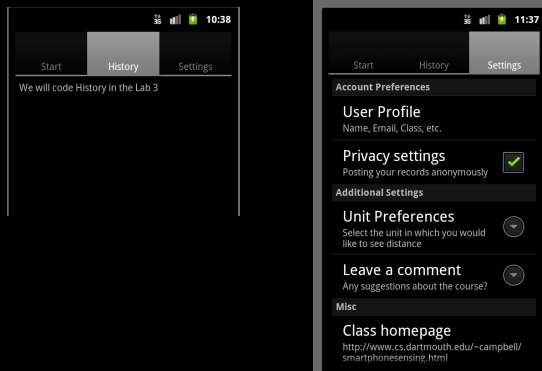


What you need to know for Lab 3

what we will cover

- More UI and adapters
- Data persistence: SQLite
- Displaying activities from the SQLite

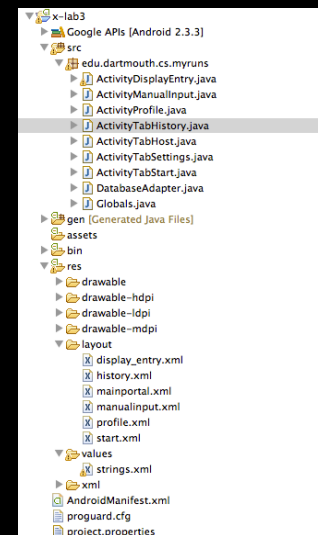
- After Lab 2 we have a settings all set but nothing in start and history tabs.
- Start allows the user to select an activity and history keeps a list of all activities undertaken in the past



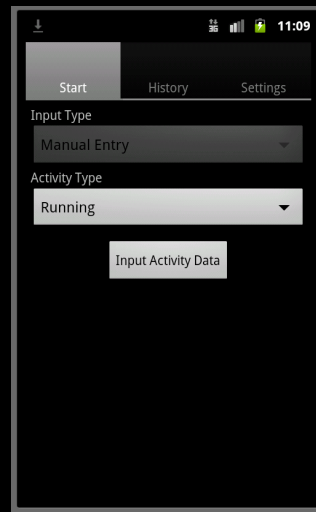
Quick look at our project for Lab 3.

There are a number of new activities and layout files: 1) ActivityManualInput and manualinput.xml; 2) ActivityDisplayEntry and display_entry.xml; 3) history.xml associates with ActivityTabHistory; and 4) DatabaseAdapter.

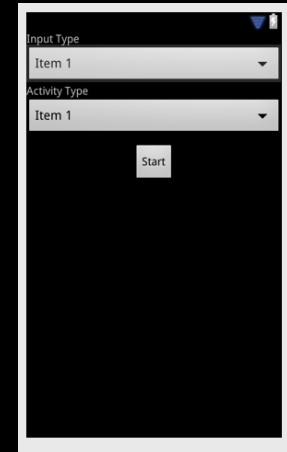
Also, we update some of the layout files we created last week: start.xml



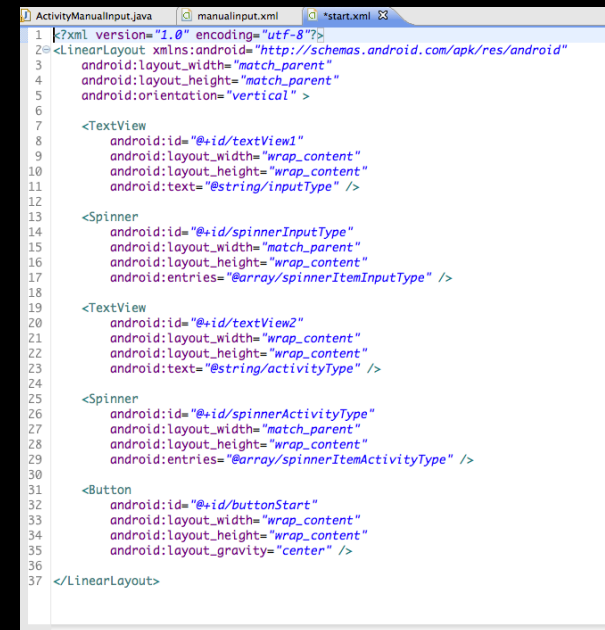
- The start activity allows automatic entry from GPS or, as in Lab3, just manual entry
- The user selects an activity and inputs activity data.
- Clicking InputActivityData fires an intent to ActivityManualInput



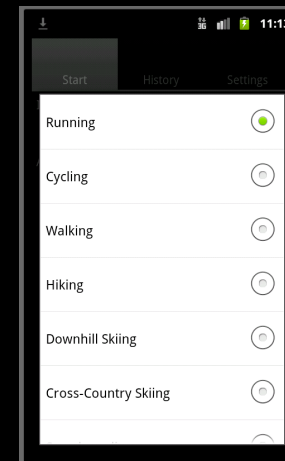
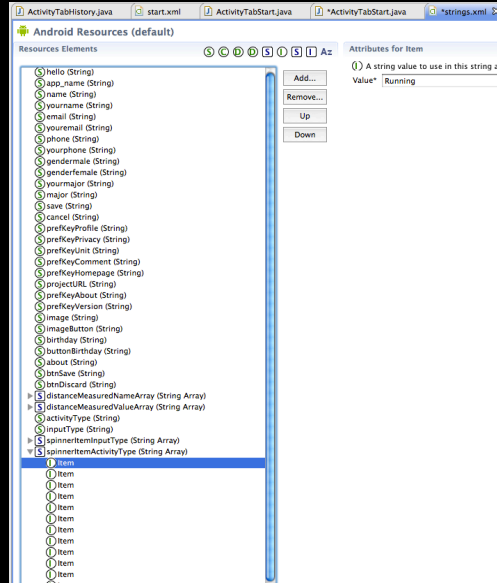
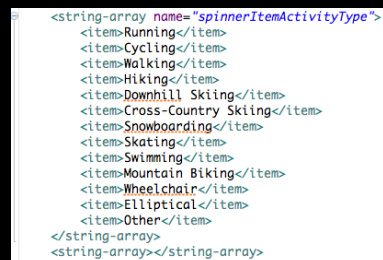
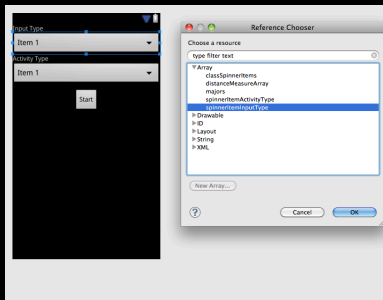
ActivityTabStart



start.xml



start.xml and strings.xml

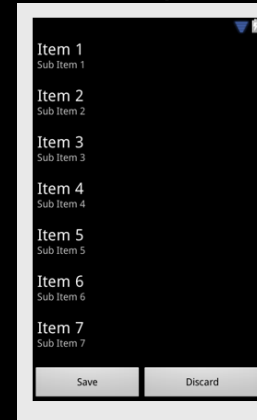


spinnerItemActivity in ActivityTabStart.java

ActivityTabStart fires intent for ActivityManualInput with extras for inputType and activityType

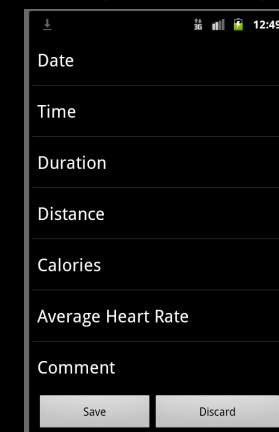
```
1 ActivityTabHistory.java 2 start.xml 3 ActivityTabStart.java 4 ActivityTabStart.java [2]
23
24 @Override
25 public void onCreate(Bundle savedInstanceState) {
26     super.onCreate(savedInstanceState);
27
28     mContext = this; // set the context to this activity
29
30     setContentView(R.layout.start);
31
32     Log.d(Globals.TAG, "entered onCreate ActivityTabStart");
33
34     // disable the GPS section of Input Type so its only Manual Entry
35
36     spinnerInputType = ((Spinner) findViewById(R.id.spinnerInputType));
37     spinnerActivityType = ((Spinner) findViewById(R.id.spinnerActivityType));
38
39     spinnerInputType.setSelection(0); // set to manual entry
40     spinnerActivityType.setEnabled(false); // disable entry
41
42     btn = ((Button) findViewById(R.id.buttonStart));
43     btn.setText("Input activity data");
44
45     View.OnClickListener myListener;
46
47     myListener = new View.OnClickListener() {
48         @Override
49         public void onClick(View v) {
50
51             // get current setting 1.0
52
53             int inputType = spinnerInputType.getSelectedItemPosition();
54
55             Bundle extras = new Bundle();
56
57             // get input selection (GPS, Manual)
58             extras.putInt("inputType", inputType);
59             // get input e.g., walking, cycling, etc.
60             extras.putInt("activityType",
61                 spinnerActivityType.getSelectedItemPosition());
62             Intent i;
63
64             if (inputType == 0) {
65                 // TODO GPS Intent
66             } else {
67                 // Manual input
68                 i = new Intent(mContext, ActivityManualInput.class);
69                 i.putExtras(extras);
70                 startActivity(i);
71             }
72         }
73     };
74 }
```

manualinput.xml



Setup UI for
ActivityManualInput in
manualinput.xml
ListView and buttons

ActivityManualInput



Setting up adapter
to display using code

```
1 manualinput.xml [2]
2 Lab3/res/layout/manualinput.xml [3]
3 <?xml version="1.0" encoding="utf-8"?>
4 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:orientation="vertical" >
8
9     <ListView
10         android:id="@android:id/list"
11         android:layout_width="match_parent"
12         android:layout_height="0dp"
13         android:layout_weight="1" >
14
15     </ListView>
16
17     <LinearLayout
18         android:id="@+id/linearLayout1"
19         android:layout_width="match_parent"
20         android:layout_height="wrap_content" >
21
22         <Button
23             android:id="@+id/btnSave"
24             android:layout_width="wrap_content"
25             android:layout_height="wrap_content"
26             android:layout_weight="1"
27             android:text="@string/btnSave" />
28
29         <Button
30             android:id="@+id/btnDiscard"
31             android:layout_width="wrap_content"
32             android:layout_height="wrap_content"
33             android:layout_weight="1"
34             android:text="@string/btnDiscard" />
35
36     </LinearLayout>
37 </LinearLayout>
```

change ListView id to
@android:id/list

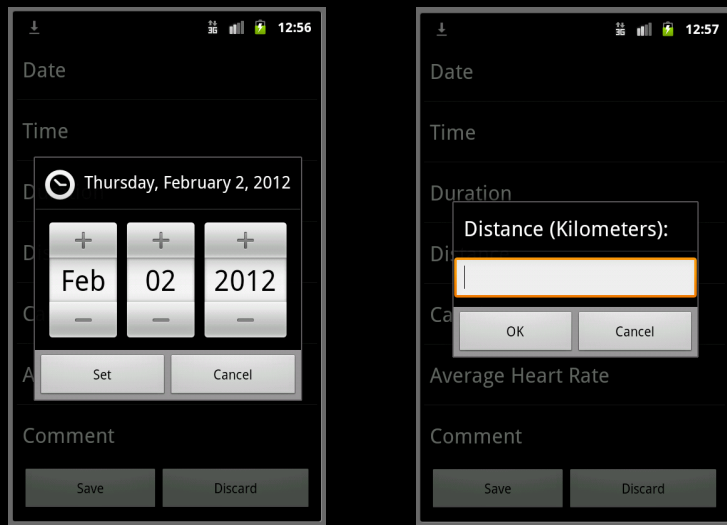
extend activity
ListActivity in
ActivityManualInput

Note, layout_height
=0 dp and
layout_weight=1

manualinput.xml

```
1 manualinput.xml [2] ActivityManualInput.java [3]
2 package edu.dartmouth.cs.myruns;
3
4 import java.util.Calendar;
5
6 public class ActivityManualInput extends ListActivity {
7
8     private static final int DIALOG_DATE_ID = 0;
9     private static final int DIALOG_TIME_ID = 1;
10    private static final int DIALOG_DURATION_ID = 2;
11    private static final int DIALOG_DISTANCE_ID = 3;
12    private static final int DIALOG_CALORIES_ID = 4;
13    private static final int DIALOG_HEARTRATE_ID = 5;
14    private static final int DIALOG_COMMENT_ID = 6;
15
16    private final String[] items = { "Date", "Time", "Duration", "Distance",
17        "Calories", "Heart Rate", "Comments" };
18
19    // Date member for exercise time.
20
21    private int mYear;
22    private int mMonth;
23    private int mDay;
24
25    private double mDistance;
26
27    @Override
28    protected void onCreate(Bundle savedInstanceState) {
29        // TODO Auto-generated method stub
30        super.onCreate(savedInstanceState);
31
32        setContentView(R.layout.manualinput);
33
34        setListAdapter(new ArrayAdapter<String>(this,
35            android.R.layout.simple_list_item_1, items));
36
37        // get today's date and update date members.
38
39        final Calendar c = Calendar.getInstance();
40        mYear = c.get(Calendar.YEAR);
41        mMonth = c.get(Calendar.MONTH);
42        mDay = c.get(Calendar.DAY_OF_MONTH);
43
44        Log.d("MyRuns", "Today's date " + mYear + " " + mMonth + " " + mDay);
45    }
46 }
```

Snippet ActivityManualInput setting up adapter for
simple_list_item_1



Example dialogs for Date and Distance

```
@Override
protected void onItemClick(ListView l, View v, int positionIsDialogId,
    long id) {
    // TODO Auto-generated method stub
    super.onItemClick(l, v, positionIsDialogId, id);
    showDialog(positionIsDialogId);
}

@Override
protected Dialog onCreateDialog(int positionIsDialogId) {
    // TODO Auto-generated method stub
    // return super.onCreateDialog(id);

    final EditText textEntryView;

    // get stored preference (miles or km) or default
    SharedPreferences settings = PreferenceManager
        .getDefaultSharedPreferences(getBaseContext());

    String distanceMeasure = settings.getString(
        getString(R.string.prefKeyUnit),
        getResources()
            .getStringArray(R.array.distanceMeasuredNameArray)[0]);

    switch (positionIsDialogId) {
```

Set up for creating a dialog (e.g., date, duration) using showDialog in onItemClick()

```
117
118
119 case DIALOG_DISTANCE_ID:
120
121     textEntryView = new EditText(this);
122     textEntryView.setInputType(InputType.TYPE_CLASS_NUMBER
123         | InputType.TYPE_NUMBER_FLAG_DECIMAL);
124
125     // onClick for clicking OK
126     DialogInterface.OnClickListener OKListener = new DialogInterface.OnClickListener() {
127
128         @Override
129         public void onClick(DialogInterface dialog, int which) {
130             SharedPreferences settings = PreferenceManager
131                 .getDefaultSharedPreferences(getBaseContext());
132
133             String distanceMeasure = settings.getString(
134                 getString(R.string.prefKeyUnit),
135                 getResources().getStringArray(
136                     R.array.distanceMeasuredNameArray)[0]);
137
138             mDistance = Double.parseDouble(textEntryView.getText()
139                 .toString());
140
141             // we always store in meters
142             mDistance *= Globals.KM2M;
143             if (distanceMeasure.compareTo("Miles") == 0)
144                 mDistance *= Globals.KM2MILE_PAT10;
145
146         }
147     };
148
149     DialogInterface.OnClickListener CancelListener = new DialogInterface.OnClickListener() {
150
151         @Override
152         public void onClick(DialogInterface dialog, int which) {
153             textEntryView.setText("");
154         }
155     };
156
157     // builds the dialog box for distance
158     AlertDialog.Builder builder = new AlertDialog.Builder(this);
159     builder.setTitle("Distance");
160     builder.setView(textEntryView);
161     builder.setPositiveButton("OK", OKListener);
162     builder.setNegativeButton("Cancel", CancelListener);
163
164     // launch dialog
165     return builder.create();
166
167
```

Construct the dialog for distance and render

data persistence

SharedPreferences for key/value pairs

Files (internal/external -- SD card)

Database (SQLite)

DataBaseAdapter.java

```
23 public class DatabaseAdapter {
24     private static final String DATABASE_NAME = "MyRunsDB";
25     private static final String TABLE_NAME_ENTRIES = "entries";
26
27     private static final int DATABASE_VERSION = 1;
28
29     public static final String KEY_ROWID = Globals.KEY_ROWID; // ".id";
30     public static final String KEY_INPUT_TYPE = Globals.KEY_INPUT_TYPE; // "input_type";
31     public static final String KEY_ACTIVITY_TYPE = Globals.KEY_ACTIVITY_TYPE; // "activity_type";
32     public static final String KEY_DATE_TIME = Globals.KEY_DATE_TIME; // "date_time";
33     public static final String KEY_DURATION = Globals.KEY_DURATION; // "duration";
34     public static final String KEY_DISTANCE = Globals.KEY_DISTANCE; // "distance";
35     public static final String KEY_AVG_PACE = Globals.KEY_AVG_PACE; // "avg_pace";
36     public static final String KEY_AVG_SPEED = Globals.KEY_AVG_SPEED; // "KEY_AVG_SPEED";
37     public static final String KEY_CALORIES = Globals.KEY_CALORIES; // "calories";
38     public static final String KEY_CLIMB = Globals.KEY_CLIMB; // "climb";
39     public static final String KEY_AVG_HEARTRATE = Globals.KEY_AVG_HEARTRATE; // "avg_hearttrate";
40     public static final String KEY_COMMENT = Globals.KEY_COMMENT; // "comment";
41     public static final String KEY_PRIVACY = Globals.KEY_PRIVACY; // "avg_privacy";
42     public static final String KEY_GPS_DATA = Globals.KEY_GPS_DATA; // "gps_data";
43
44     private static final String CREATE_TABLE_ENTRIES = "CREATE TABLE IF NOT EXISTS "
45     + "entries (_id INTEGER PRIMARY KEY AUTOINCREMENT, "
46     + "input_type INTEGER NOT NULL, "
47     + "activity_type INTEGER NOT NULL, "
48     + "date_time DATETIME NOT NULL, "
49     + "duration INTEGER NOT NULL, "
50     + "distance FLOAT, "
51     + "avg_pace INTEGER, "
52     + "avg_speed INTEGER, "
53     + "calories INTEGER, "
54     + "climb INTEGER, "
55     + "avg_hearttrate INTEGER, "
56     + "comment TEXT, "
57     + "privacy INTEGER, " + "gps_data BLOB " + ");";
58
59     // Variable to hold the database instance
60     private SQLiteDatabase db;
61     // Context of the application using the database.
62     // private final Context context;
63     // Database open/upgrade helper
64     private DBHelper dbHelper;
65
66     public DatabaseAdapter(Context context) {
67         dbHelper = new DBHelper(context, DATABASE_NAME, null, DATABASE_VERSION);
68     }
69
70 }
```

set up all pair/values (e.g., key_input_type) and
create_table_entries that forms

```
        mInput, mActivityType,
        DatabaseAdapter db = new DatabaseAdapter(mContext);
        db.open();
        long id = db.insertEntry(mInputType, mActivityType,
        mDateTime, mDuration, mDistance, mCalories,
        mComment);
        db.close();
        Toast.makeText(getApplicationContext(),
        "Entry #" + id + " saved.", Toast.LENGTH_SHORT)
        .show();
        finish();
    }
```

Operations on the DB from ActivityManualInput

```
132 private static class DBHelper extends SQLiteOpenHelper {
133
134     public DBHelper(Context context, String name, CursorFactory factory,
135                     int version) {
136         super(context, name, factory, version);
137     }
138
139     // Called when no database exists in disk and the helper class needs
140     // to create a new one.
141     @Override
142     public void onCreate(SQLiteDatabase db) {
143
144         try {
145             db.execSQL(CREATE_TABLE_ENTRIES);
146         } catch (SQLException e) {
147             e.printStackTrace();
148         }
149     }
150
151 }
```

helper class creates a new database when constructed
using the db.execSQL(create_table_entries)

```

10 public DatabaseAdapter(Context context) {
11     dbHelper = new DBHelper(context, DATABASE_NAME, null, DATABASE_VERSION);
12 }
13
14
15 public DatabaseAdapter open() throws SQLException {
16     db = dbHelper.getWritableDatabase();
17     return this;
18 }
19
20 public void close() {
21     db.close();
22 }
23
24 public long insertEntry(int inputType, int activityType, GregorianCalendar time, long durationInSeconds,
25     int calories, String comment){
26
27     ContentValues value = new ContentValues();
28
29     value.put(KEY_INPUT_TYPE, inputType);
30     value.put(KEY_ACTIVITY_TYPE, activityType);
31     value.put(KEY_DATE_TIME, time.getTimeInMillis()/1000);
32     value.put(KEY_DURATION, durationInSeconds);
33     value.put(KEY_DISTANCE, distanceInMeters);
34     value.put(KEY_CALORIES, calories);
35     value.put(KEY_COMMENT, comment);
36
37     return db.insert(TABLE_NAME_ENTRIES, null, value);
38 }
39
40

```

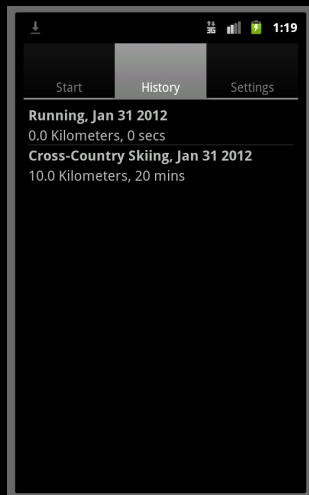
Setting up the database, open() and insertEntry()

```

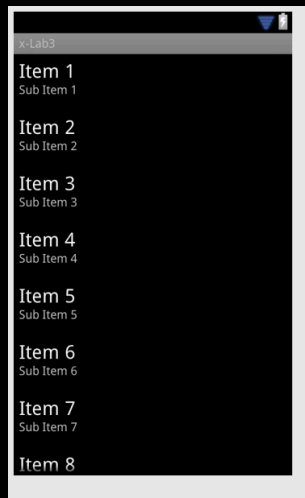
117 public Cursor fetchEntries() {
118     return db.query(TABLE_NAME_ENTRIES, new String[] { KEY_ROWID,
119         KEY_ACTIVITY_TYPE, KEY_DATE_TIME, KEY_DURATION, KEY_DISTANCE },
120         null, null, null, null, null);
121 }
122

```

fetchEntries()



ActivityTabHistory is
extended from
ListActivity



history.xml ListView,
use @android:id/list

```

27
28 public class ActivityTabHistory extends ListActivity {
29
30     private DatabaseAdapter mDb;
31     private Context mContext;
32     private ActivityEntriesCursorAdapter mAdapter;
33     private Cursor mActivityEntryCursor;
34
35     private int mRowIndex;
36     private int mActivityIndex;
37     private int mTimeIndex;
38     private int mDurationIndex;
39     private int mDistanceIndex;
40
41     @Override
42     public void onCreate(Bundle savedInstanceState) {
43
44         super.onCreate(savedInstanceState);
45
46         setContentView(R.layout.history);
47         mContext = this;
48
49         mDb = new DatabaseAdapter(this);
50         mDb.open();
51         mActivityEntryCursor = mDb.fetchEntries();
52
53         mRowIndex = mActivityEntryCursor.getColumnIndex(DatabaseAdapter.KEY_ROWID);
54         mActivityIndex = mActivityEntryCursor.getColumnIndex(DatabaseAdapter.KEY_ACTIVITY_TYPE);
55         mTimeIndex = mActivityEntryCursor.getColumnIndex(DatabaseAdapter.KEY_DATE_TIME);
56         mDurationIndex = mActivityEntryCursor.getColumnIndex(DatabaseAdapter.KEY_DURATION);
57         mDistanceIndex = mActivityEntryCursor.getColumnIndex(DatabaseAdapter.KEY_DISTANCE);
58
59         startManagingCursor(mActivityEntryCursor);
60         mAdapter = new ActivityEntriesCursorAdapter(
61             this, mActivityEntryCursor);
62         setListAdapter(mAdapter);
63
64     }
65

```

History uses a cursor and adapter to manage data

```

82
83 @Override
84 protected void onItemClick(ListView l, View v, int position, long id) {
85
86     String activityTypes[] = getResources().getStringArray(R.array.spinnerItemActivityType);
87
88     mActivityEntryCursor.moveToPosition(position);
89
90     int i = mActivityEntryCursor.getInt(mActivityEntryCursor
91         .getColumnIndex(Globals.KEY_ACTIVITY_TYPE));
92
93     Bundle extras = new Bundle();
94
95     extras.putLong(Globals.KEY_ROWID, mActivityEntryCursor.getLong(mRowIndex));
96     extras.putString(Globals.KEY_ACTIVITY_TYPE, parseActivityType(mActivityEntryCursor.getInt(mActivityIndex)));
97     extras.putString(Globals.KEY_DATE_TIME, parseTime(mActivityEntryCursor.getLong(mTimeIndex), false));
98     extras.putString(Globals.KEY_DURATION, parseDistance(mActivityEntryCursor.getDouble(mDistanceIndex)));
99     extras.putString(Globals.KEY_DISTANCE, parseDuration(mActivityEntryCursor.getInt(mDurationIndex)));
100
101     Intent intent = new Intent(mContext, ActivityDisplayEntry.class);
102     intent.putExtras(extras);
103
104     startActivity(intent);
105
106

```

The intent used to start ActivityDisplayEntry uses extras to pass the data fetched from the cursor/ database

Displaying activities: in our design ActivityTabHistory reads the DB and passes (via extras and intent) the data to ActivityDisplayEntry for rendering

Activity type
Running

Date and time
13:24:00 Feb 2 2012

Duration
12.0 Kilometers

Distance
30 mins

Delete

ActivityDisplayEntry

Activity type

Date and time

Duration

Distance

display_entries.xml

Let's look at the code