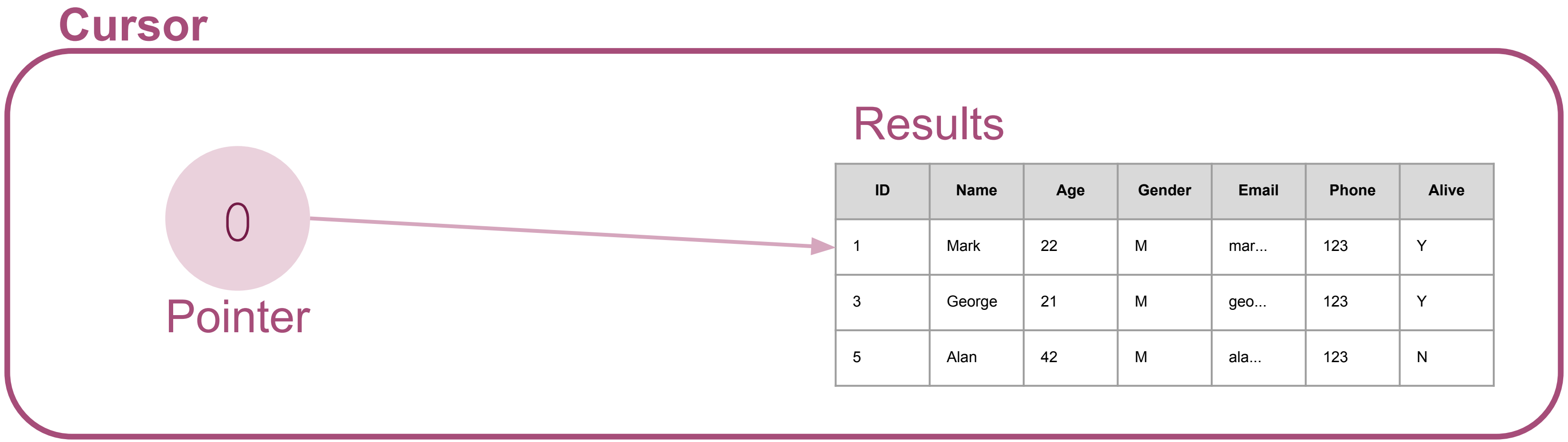
- We can call `moveToFirst()` to start looking at the first result.
- This returns `false` if there **isn't** a first result (i.e. the result set is empty).



ID	Name	Age	Gender	Email	Phone	Alive
1	Mark	22	M	mar...	123	Y
3	George	21	M	geo...	123	Y
5	Alan	42	M	ala...	123	N

Databases: Reading Tasks / Cursors

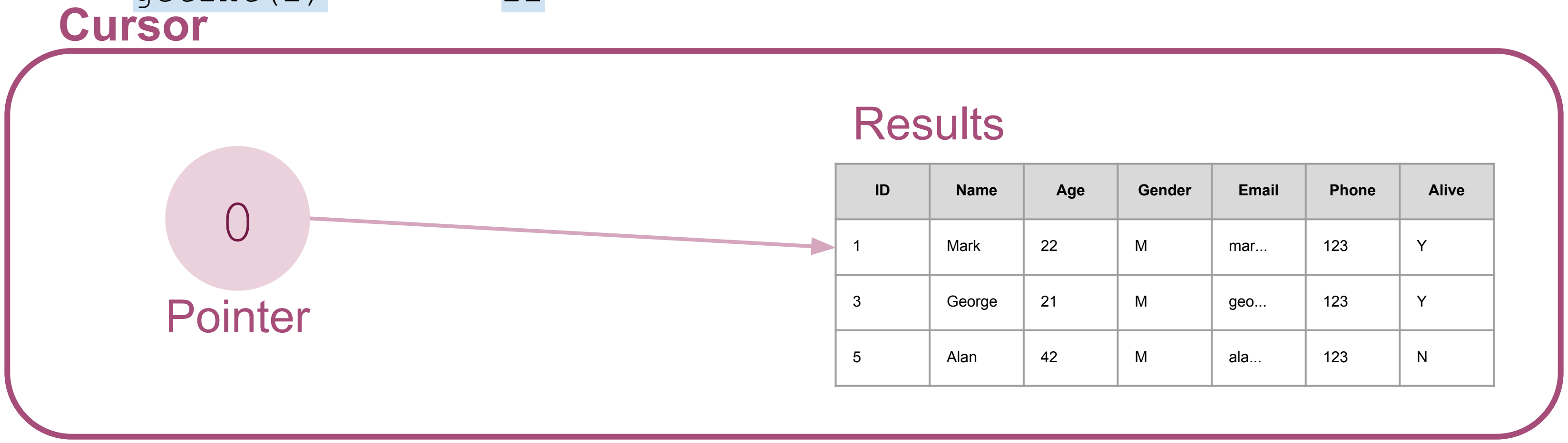
- We can now use `getInt(...)`, `getString(...)` to read information about the active result.



ID	Name	Age	Gender	Email	Phone	Alive
1	Mark	22	M	mar...	123	Y
3	George	21	M	geo...	123	Y
5	Alan	42	M	ala...	123	N

Databases: Reading Tasks / Cursors

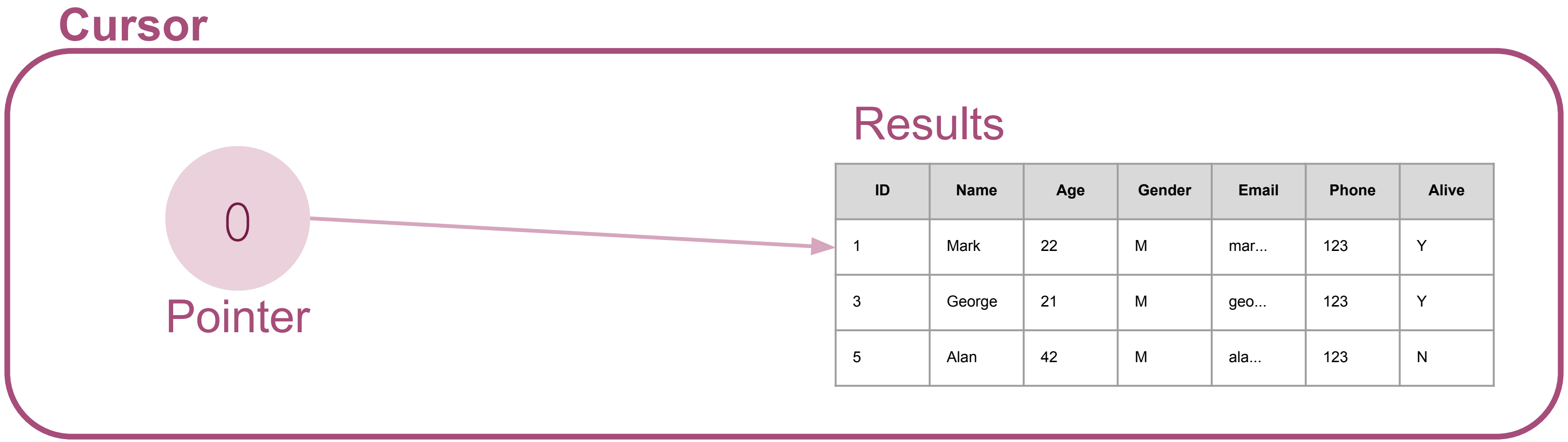
- These methods take the column number as an argument (starting at zero).
  - `getInt(0)` → `1`
    `getString(1)` → `"Mark"`
    `getInt(2)` → `22`



ID	Name	Age	Gender	Email	Phone	Alive
1	Mark	22	M	mar...	123	Y
3	George	21	M	geo...	123	Y
5	Alan	42	M	ala...	123	N

Databases: Reading Tasks / Cursors

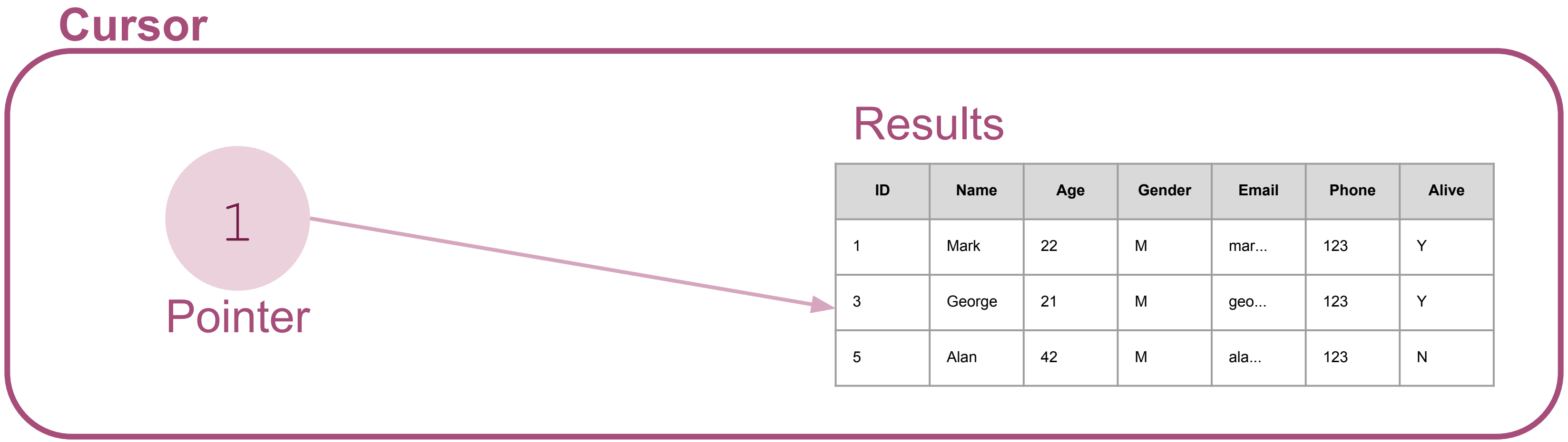
- To move onwards, we can call `moveToNext()`.



ID	Name	Age	Gender	Email	Phone	Alive
1	Mark	22	M	mar...	123	Y
3	George	21	M	geo...	123	Y
5	Alan	42	M	ala...	123	N

Databases: Reading Tasks / Cursors

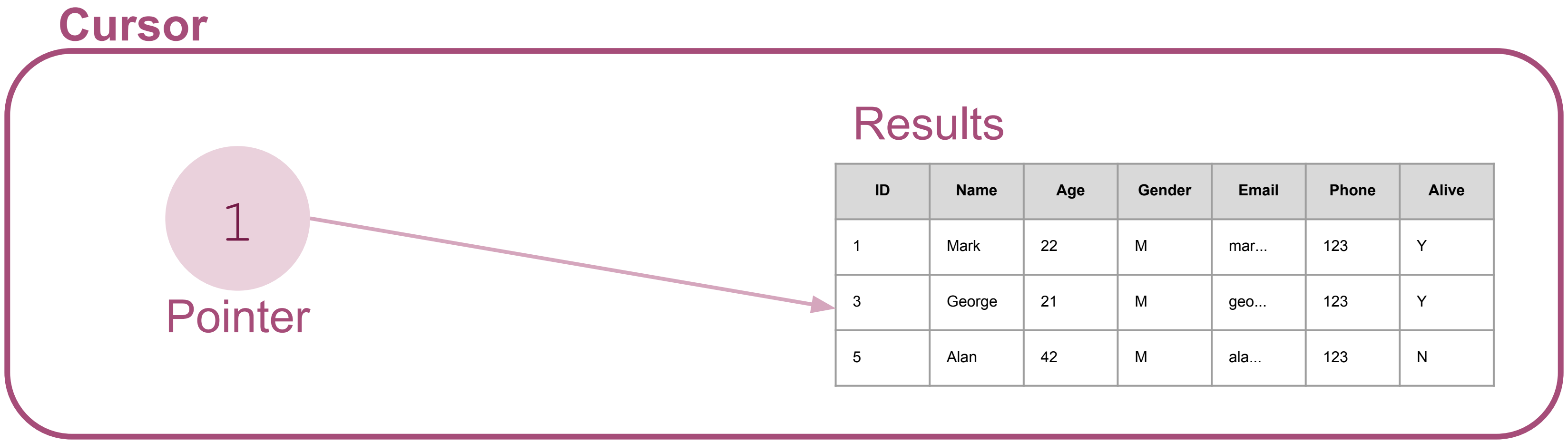
- To move onwards, we can call `moveToNext()`.



ID	Name	Age	Gender	Email	Phone	Alive
1	Mark	22	M	mar...	123	Y
3	George	21	M	geo...	123	Y
5	Alan	42	M	ala...	123	N

Databases: Reading Tasks / Cursors

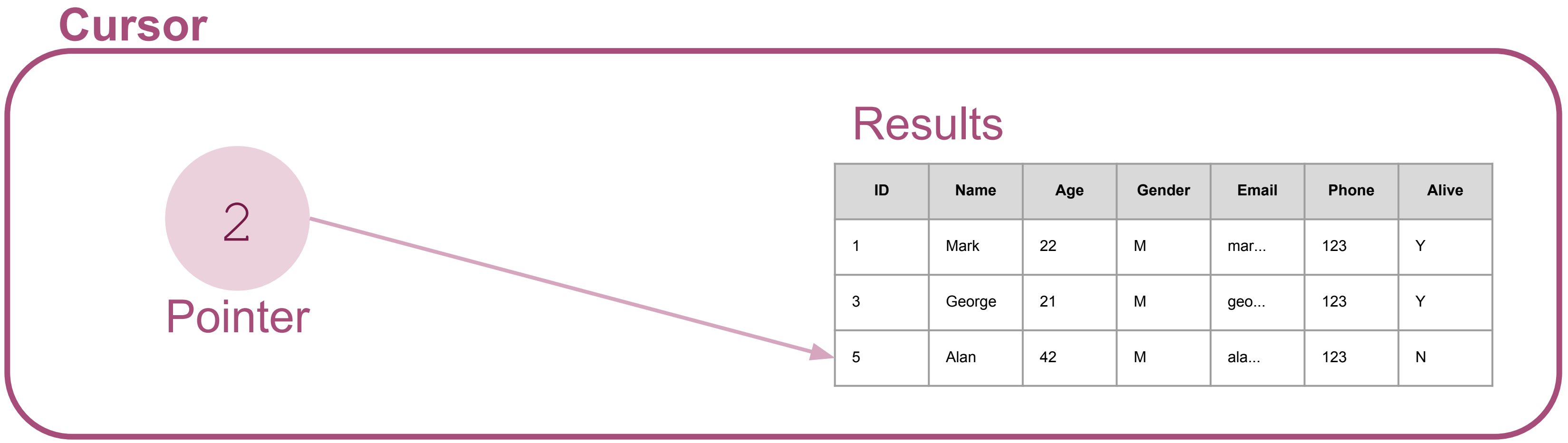
- If we call `moveToNext()` again...



ID	Name	Age	Gender	Email	Phone	Alive
1	Mark	22	M	mar...	123	Y
3	George	21	M	geo...	123	Y
5	Alan	42	M	ala...	123	N

Databases: Reading Tasks / Cursors

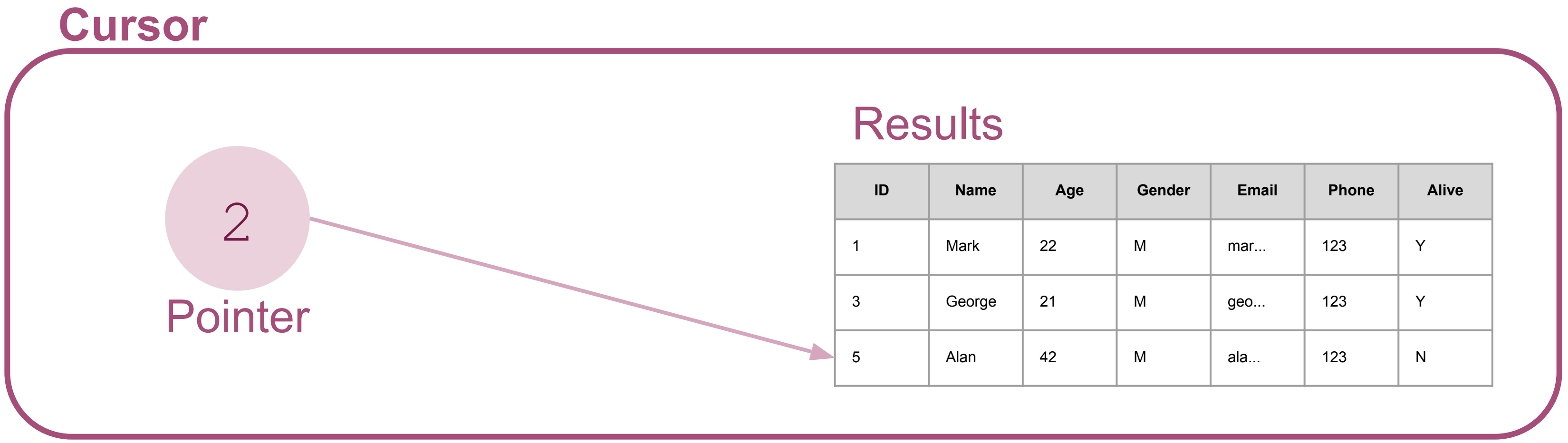
- If we call `moveToNext()` again...



ID	Name	Age	Gender	Email	Phone	Alive
1	Mark	22	M	mar...	123	Y
3	George	21	M	geo...	123	Y
5	Alan	42	M	ala...	123	N

Databases: Reading Tasks / Cursors

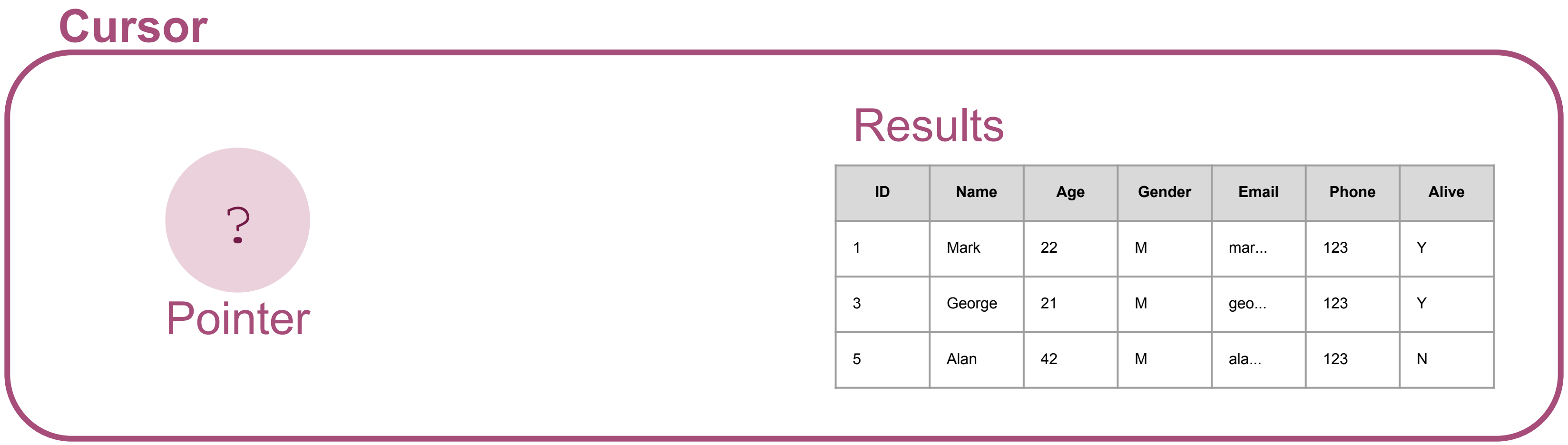
- If we call `moveToNext()` once more we run out of records, so it returns `false` to tell us that there wasn't a "next" record.



ID	Name	Age	Gender	Email	Phone	Alive
1	Mark	22	M	mar...	123	Y
3	George	21	M	geo...	123	Y
5	Alan	42	M	ala...	123	N

Databases: Reading Tasks / Cursors

- If we call `moveToNext()` once more we run out of records, so it returns `false` to tell us that there wasn’t a “next” record.



ID	Name	Age	Gender	Email	Phone	Alive
1	Mark	22	M	mar...	123	Y
3	George	21	M	geo...	123	Y
5	Alan	42	M	ala...	123	N

Coding Time

WHO'S AWESOME?

Next Week: Putting it Together

- We will put everything together into a **complete, working app**!

- Make sure you are up to date with everything we have done so far.
  - I will publish the “work so far” code in the middle of the week.

- Questions? The Slack is there for you!
  - **kcltechhq.slack.com → #android-programming**

Done!