```
 1 package tirzad.starunique.wordguess;
 2
 3 import android.content.Intent;
 4 import android.support.v7.app.AppCompatActivity;
 5 import android.os.Bundle;
 6 import android.view.View;
 7 import android.widget.Button;
 8
 9 public class MainActivity extends AppCompatActivity {
10
11     Button addButton;
12     Button studyButton;
13     Button viewButton;
14
15     @Override
16     protected void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         setContentView(R.layout.activity_main);
19
20         addButton = (Button) findViewById(R.id.btn_add);
21         studyButton = (Button) findViewById(R.id.btn_study
   );
22         viewButton = (Button) findViewById(R.id.btn_view);
23
24         addButton.setOnClickListener(new View.
   OnClickListener() {
25             @Override
26             public void onClick(View view) {
27                 startActivity(new Intent(getBaseContext(),
    AddEditActivity.class));
28             }
29         });
30         studyButton.setOnClickListener(new View.
   OnClickListener() {
31             @Override
32             public void onClick(View view) {
33                 startActivity(new Intent(getBaseContext(),
    StudyActivity.class));
34             }
35         });
36         viewButton.setOnClickListener(new View.
   OnClickListener() {
37             @Override
38             public void onClick(View view) {
39                 startActivity(new Intent(getBaseContext(),
```

```
39  ViewActivity.class));
40              }
41          });
42
43      }
44  }
45
```

```
 1 package tirzad.starunique.wordguess;
 2
 3 import android.app.LoaderManager;
 4 import android.content.ContentUris;
 5 import android.content.ContentValues;
 6 import android.content.CursorLoader;
 7 import android.content.DialogInterface;
 8 import android.content.Intent;
 9 import android.content.Loader;
10 import android.database.Cursor;
11 import android.net.Uri;
12 import android.os.Bundle;
13 import android.support.design.widget.FloatingActionButton;
14 import android.support.v7.app.AlertDialog;
15 import android.support.v7.app.AppCompatActivity;
16 import android.view.Menu;
17 import android.view.MenuItem;
18 import android.view.View;
19 import android.widget.AdapterView;
20 import android.widget.ListView;
21
22 import tirzad.starunique.wordguess.data.WordContract.
   WordEntry;
23
24 public class ViewActivity extends AppCompatActivity
   implements LoaderManager.LoaderCallbacks<Cursor>{
25
26     WordCursorAdapter mCursorAdapter;
27     static final String[] PROJECTION = {
28             WordEntry._ID,
29             WordEntry.COLUMN_WORD,
30             WordEntry.COLUMN_SENTENCE
31     };
32     @Override
33     protected void onCreate(Bundle savedInstanceState) {
34         super.onCreate(savedInstanceState);
35         setContentView(R.layout.activity_view);
36
37         FloatingActionButton fab = (FloatingActionButton)
   findViewById(R.id.fab);
38         fab.setOnClickListener(new View.OnClickListener()
   {
39             @Override
40             public void onClick(View view) {
41                 startActivity(new Intent(ViewActivity.this
```

```
41 , AddEditActivity.class));
42             }
43         });
44 
45         ListView wordListView = (ListView) findViewById(R.
   id.lvItems);
46         View emptyView = findViewById(R.id.empty_view);
47         wordListView.setEmptyView(emptyView);
48 
49         mCursorAdapter = new WordCursorAdapter(this, null)
   ;
50         wordListView.setAdapter(mCursorAdapter);
51 
52         wordListView.setOnItemClickListener(new
   AdapterView.OnItemClickListener() {
53             @Override
54             public void onItemClick(AdapterView<?>
   adapterView, View view, int position, long id) {
55                 Intent intent = new Intent(ViewActivity.
   this, AddEditActivity.class);
56                 intent.setData(ContentUris.withAppendedId(
   WordEntry.CONTENT_URI, id));
57                 startActivity(intent);
58             }
59         });
60 
61         getLoaderManager().initLoader(1, null, this);
62     }
63 
64     @Override
65     public boolean onCreateOptionsMenu(Menu menu) {
66         getMenuInflater().inflate(R.menu.view_menu, menu);
67         return true;
68     }
69 
70     @Override
71     public boolean onOptionsItemSelected(MenuItem item) {
72 
73         switch (item.getItemId()) {
74 
75             case R.id.action_insert_data:
76                 insertWord();
77                 return true;
78 
79             case R.id.action_delete_all_entries:
```

```
 80                 showAllWordsDeletionConfirmation();
 81                 return true;
 82         }
 83         return super.onOptionsItemSelected(item);
 84     }
 85
 86     private void deleteAllWords() {
 87         getContentResolver().delete(WordEntry.CONTENT_URI
    , null, null);
 88     }
 89
 90     private void insertWord() {
 91         ContentValues values = new ContentValues();
 92         values.put(WordEntry.COLUMN_WORD, "analysis");
 93         values.put(WordEntry.COLUMN_SENTENCE, "Further
    analysis of the data is needed.");
 94         Uri newUri = getContentResolver().insert(
    WordEntry.CONTENT_URI, values);
 95     }
 96
 97     @Override
 98     public Loader<Cursor> onCreateLoader(int i, Bundle
    bundle) {
 99         return new CursorLoader(this,
100                 WordEntry.CONTENT_URI,
101                 PROJECTION,
102                 null,
103                 null,
104                 null);
105     }
106
107     @Override
108     public void onLoadFinished(Loader<Cursor> loader,
    Cursor cursor) {
109         mCursorAdapter.swapCursor(cursor);
110     }
111
112     @Override
113     public void onLoaderReset(Loader<Cursor> loader) {
114         mCursorAdapter.swapCursor(null);
115     }
116
117     private void showAllWordsDeletionConfirmation() {
118
119         AlertDialog.Builder builder = new AlertDialog.
```

```
119 Builder(this);
120         builder.setMessage("Delete all words in the
    database?")
121                 .setPositiveButton("DELETE ALL", new
    DialogInterface.OnClickListener() {
122                     @Override
123                     public void onClick(DialogInterface
    dialogInterface, int i) {
124                         deleteAllWords();
125                     }
126                 })
127                 .setNegativeButton("CANCEL", new
    DialogInterface.OnClickListener() {
128                     @Override
129                     public void onClick(DialogInterface
    dialogInterface, int i) {
130                         if (dialogInterface != null)
131                             dialogInterface.dismiss();
132                     }
133                 });
134         AlertDialog alertDialog = builder.create();
135         alertDialog.show();
136     }
137 }
138
```

```java
package tirzad.starunique.wordguess;

import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import java.security.SecureRandom;

import tirzad.starunique.wordguess.data.SentenceHandling;
import tirzad.starunique.wordguess.data.WordContract;
import tirzad.starunique.wordguess.data.WordDbHelper;

public class StudyActivity extends AppCompatActivity {

    WordDbHelper mWordDbHelper = new WordDbHelper(this);

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

        loadANewSentence();

        Button skipButton = (Button) findViewById(R.id.skip_button);
        skipButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                loadANewSentence();
            }
        });



    }

    private void loadANewSentence() {
        SQLiteDatabase database = mWordDbHelper.getReadableDatabase();
```

```
43
44          Cursor cursor = getContentResolver().query(
    WordContract.WordEntry.CONTENT_URI,
45                  null,
46                  null,
47                  null,
48                  null,
49                  null);
50          if (cursor.moveToNext()) {
51              SecureRandom randomNumbers = new SecureRandom(
    );
52              int randomWord = randomNumbers.nextInt(cursor.
    getCount());
53              cursor.moveToPosition(randomWord);
54
55              final TextView textViewWord = (TextView)
    findViewById(R.id.textview_word);
56              textViewWord.setText(String.valueOf(cursor.
    getString(1)));
57
58              final TextView textViewSentence = (TextView)
    findViewById(R.id.textview_sentence);
59              textViewSentence.setText(SentenceHandling.
    changeToStars(String.valueOf(cursor.getString(2))));
60
61              cursor.close();
62
63              Button hintButton = (Button) findViewById(R.id
    .hint_button);
64              hintButton.setOnClickListener(new View.
    OnClickListener() {
65                  @Override
66                  public void onClick(View view) {
67                      textViewSentence.setText(
68                              SentenceHandling.giveHintWord(
    String.valueOf(textViewSentence.getText())));
69                  }
70              });
71
72              final EditText guessedWord = (EditText)
    findViewById(R.id.guessed_word_edit_text);
73              Button checkButton = (Button) findViewById(R.
    id.check_button);
74              checkButton.setOnClickListener(new View.
    OnClickListener() {
```

```
75                  @Override
76                  public void onClick(View view) {
77                      textViewSentence.setText(
78                              SentenceHandling.
   checkGuessedWord(String.valueOf(textViewSentence.getText(
   )),
79                                      guessedWord.getText()
   .toString().trim()));
80                      guessedWord.setText("");
81                  }
82              });
83          }
84
85      }
86
87
88  }
89
```

```java
package tirzad.starunique.wordguess;

import android.app.LoaderManager;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.CursorLoader;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.Loader;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.support.v4.app.NavUtils;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import tirzad.starunique.wordguess.data.WordContract;

public class AddEditActivity extends AppCompatActivity implements LoaderManager.LoaderCallbacks<Cursor> {

    private EditText mWordEditText;
    private EditText mSentenceEditText;
    private String mSelection;
    private String[] mSelectionArgs;
    private Uri mCurrentWordUri;

    private boolean mWordHasChanged = false;
    private View.OnTouchListener mTouchListener = new View.OnTouchListener() {
        @Override
        public boolean onTouch(View view, MotionEvent motionEvent) {
            mWordHasChanged = true;
            return false;
        }
    };

    @Override
```

```
43      protected void onCreate(Bundle savedInstanceState) {
44          super.onCreate(savedInstanceState);
45          setContentView(R.layout.activity_add);
46
47          Intent intent = getIntent();
48          mCurrentWordUri = intent.getData();
49          if (mCurrentWordUri != null) {
50              setTitle("Edit Word");
51              mSelection = WordContract.WordEntry._ID + "=?"
    ;
52              mSelectionArgs = new String[]{String.valueOf(
    ContentUris.parseId(mCurrentWordUri))};
53
54              getLoaderManager().initLoader(1, null, this);
55          } else {
56              setTitle("Add Word");
57              invalidateOptionsMenu();
58          }
59
60          mWordEditText = (EditText) findViewById(R.id.
    edit_word);
61          mSentenceEditText = (EditText) findViewById(R.id.
    edit_sentence);
62
63          mWordEditText.setOnTouchListener(mTouchListener);
64          mSentenceEditText.setOnTouchListener(
    mTouchListener);
65
66      }
67
68      @Override
69      public boolean onCreateOptionsMenu(Menu menu) {
70          getMenuInflater().inflate(R.menu.editor_menu, menu
    );
71          return true;
72      }
73
74      @Override
75      public boolean onOptionsItemSelected(MenuItem item) {
76          switch (item.getItemId()) {
77
78              case R.id.action_save:
79                  saveWord();
80                  return true;
81
```

```
 82              case R.id.action_delete_word:
 83                  showDeleteConfirmationDialog();
 84                  return true;
 85
 86              // R.id.home wouldn't trigger an error but
    you need android before that to work
 87              case android.R.id.home:
 88
 89                  if (!mWordHasChanged) {
 90                      NavUtils.navigateUpFromSameTask(
    AddEditActivity.this);
 91                      return true;
 92                  }
 93
 94                  DialogInterface.OnClickListener
    discardButtonOnClickListener =
 95                          new DialogInterface.
    OnClickListener() {
 96                              @Override
 97                              public void onClick(
    DialogInterface dialogInterface, int i) {
 98 //                                  NavUtils.
    navigateUpFromSameTask(AddEditActivity.this);
 99                                  finish();
100                              }
101                          };
102
103                  showUnsavedChangesDialog(
    discardButtonOnClickListener);
104                  return true;
105          }
106          return super.onOptionsItemSelected(item);
107      }
108
109      private void deleteWord() {
110
111          if (mCurrentWordUri != null) {
112              getContentResolver().delete(
113                      /*WordContract.WordEntry.CONTENT_URI
    */ mCurrentWordUri,
114                      null/*mSelection*/,
115                      null/*mSelectionArgs*/);
116          }
117          finish();
118      }
```

```
119
120     private void saveWord() {
121
122         String wordString = mWordEditText.getText().
    toString().trim();
123         String sentenceString = mSentenceEditText.getText
    ().toString().trim();
124
125         ContentValues values = new ContentValues();
126         values.put(WordContract.WordEntry.COLUMN_WORD,
    wordString);
127         values.put(WordContract.WordEntry.COLUMN_SENTENCE
    , sentenceString);
128
129         try {
130             if (mCurrentWordUri != null) {
131                 getContentResolver().update(
    mCurrentWordUri, values,
132                         mSelection, mSelectionArgs);
133             } else {
134                 Uri newUri = getContentResolver().insert(
    WordContract.WordEntry.CONTENT_URI, values);
135                 if (newUri == null) {
136                     Toast.makeText(this, "Insertion
    failed", Toast.LENGTH_SHORT).show();
137                 } else {
138                     Toast.makeText(this, "Word Saved",
    Toast.LENGTH_SHORT).show();
139                 }
140             }
141             finish();
142         } catch (IllegalArgumentException e) {
143             Toast.makeText(this, "Please fill all fields
    .", Toast.LENGTH_SHORT).show();
144         }
145     }
146
147     @Override
148     public Loader<Cursor> onCreateLoader(int i, Bundle
    bundle) {
149         String[] projection = {
150                 WordContract.WordEntry._ID,
151                 WordContract.WordEntry.COLUMN_WORD,
152                 WordContract.WordEntry.COLUMN_SENTENCE};
153
```

```
154         return new CursorLoader(this,
155                 WordContract.WordEntry.CONTENT_URI /*
    mCurrentWordUri*/,
156                 projection,
157                 mSelection,
158                 mSelectionArgs,
159                 null);
160     }
161
162     @Override
163     public void onLoadFinished(Loader<Cursor> loader,
    Cursor cursor) {
164
165         if (cursor.moveToNext()) {
166 //            cursor.moveToNext();
167             mWordEditText.setText(cursor.getString(
168                     cursor.getColumnIndexOrThrow(
    WordContract.WordEntry.COLUMN_WORD)));
169             mSentenceEditText.setText(cursor.getString(
170                     cursor.getColumnIndexOrThrow(
    WordContract.WordEntry.COLUMN_SENTENCE)));
171         }
172
173     }
174
175     @Override
176     public void onLoaderReset(Loader<Cursor> loader) {
177         mWordEditText.setText("");
178         mSentenceEditText.setText("");
179     }
180
181     private void showUnsavedChangesDialog(
182             DialogInterface.OnClickListener
    discardButtonClickListener) {
183
184         AlertDialog.Builder builder = new AlertDialog.
    Builder(this);
185         builder.setMessage("Discard your changes and quit
     editing?")
186                 .setPositiveButton("DISCARD",
    discardButtonClickListener)
187                 .setNegativeButton("KEEP EDITING", new
    DialogInterface.OnClickListener() {
188                     @Override
189                     public void onClick(DialogInterface
```

```
189 dialogInterface, int i) {
190                         if (dialogInterface != null) {
191                             dialogInterface.dismiss();
192                         }
193                     }
194                 });
195 
196         AlertDialog alertDialog = builder.create();
197         alertDialog.show();
198     }
199 
200     @Override
201     public void onBackPressed() {
202 
203         if (!mWordHasChanged) {
204             super.onBackPressed();
205             return;
206         }
207 
208         DialogInterface.OnClickListener
    discardButtonOnClickListener
209                 = new DialogInterface.OnClickListener() {
210             @Override
211             public void onClick(DialogInterface
    dialogInterface, int i) {
212                 finish();
213             }
214         };
215 
216         showUnsavedChangesDialog(
    discardButtonOnClickListener);
217     }
218 
219     @Override
220     public boolean onPrepareOptionsMenu(Menu menu) {
221         super.onPrepareOptionsMenu(menu);
222 
223         if (mCurrentWordUri == null) {
224             MenuItem menuItem = menu.findItem(R.id.
    action_delete_word);
225             menuItem.setVisible(false);
226         }
227 
228         return true;
229     }
```

```
230
231     private void showDeleteConfirmationDialog() {
232
233         AlertDialog.Builder builder = new AlertDialog.
    Builder(this);
234         builder.setMessage("Delete Word?")
235                 .setPositiveButton("DELETE", new
    DialogInterface.OnClickListener() {
236                     @Override
237                     public void onClick(DialogInterface
    dialogInterface, int i) {
238                         deleteWord();
239                     }
240                 })
241                 .setNegativeButton("CANCEL", new
    DialogInterface.OnClickListener() {
242                     @Override
243                     public void onClick(DialogInterface
    dialogInterface, int i) {
244                         if (dialogInterface != null) {
245                             dialogInterface.dismiss();
246                         }
247                     }
248                 });
249         AlertDialog alertDialog = builder.create();
250         alertDialog.show();
251     }
252 }
253
```

```java
package tirzad.starunique.wordguess;

import android.content.Context;
import android.database.Cursor;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CursorAdapter;
import android.widget.TextView;

import tirzad.starunique.wordguess.data.WordContract;

/**
 * Created by StarUnique on 18/09/2017.
 */

public class WordCursorAdapter extends CursorAdapter {


    public WordCursorAdapter(Context context, Cursor c/*, int flags*/) {
        super(context, c, 0/*flags*/);
    }

    @Override
    public View newView(Context context, Cursor cursor, ViewGroup viewGroup) {
        return LayoutInflater.from(context).inflate(R.layout.list_item, viewGroup, false);
    }

    @Override
    public void bindView(View view, Context context, Cursor cursor) {

        TextView lvWord = (TextView) view.findViewById(R.id.lv_word);
        TextView lvSentence = (TextView) view.findViewById(R.id.lv_sentence);

        lvWord.setText(cursor.getString(cursor.getColumnIndexOrThrow(WordContract.WordEntry.COLUMN_WORD)));
        lvSentence.setText(cursor.getString(cursor.getColumnIndexOrThrow(WordContract.WordEntry.
```

```
36 COLUMN_SENTENCE));
37
38
39      }
40 }
41
```

```java
package tirzad.starunique.wordguess.data;

import android.net.Uri;
import android.provider.BaseColumns;

/**
 * Created by StarUnique on 13/09/2017.
 */

public class WordContract {

    public WordContract() {
    }
    public static final String CONTENT_AUTHORITY = "tirzad.starunique.wordguess";
    public static final Uri BASE_CONTENT_URI = Uri.parse("content://" + CONTENT_AUTHORITY);
    public static final String PATH_WORDS = "words";

    public static final class WordEntry implements BaseColumns {

        public static final Uri CONTENT_URI = Uri.withAppendedPath(BASE_CONTENT_URI, PATH_WORDS);

        public static final String TABLE_NAME = "words";

        public static final String _ID = BaseColumns._ID;
        public static final String COLUMN_WORD = "word";
        public static final String COLUMN_SENTENCE = "sentence";
        public static final String COLUMN_DATE = "date";
        public static final String COLUMN_READABLE = "readable";

        public static final String DATE_DEFAULT = "13/09/2017";

        public static final String READABLE_DEFAULT = "true";


    }
}

```

```java
package tirzad.starunique.wordguess.data;

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

import tirzad.starunique.wordguess.data.WordContract.WordEntry;
/**
 * Created by StarUnique on 13/09/2017.
 */

public class WordDbHelper extends SQLiteOpenHelper {

    public static final String DATA_BASE_NAME = "notebook.db";
    public static final int DATA_BASE_VERSION = 1;
//    CREATE TABLE words(_ID INTEGER PRIMARY KEY AUTOINCREMENT, word TEXT NOT NULL, sentence TEXT NOT NULL,
// date TEXT NOT NULL DEFAULT 13/09/2017

    public static final String SQL_CREATE_WORDS_TABLE =
            "CREATE TABLE " +
                    WordEntry.TABLE_NAME + "(" +
                    WordEntry._ID + " INTEGER PRIMARY KEY AUTOINCREMENT," +
                    WordEntry.COLUMN_WORD + " TEXT NOT NULL," +
                    WordEntry.COLUMN_SENTENCE + " TEXT NOT NULL," +
                    WordEntry.COLUMN_DATE + " TEXT NOT NULL DEFAULT " + "today" + "," +
                    WordEntry.COLUMN_READABLE + " TEXT NOT NULL DEFAULT " + WordEntry.READABLE_DEFAULT +
            ");";

    public static final String SQL_DELETE_WORDS_TABLE =
            "DROP TABLE IF EXISTS " + WordEntry.TABLE_NAME;

    public WordDbHelper(Context context) {
        super(context, DATA_BASE_NAME, null, DATA_BASE_VERSION);
    }
    @Override
```

```
36      public void onCreate(SQLiteDatabase db) { db.execSQL(
    SQL_CREATE_WORDS_TABLE);}
37
38      @Override
39      public void onUpgrade(SQLiteDatabase db, int i, int i1
    ) {
40          db.execSQL(SQL_DELETE_WORDS_TABLE);
41          onCreate(db);
42      }
43  }
44
```

```java
package tirzad.starunique.wordguess.data;

import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.support.annotation.Nullable;
import android.text.TextUtils;
import android.util.Log;
/**
 * Created by StarUnique on 16/09/2017.
 */

public class WordProvider extends ContentProvider {

    public static final int WORDS = 200;
    public static final int WORDS_ID = 201;

    public static final String LOG_TAG = WordProvider.class.getSimpleName();
    public static final UriMatcher sUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);

    static {
        sUriMatcher.addURI(WordContract.CONTENT_AUTHORITY, WordContract.PATH_WORDS, WORDS);
        sUriMatcher.addURI(WordContract.CONTENT_AUTHORITY, WordContract.PATH_WORDS + "/#", WORDS_ID);
    }

    private WordDbHelper mDbHelper;

    @Override
    public boolean onCreate() {
        mDbHelper = new WordDbHelper(getContext());
        return true;
    }

    @Nullable
    @Override
    public Cursor query(Uri uri, String[] projection, String selection,
```

```
41                          String[] selectionArgs, String
   sortOrder) {
42
43          SQLiteDatabase database = mDbHelper.
   getReadableDatabase();
44          Cursor cursor;
45
46          int match = sUriMatcher.match(uri);
47          switch (match) {
48              case WORDS:
49                  cursor = database.query(WordContract.
   WordEntry.TABLE_NAME,
50                          projection,
51                          selection,
52                          selectionArgs,
53                          null,
54                          null,
55                          sortOrder);
56                  break;
57              case WORDS_ID:
58                  selection = WordContract.WordEntry._ID +
   "=?";
59                  selectionArgs = new String[]{String.
   valueOf(ContentUris.parseId(uri))};
60                  cursor = database.query(WordContract.
   WordEntry.TABLE_NAME,
61                          projection,
62                          selection,
63                          selectionArgs,
64                          null,
65                          null,
66                          sortOrder);
67                  break;
68              default:
69                  throw new IllegalArgumentException("Cannot
    query unknown Uri " + uri);
70          }
71          cursor.setNotificationUri(getContext().
   getContentResolver(),uri);
72          return cursor;
73      }
74
75      @Nullable
76      @Override
77      public String getType(Uri uri) {
```

```
 78          return null;
 79      }
 80
 81      @Nullable
 82      @Override
 83      public Uri insert(Uri uri, ContentValues
    contentValues) {
 84          int match = sUriMatcher.match(uri);
 85          switch (match) {
 86              case WORDS:
 87                  return insertPet(uri, contentValues);
 88              default:
 89                  throw new IllegalArgumentException("
    Insertion is not supported for " + uri);
 90          }
 91      }
 92
 93      private Uri insertPet(Uri uri, ContentValues values)
    {
 94
 95          String word = values.getAsString(WordContract.
    WordEntry.COLUMN_WORD);
 96          if (TextUtils.isEmpty(word))
 97              throw new IllegalArgumentException("Word
    cannot be empty");
 98          String sentence = values.getAsString(WordContract
    .WordEntry.COLUMN_SENTENCE);
 99          if (TextUtils.isEmpty(sentence))
100              throw new IllegalArgumentException("Sentence
    cannot be empty");
101
102          SQLiteDatabase database = mDbHelper.
    getWritableDatabase();
103          long id = database.insert(WordContract.WordEntry.
    TABLE_NAME, null, values);
104
105          if (id == -1) {
106              Log.e(LOG_TAG, "Failed to insert row for " +
    uri);
107              return null;
108          }
109          getContext().getContentResolver().notifyChange(
    uri, null);
110          return ContentUris.withAppendedId(uri, id);
111      }
```

```
112
113     @Override
114     public int delete(Uri uri, String selection, String[]
     selectionArgs) {
115         int deleteCount;
116
117         SQLiteDatabase database = mDbHelper.
    getWritableDatabase();
118         final int match = sUriMatcher.match(uri);
119         switch (match) {
120             case WORDS:
121                 deleteCount = database.delete(
    WordContract.WordEntry.TABLE_NAME, selection,
    selectionArgs);
122                 break;
123             case WORDS_ID:
124                 selection = WordContract.WordEntry._ID +
    "=?";
125                 selectionArgs = new String[]{String.
    valueOf(ContentUris.parseId(uri))};
126                 deleteCount = database.delete(
    WordContract.WordEntry.TABLE_NAME, selection,
    selectionArgs);
127                 break;
128             default:
129                 throw new IllegalArgumentException("
    Deletion is not supported for " + uri);
130         }
131         if (deleteCount > 0) {
132             getContext().getContentResolver().
    notifyChange(uri, null);
133         }
134         return deleteCount;
135     }
136
137     @Override
138     public int update(Uri uri, ContentValues
    contentValues,
139                       String selection, String[]
    selectionArgs) {
140         final int match = sUriMatcher.match(uri);
141         switch (match) {
142             case WORDS:
143                 getContext().getContentResolver().
    notifyChange(uri, null);
```

```
144                 return updateWord(uri,contentValues,
    selection, selectionArgs);
145             case WORDS_ID:
146                 selection = WordContract.WordEntry._ID +
    "=?";
147                 selectionArgs = new String[] {String.
    valueOf(ContentUris.parseId(uri))};
148                 getContext().getContentResolver().
    notifyChange(uri, null);
149                 return updateWord(uri, contentValues,
    selection, selectionArgs);
150             default:
151                 throw new IllegalArgumentException("
    Update is not supported for " + uri);
152         }
153
154     }
155
156     private int updateWord(Uri uri, ContentValues values,
157                            String selection, String[]
    selectionArgs) {
158         if (values.size() == 0)
159             return 0;
160
161         if (values.containsKey(WordContract.WordEntry.
    COLUMN_WORD)) {
162             String word = values.getAsString(WordContract
    .WordEntry.COLUMN_WORD);
163             if (TextUtils.isEmpty(word))
164                 throw new IllegalArgumentException("Word
    is needed.");
165         }
166
167         if (values.containsKey(WordContract.WordEntry.
    COLUMN_SENTENCE)) {
168             String sentence = values.getAsString(
    WordContract.WordEntry.COLUMN_SENTENCE);
169             if (TextUtils.isEmpty(sentence))
170                 throw new IllegalArgumentException("
    Sentence is needed.");
171         }
172
173         SQLiteDatabase database = mDbHelper.
    getWritableDatabase();
174         return database.update(WordContract.WordEntry.
```

```
174 TABLE_NAME, values, selection, selectionArgs);
175     }
176 }
177
```

```java
package tirzad.starunique.wordguess.data;

import android.text.TextUtils;

import java.security.SecureRandom;
import java.util.ArrayList;

/**
 * Created by StarUnique on 20/09/2017.
 */

public class SentenceHandling {

    private static int mArrayIndex;
    private static String[][] mArrayOfWords;
    private static ArrayList<String> mStringArray;

    public static String changeToStars(String sentence) {
        StringBuilder starredSentence = new StringBuilder(
);
        int sentenceLength = sentence.length();
        int characterCounter = 0;
        mArrayIndex = 0;
        char eachLetter = ' ';


        mArrayOfWords = new String[sentenceLength / 3][4];

        while (characterCounter < sentenceLength) {
            int charCounter = 0;
            String eachWord = "", starredWord = "";

            eachLetter = sentence.charAt(characterCounter)
;
            while (Character.isLowerCase(eachLetter) && ((
characterCounter + charCounter +1)<sentenceLength)) {
                eachWord = eachWord + eachLetter;
                charCounter++;
                eachLetter = sentence.charAt(
characterCounter + charCounter);
            }

            if (charCounter > 0) {
                mArrayIndex++;
                mArrayOfWords[mArrayIndex][3] = Integer.
```

```
41 toString(starredSentence.length());
42                 mArrayOfWords[mArrayIndex][2] = eachWord;
43                 mArrayOfWords[mArrayIndex][0] = Integer.
   toString(eachWord.length());
44                 mArrayOfWords[mArrayIndex][1] = Integer.
   toString(mArrayIndex);
45
46                 for (int counter = 1; counter <=
   charCounter - mArrayOfWords[mArrayIndex][0].length();
   counter++)
47                     starredWord = starredWord + "*";
48
49                 starredSentence.append(Integer.toString(
   eachWord.length()) +
50                         starredWord + eachLetter)
51                         ;
52             } else {
53                 starredSentence.append(eachLetter);
54             }
55             //      System.out.println(starredWord);
56             characterCounter += charCounter + 1;
57
58         }
59
60         return starredSentence.toString();
61     }
62
63     public static String giveHintWord(String
   starredSentence) {
64         StringBuilder buffer = new StringBuilder(
   starredSentence);
65         if (mArrayIndex > 0) {
66             SecureRandom randomNumbers = new SecureRandom(
   );
67             int randomWord = 1 + randomNumbers.nextInt(
   mArrayIndex);
68
69             while (mArrayOfWords[randomWord][1] == "")
70                 randomWord = 1 + randomNumbers.nextInt(
   mArrayIndex);
71
72             String chosenHint = mArrayOfWords[randomWord][
   2];
73             int hintWordPosition = Integer.parseInt(
   mArrayOfWords[randomWord][3]);
```

```
 74
 75             mArrayOfWords[randomWord][1] = mArrayOfWords[
    mArrayIndex][1];
 76             mArrayOfWords[randomWord][2] = mArrayOfWords[
    mArrayIndex][2];
 77             mArrayOfWords[randomWord][0] = mArrayOfWords[
    mArrayIndex][0];
 78             mArrayOfWords[randomWord][3] = mArrayOfWords[
    mArrayIndex][3];
 79
 80             mArrayIndex--;
 81
 82             buffer.delete(hintWordPosition,
    hintWordPosition + chosenHint.length());
 83             buffer.insert(hintWordPosition, chosenHint);
 84         }
 85         return buffer.toString();
 86     }
 87
 88     public static String checkGuessedWord(String
    starredSentence, String guessedWord) {
 89         StringBuilder builder = new StringBuilder(
    starredSentence);
 90         if (!TextUtils.isEmpty(guessedWord) &&
    mArrayIndex > 0) {
 91             for (int counter = 1; counter <= mArrayIndex;
     counter++) {
 92                 if (mArrayOfWords[counter][2].equals(
    guessedWord)) {
 93
 94                     int position = Integer.parseInt(
    mArrayOfWords[counter][3]);
 95                     builder.delete(position,
 96                             position + mArrayOfWords[
    counter][2].length());
 97                     builder.insert(position,
    mArrayOfWords[counter][2]);
 98
 99                     mArrayOfWords[counter][1] =
    mArrayOfWords[mArrayIndex][1];
100                     mArrayOfWords[counter][2] =
    mArrayOfWords[mArrayIndex][2];
101                     mArrayOfWords[counter][0] =
    mArrayOfWords[mArrayIndex][0];
102                     mArrayOfWords[counter][3] =
```

```
102 mArrayOfWords[mArrayIndex][3];
103
104                     mArrayOfWords[mArrayIndex][1] = "";
105                     mArrayOfWords[mArrayIndex][2] = "";
106                     mArrayOfWords[mArrayIndex][0] = "";
107                     mArrayOfWords[mArrayIndex][3] = "";
108
109                     mArrayIndex--;
110                 }
111             }
112         }
113         return builder.toString();
114
115     }
116
117
118 }
119
```