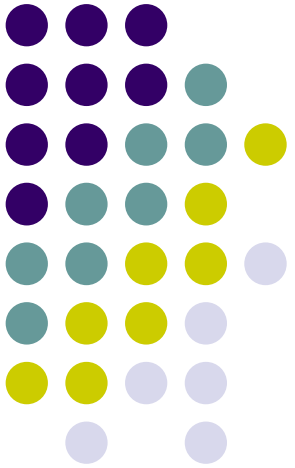
# Τεχνολογία και Προγραμματισμός Κινητών Συσκευών

# 6a



**Αποθήκευση δεδομένων**

**Ιωάννης Έλληνας**

# Αποθήκευση Δεδομένων

- Μηχανισμός Κοινής Προτίμησης (Shared Preferences) για μικρό όγκο δεδομένων.
- Αποθήκευση σε αρχείο για μεγάλο όγκο δεδομένων.
- Αποθήκευση σε βάση δεδομένων για μεγάλο όγκο με σχεσιακές δυνατότητες χειρισμών (ψάξιμο, ενημέρωση, κλπ).

**Αντικείμενο SharedPreferences**

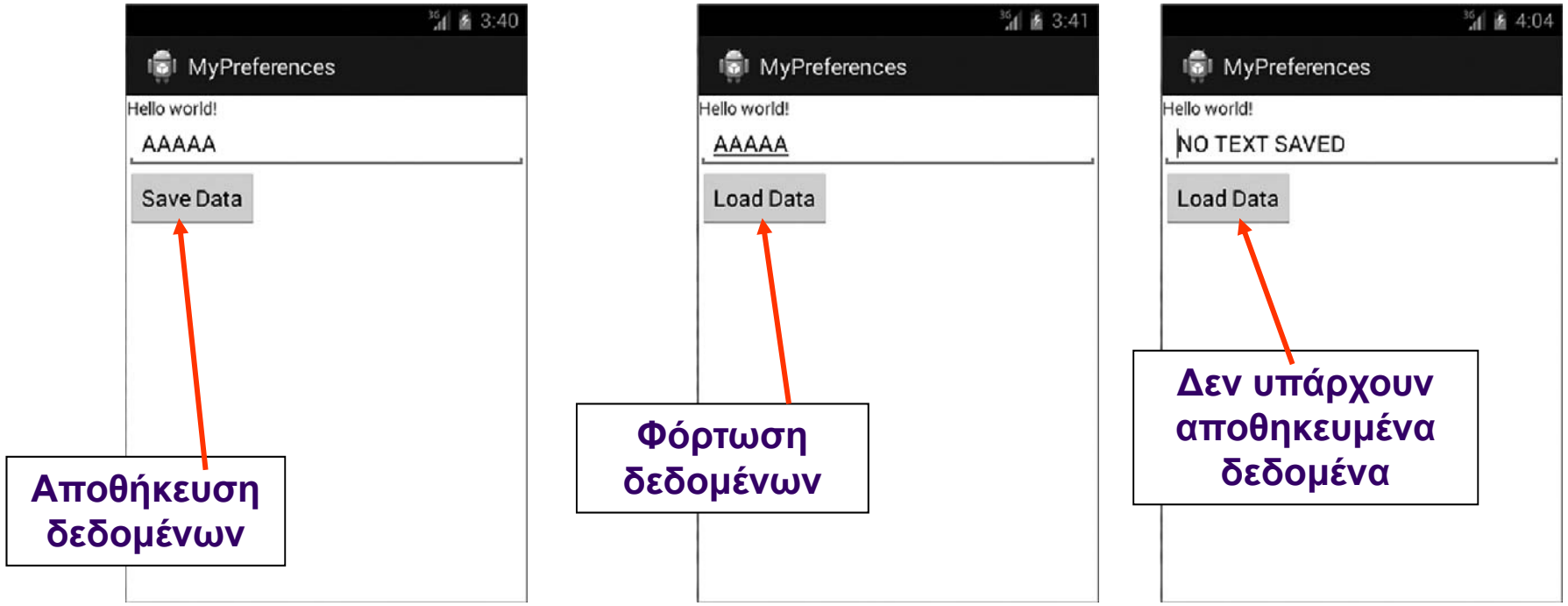
getPreferences()

getSharedPreferences()

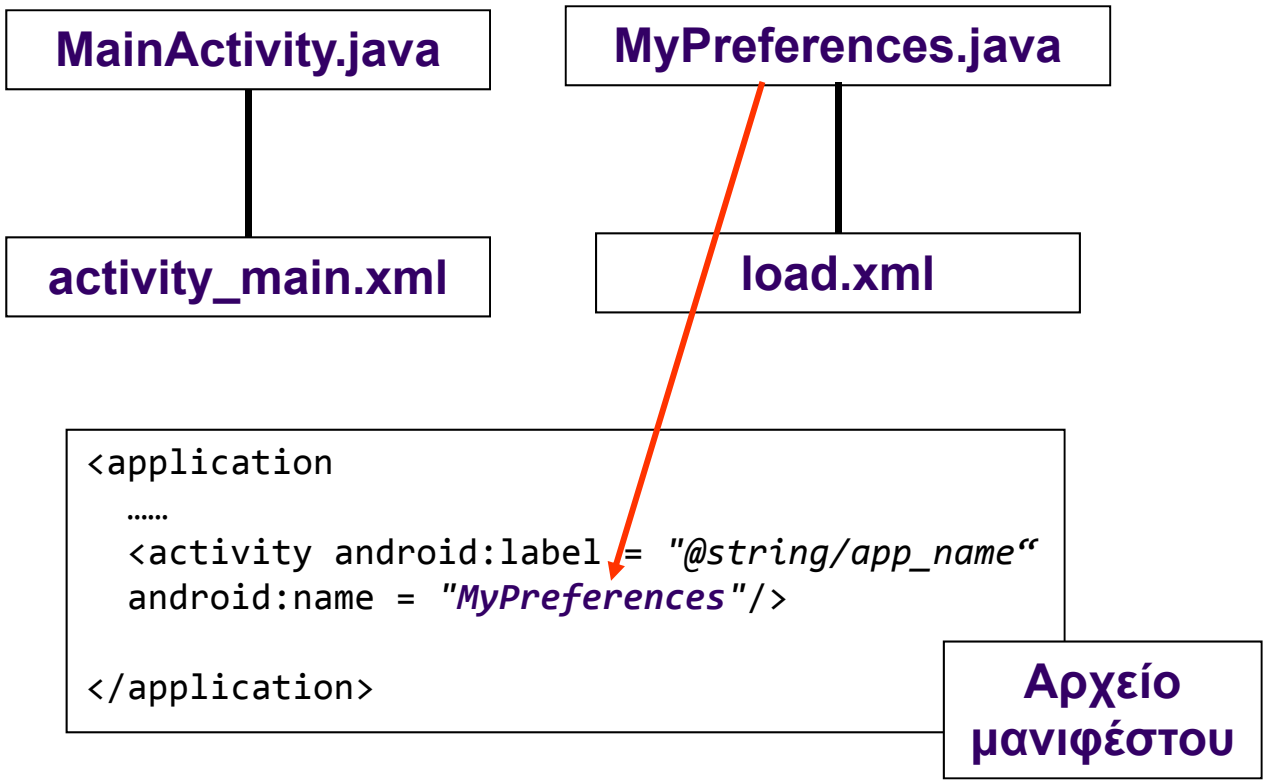
getDefaultSharedPreferences()

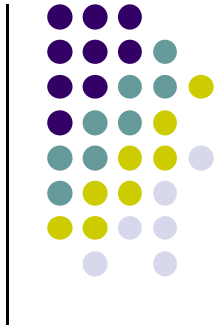
- **edit() – Τροποποίηση**
- **commit() – Αποθήκευση μέσω του editor**
- **clear() – Διαγραφή προτιμήσεων**
- **remove() – Διαγραφή του αντικειμένου προτιμήσεων**

# Παράδειγμα MyPreferences



# Δραστηριότητες





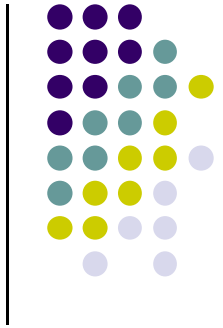
```
public class MainActivity extends Activity {
  EditText etxt;
  Button btn;
  SharedPreferences prefs;
  String fileName = "myPrefs";

  private static final String KEY_1 = "fontsize";
  private static final String KEY_2 = "textfield";

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    etxt = (EditText)findViewById(R.id.text1);
    btn = (Button)findViewById(R.id.btn1);
  }

  public void save(View v) {
    // create preferences object
    prefs = getSharedPreferences(fileName, MODE_PRIVATE);
    // open preferences editor
    SharedPreferences.Editor editor = prefs.edit();
    // save data
    editor.putFloat(KEY_1, etxt.getTextSize()); //in pixels
    editor.putString(KEY_2, etxt.getText().toString());
    editor.commit();
```

```
    // start new activity
    Intent intent = new Intent(this,
            MyPreferences.class);
    startActivity(intent);
    }
}
```

MODE_WORLD_READABLE

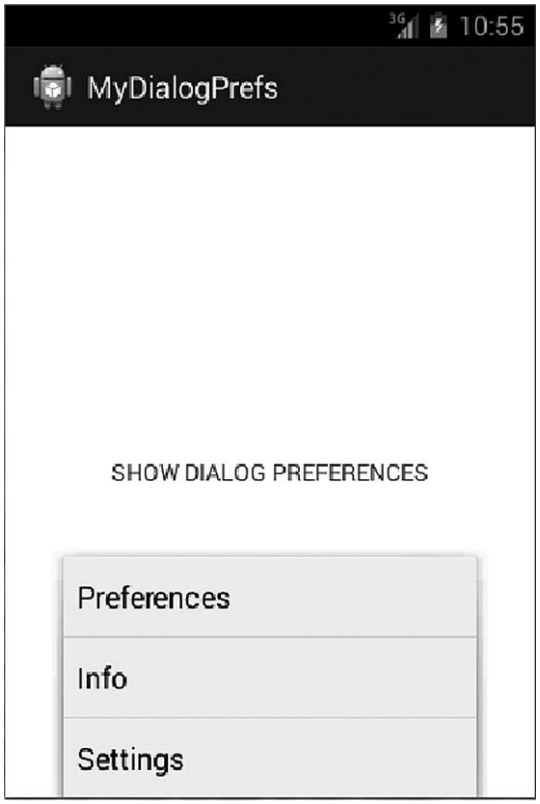
MODE_WORLD_WRITABLE

MODE_MULTI_PURPOSE



```
public class MyPreferences extends Activity {
  EditText etxt;
  Button btn;
  SharedPreferences prefs;
  String fileName = "myPrefs";
  private static final String KEY_1 = "fontsize";
  private static final String KEY_2 = "textfield";
  private static final float defValue_1 = 14;
  private static final String defValue_2 = "NO TEXT SAVED";
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.load);
    etxt = (EditText)findViewById(R.id.text2);
    btn = (Button)findViewById(R.id.btn2);
  }

  public void load(View v) {
    // create preferences object
    prefs = getSharedPreferences(fileName,MODE_PRIVATE);
    // get font size from file
    float fontSize = prefs.getFloat(KEY_1, defValue_1);
    etxt.setText(prefs.getString(KEY_2, defValue_2));
    etxt.setTextSize(fontSize);
  }
}
```

# Κοινή προτίμηση με διάλογο




Πρόγραμμα “MyDialogPrefs

MainActivity

**Menu:** Preferences
Info
Settings

showInfo()

ShowSettings.java

EditPreferences.java

Alert Dialog

Dialog Preferences
**preferences.xml**

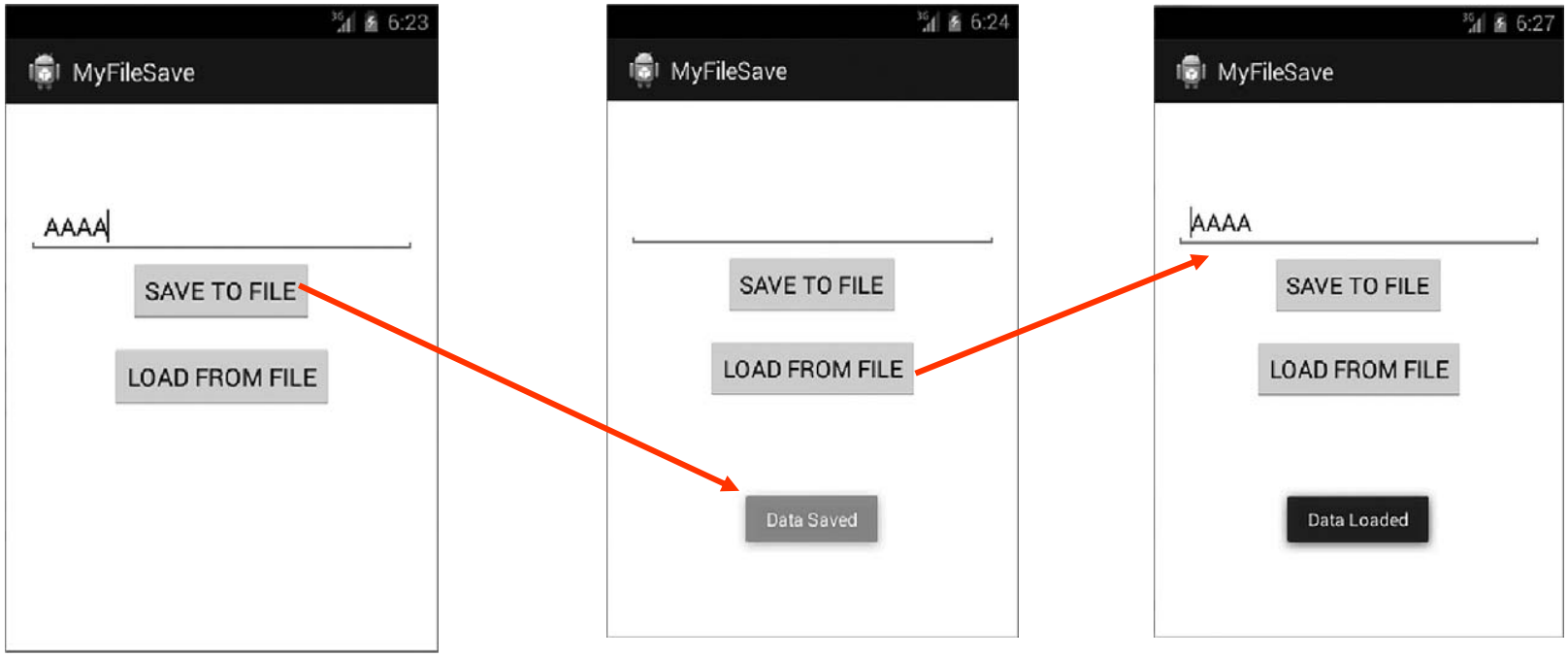
- @array/entries_fontsize
- @array/Entryvalues_fontsize
- @array/entries_color
- @array/entryvalues_color

# Αποθήκευση σε αρχείο

- Πολλές φορές είναι καλύτερα να αποθηκεύονται τα δεδομένα σε αρχείο.
- Το αρχείο μπορεί να δημιουργηθεί κατά την εκτέλεση της εφαρμογής ή να έχει δημιουργηθεί από την αρχή στον φάκελο res/raw και να ενσωματωθεί στο τελικό εκτελέσιμο αρχείο της εφαρμογής.

- ***Αποθήκευση δεδομένων κατά την εκτέλεση***
  - Χρησιμοποιούμε τις μεθόδους *openFileInput()* και *openFileOutput()* για να καταχωρήσουμε αντίστοιχα ένα *InputStream* ή ένα *OutputStream*.
  - Συνδέουμε τη ροή δεδομένων όπως γίνεται στην Java για είσοδο-έξοδο.
  - Πραγματοποιούμε την ανάγνωση ή την εγγραφή δεδομένων.
  - Με τη μέθοδο *close()* κλείνει η ροή δεδομένων.

# Παράδειγμα MyFileSave



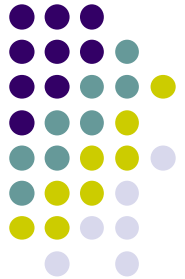
```
public class MainActivity extends Activity {
  Button btn1, btn2;
  EditText text;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    btn1 = (Button)findViewById(R.id.button1);
    btn2 = (Button)findViewById(R.id.button2);
    text =(EditText)findViewById(R.id.editText1);
  }
  public void Save(View v) {
    String str = text.getText().toString();
    try {
      FileOutputStream fOut = openFileOutput("temp.txt", MODE_PRIVATE);
      OutputStreamWriter outstr = new OutputStreamWriter(fOut);
      outstr.write(str);
      outstr.flush();
      outstr.close();
      Toast.makeText(this, "Data Saved", Toast.LENGTH_SHORT). show();
      text.setText("");
    }
    catch (IOException e) {
      e.printStackTrace();
    }
  }
```

MODE_WORLD_READABLE

MODE_WORLD_WRITABLE

MODE_APPEND

Εγγραφή



Ανάγνωση

```
  public void Load(View v) {
    try {
      FileInputStream fIn = openFileInput("temp.txt");
      InputStreamReader instr = new InputStreamReader(fIn);
      BufferedReader bread = new BufferedReader(instr);
      String str;
      StringBuilder buffer = new StringBuilder();
      while ((str = bread.readLine()) != null) {
        buffer.append(str+"\n");
      }
      fIn.close();
      text.setText(buffer.toString());
      Toast.makeText(this, "Data Loaded", Toast.LENGTH_SHORT). show();
    }
    catch (IOException ioe) {
      ioe.printStackTrace();
    }
  }
}
```

# Αποθήκευση σε εξωτερική μνήμη

- Η αποθήκευση ή η ανάκτηση δεδομένων μπορεί να γίνει σε ή από αρχεία τα οποία βρίσκονται σε εξωτερική μνήμη (κάρτα SD). Η διαφοροποίηση με την προηγούμενη περίπτωση είναι ότι τα δημιουργούμενα αρχεία μπορούν να προσπελαστούν από κάθε εφαρμογή.
- Για την αποθήκευση δεδομένων χρησιμοποιείται η κλάση *Environment.getExternalStorageDirectory()* για τη δημιουργία ενός αντικειμένου του επιθυμητού αρχείου στο ριζικό της εξωτερικής μνήμης.

```
String baseDir = Environment.getExternalStorageDirectory().getAbsolutePath();
File f = new File(baseDir, “temp.txt”);
FileOutputStream fOut = new FileOutputStream(f);

OutputStreamWriter outstr = new OutputStreamWriter(fOut);
outstr.write(str);
outstr.flush();
outstr.close();
```

- Η μέθοδος *getAbsolutePath()* επιστρέφει τη διαδρομή */mnt/sdcard* στον προσομοιωτή και τη διαδρομή */sdcard* στις κινητές συσκευές.
- Η ανάκτηση δεδομένων από αρχείο που βρίσκεται σε εξωτερική μνήμη πραγματοποιείται ως εξής:

```
String baseDir = Environment.getExternalStorageDirectory().getAbsolutePath();
File f = new File(baseDir, “temp.txt”);
FileInputStream fIn = new FileInputStream(f);
InputStreamReader instr = new InputStreamReader(fIn);
```

- Θα πρέπει να σημειωθεί ότι για να έχει πρόσβαση η εφαρμογή στην εξωτερική μνήμη, πρέπει να έχει άδεια πρόσβασης που να είναι δηλωμένη στο αρχείο μανιφέστου.

```
<uses-permission
  android:name = “android.permission.WRITE_EXTERNAL_STORAGE”>
</uses-permission>
```

- **Αποθήκευση στατικών δεδομένων**
  - **Τα στατικά δεδομένα βρίσκονται σε ένα αρχείο στον φάκελο *res/raw* και η εφαρμογή τα ενσωματώνει στο τελικά παραγόμενο εκτελέσιμο αρχείο.**
  - **Για την ανάγνωση των δεδομένων του αρχείου χρησιμοποιείται η μέθοδος *getResources()* και η μέθοδος *openRawResource()*.**

```
InputStream fIn = this.getResources().openRawResources(R.raw.temp);
InputStreamReader instr = new InputStreamReader(fIn);
BufferedReader bread = new BufferedReader(instr);
String str;
StringBuilder buffer = new StringBuilder();
while ((str = bread.readLine()) != null) {
  buffer.append(str);
}
fIn.close();
```

**Το ID του αρχείου *temp.txt* είναι το όνομα του αρχείου.**

# Αποθήκευση σε Βάση Δεδομένων

- SQLite (βιβλιοθήκη με πρόσβαση μέσω μεθόδων του λειτουργικού, μικρές απαιτήσεις σε μνήμη, χαρακτηριστικά της SQL).
- Βασικό στοιχείο ο πίνακας (Table)=σειρές και στήλες με πληροφορίες ορισμένου τύπου.
- Το σχηματικό (Database Schema) καθορίζει τον τύπο δεδομένων.
- Κάθε εγγραφή πραγματοποιείται σε μια σειρά.
- Μια ή περισσότερες στήλες έχουν το “Primary Key”, δηλαδή το κλειδί ταυτοποίησης κάθε σειράς.
- Κλάσεις του λειτουργικού για διαχείρισητης βάσης δεδομένων.
  - **SQLiteDatabase** – Προσαρμογέας εφαρμογής με βάση δεδομένων
  - **SQLiteOpenHelper** – Δημιουργία ή αναβάθμιση βάσης δεδομένων
  - **Cursor** – Πρόσβαση στα αποτελέσματα αναζήτησης
  - **ContentValues** – Εισαγωγή ή ανανέωση δεδομένων σε πίνακα (στήλη, τιμή)

# SQLiteDatabase

- ***insert()*** – Εισαγωγή εγγραφής ή σειράς σε έναν πίνακα.
- ***delete()*** – Διαγραφή μιας σειράς σε έναν πίνακα.
- ***update()*** – Ενημέρωση μιας σειράς σε έναν πίνακα με νέα δεδομένα.
- ***query()*** – Δημιουργεί μια ερώτηση στη βάση δεδομένων και λαμβάνει απάντηση μέσω του αντικειμένου του δρομέα (cursor object).
- ***execSQL()*** – Εκτελεί μια συγκεκριμένη εντολή της γλώσσας SQL η οποία δεν επιστρέφει δεδομένα.
- ***rawQuery()*** – Εκτελεί μια εντολή ερώτησης και επιστρέφει δεδομένα μέσω του δρομέα.

# SQLiteOpenHelper

***Σε μια εφαρμογή πρέπει να δημιουργηθεί μια υποκλάση αυτής με τις εξής δύο μεθόδους επανάκλησης:***

- ***onCreate()*** – Καλείται όταν πρέπει να δημιουργηθεί μια νέα βάση δεδομένων. Σαν παράμετρο δέχεται ένα αντικείμενο της προηγούμενης κλάσης για τη νέα βάση δεδομένων, η οποία αρχικοποιείται και στην οποία μπορεί να τοποθετηθούν οι αρχικές εγγραφές.
- ***onUpgrade()*** – Καλείται όταν η εφαρμογή περιέχει νεότερη βάση δεδομένων.

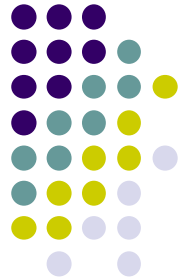
# Άλλες μέθοδοι της κλάσης

- ***onOpen()*** – Καλείται όταν ανοίγει η βάση δεδομένων.
- ***onClose()*** - Καλείται για να κλείσει η βάση δεδομένων.
- ***getReadableDatabase()*** – Χρησιμοποιείται για να δημιουργήσει ή ανοίξει μια βάση μόνο για ανάγνωση. Επιστρέφει ένα αντικείμενο της προηγούμενης κλάσης.
- ***getWritableDatabase()*** – Χρησιμοποιείται για να δημιουργήσει ή ανοίξει μια βάση μόνο για ανάγνωση και εγγραφή. Επιστρέφει ένα αντικείμενο της προηγούμενης κλάσης.

# Cursor

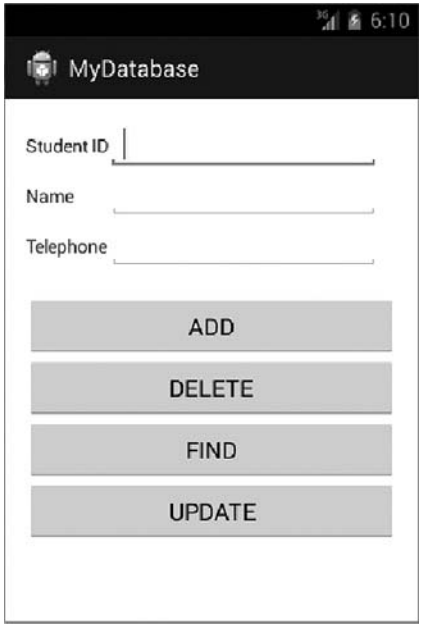
- ***moveToFirst()*** – Μετακινείται στην πρώτη σειρά που προέκυψε από την αναζήτηση.
- ***moveToLast()*** - Μετακινείται στην τελευταία σειρά.
- ***moveToNext()*** - Μετακινείται στην επόμενη σειρά.
- ***move()*** – Μετακινείται με ορισμένη μετατόπιση.
- ***getCount()*** – Επιστρέφει τον αριθμό των σειρών που προέκυψαν μετά από μια αναζήτηση.
- ***get<type>()*** – Επιστρέφει την τιμή του συγκεκριμένου τύπου <type> που περιέχεται στη συγκεκριμένη στήλη στην τρέχουσα σειρά του δρομέα (*getString()*, *getInt()*, *getFloat()*, *getShort()*, *getDouble()*).

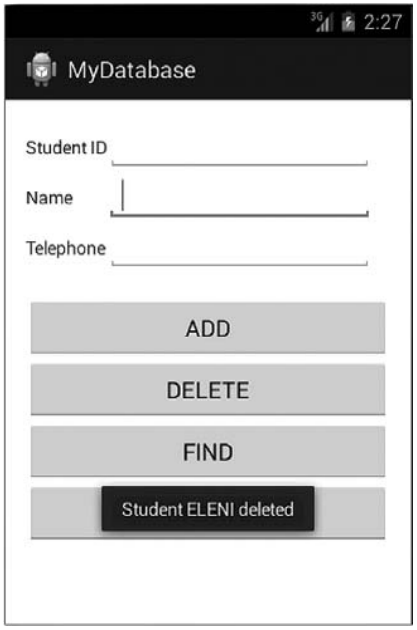
# Παράδειγμα - MyDatabase

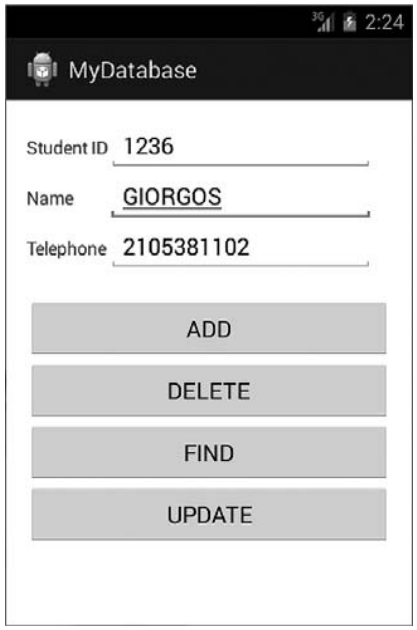


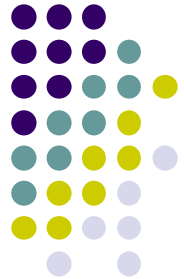
- Βάση δεδομένων με τρεις στήλες:
  - Όνομα σπουδαστή
  - Αριθμό μητρώου
  - Τηλέφωνο
- Αποθηκεύεται στο φάκελο:
  *data/data/<package_name>/databases*

# Διεπαφή χρήστη



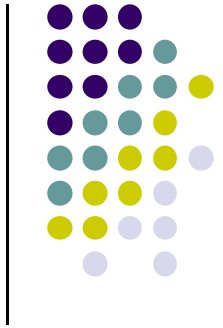





- Για να δούμε ένα αρχείο database στο Android Studio ανοίγουμε τον Device File Explorer → data/data/package name (gr.mybook.mydatabase) → databases → MyDB.db.
- Κάνουμε δεξί κλικ και Save As όπου θέλουμε.
- Εγκαθιστούμε το πρόγραμμα “SQLite Browser” (http://sqlitebrowser.org/).
- Με αυτό διαβάζουμε το αρχείο database.

- Μπορούμε να δημιουργήσουμε τη δική μας database στον φάκελο res/assets/mydb.db με τη βοήθεια αυτού του προγράμματος. Στη συνέχεια να δημιουργήσουμε το αρχείο mydb.db στο path data/data/package name (gr.mybook.mydatabase) → databases → mydb.db μέσα στην μέθοδο onCreate() της εφαρμογής μας και τέλος να αντιγράψουμε τα δεδομένα της έτοιμης βάσης σε αυτό (βιβλίο σελίδα 262).

# Εργασία



- Να δημιουργηθεί το project MyData το οποίο θα δημιουργεί μια οθόνη με δύο πεδία εισαγωγής κειμένου και ένα πλήκτρο με όνομα “Save”.
- Θα εισάγουμε ένα κείμενο στο πεδίο 1 και θα πατάμε το πλήκτρο. Τα δεδομένα του πεδίου 1 θα αποθηκεύονται με μηχανισμό κοινής προτίμησης, θα μηδενίζεται το πεδίο 1, ενώ το πλήκτρο θα αλλάζει ονομασία σε “Load”.
- Πατώντας το πλήκτρο “Load” θα ανακτώνται τα αποθηκευμένα δεδομένα και θα απεικονίζονται στο πεδίο 2, ενώ το πλήκτρο θα αλλάζει ονομασία σε “Save”.