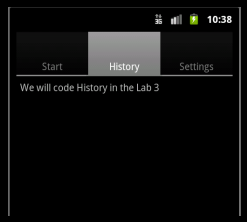
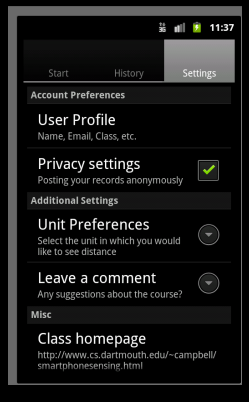
What you need to know for Lab 3

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

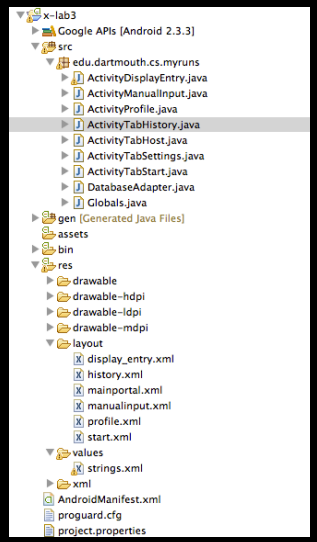




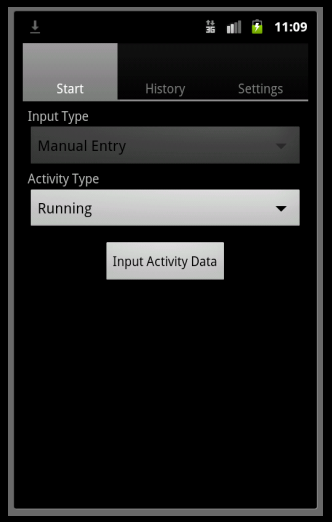
Quick look at our project for Lab 3.

There are a number of new activities and layout files: 1) ActivityManualInput and manualinput.xml; 2) ActivityDisplayEntry and display_entry; 4) history.xml associates with ActivityTabHistory; and3) DatabaseAdapter.

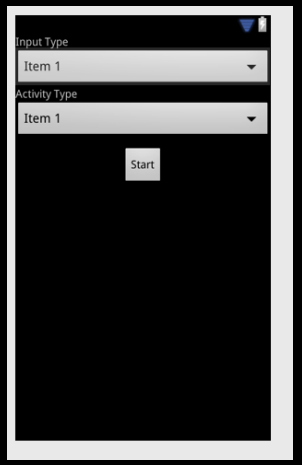
Also, we update some of the layout files we created last week: start.xml



- 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



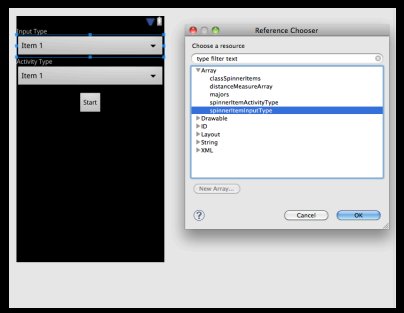
ActivityTabStart



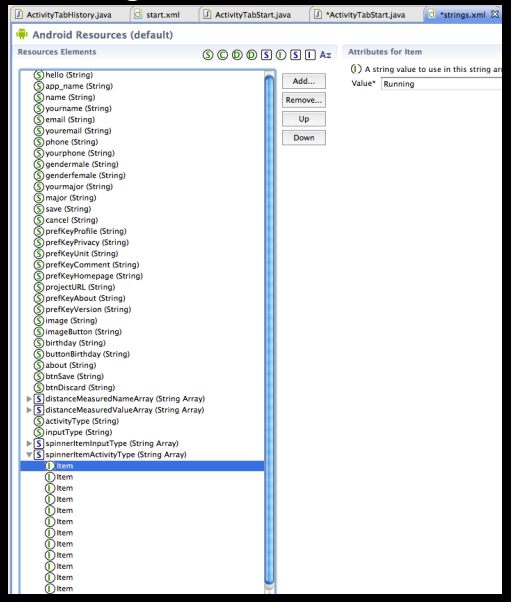
start.xml

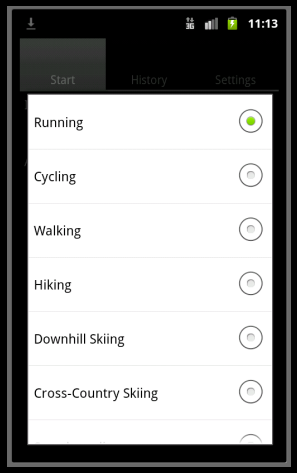
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/inputType" />

    <Spinner
        android:id="@+id/spinnerInputType"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:entries="@array/spinnerItemInputType" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/activityType" />

    <Spinner
        android:id="@+id/spinnerActivityType"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:entries="@array/spinnerItemActivityType" />

    <Button
        android:id="@+id/buttonStart"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center" />

</LinearLayout>
```

start.xml and strings.xml



```
<string-array name="spinnerItemActivityType">
    <item>Running</item>
    <item>Cycling</item>
    <item>Walking</item>
    <item>Hiking</item>
    <item>Downhill Skiing</item>
    <item>Cross-Country Skiing</item>
    <item>Snowboarding</item>
    <item>Skating</item>
    <item>Swimming</item>
    <item>Mountain Biking</item>
    <item>Wheelchair</item>
    <item>Elliptical</item>
    <item>Other</item>
</string-array>
<string-array></string-array>
```



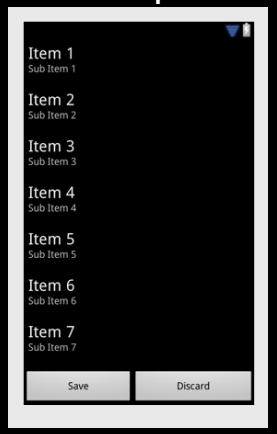


spinnerItemActivity in ActivityTabStart.java

ActivityTabStart fires intent for ActivityManualInput with extras for inputType and activityType

```
ActivityTabHistory.java | start.xml | ActivityTabStart.java | *ActivityTabStart.java

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

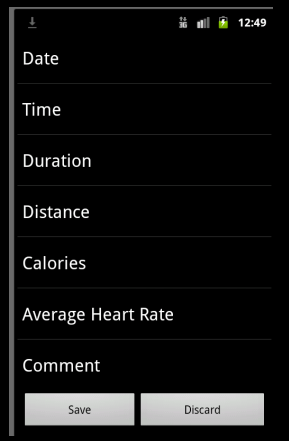
manualinput.xml



Setup UI for ActivityManualInput in manualinput.xml

ListView and buttons

ActivityManualInput



Setting up adapter to display using code

```
manualinput.xml
Lab3/res/layout/manualinput.xml

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

change ListView id to @android:id/list

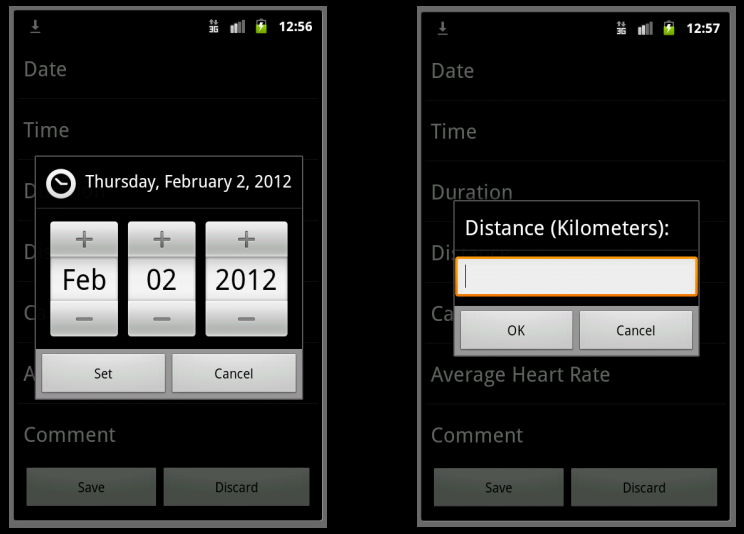
extend activity ListActivity in ActivityManualInput

Note, layout_height =0 dp and layout_weight=1

manualinput.xml

```
manualinput.xml | ActivityManualInput.java

1  package edu.dartmouth.cs.myruns;
2
3  import java.util.Calendar;
20
21 public class ActivityManualInput extends ListActivity {
22
23     private static final int DIALOG_DATE_ID = 0;
24     private static final int DIALOG_TIME_ID = 1;
25     private static final int DIALOG_DURATION_ID = 2;
26     private static final int DIALOG_DISTANCE_ID = 3;
27     private static final int DIALOG_CALORIES_ID = 4;
28     private static final int DIALOG_HEARTRATE_ID = 5;
29     private static final int DIALOG_COMMENT_ID = 6;
30
31     private final String[] items = { "Date", "Time", "Duration", "Distance",
32             "Calories", "Heart Rate", "Comments" };
33
34     // Date member for exercise time.
35
36     private int mYear;
37     private int mMonth;
38     private int mDay;
39
40     private double mDistance;
41
42     @Override
43     protected void onCreate(Bundle savedInstanceState) {
44         // TODO Auto-generated method stub
45         super.onCreate(savedInstanceState);
46
47         setContentView(R.layout.manualinput);
48
49         setListAdapter(new ArrayAdapter<String>(this,
50                 android.R.layout.simple_list_item_1, items));
51
52         // get todays date and update date members.
53
54         final Calendar c = Calendar.getInstance();
55         mYear = c.get(Calendar.YEAR);
56         mMonth = c.get(Calendar.MONTH);
57         mDay = c.get(Calendar.DAY_OF_MONTH);
58
59         Log.d("MyRuns", "Today's date " + mYear + " " + mMonth + " " + mDay);
60     }
```

Snippet ActivityManualInput setting up adapter for simple_list_item_1



Example dialogs for Date and Distance

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

manualinput.xml *ActivityManualInput.java

```
case DIALOG_DISTANCE_ID:

    textEntryView = new EditText(this);
    textEntryView.setInputType(InputType.TYPE_CLASS_NUMBER
            | InputType.TYPE_NUMBER_FLAG_DECIMAL);

    // OnClick for clicking OK
    DialogInterface.OnClickListener OKListener = new DialogInterface.OnClickListener() {

        @Override
        public void onClick(DialogInterface dialog, int which) {
            SharedPreferences settings = PreferenceManager
                    .getDefaultSharedPreferences(getBaseContext());

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

            mDistance = Double.parseDouble(textEntryView.getText()
                    .toString());

            // we always store in meters
            mDistance *= Globals.KILO;
            if (distanceMeasure.compareTo("Miles") == 0)
                mDistance *= Globals.KM2MILE_RATIO;
        }

    };

    DialogInterface.OnClickListener CancelListener = new DialogInterface.OnClickListener() {

        @Override
        public void onClick(DialogInterface dialog, int which) {
            textEntryView.setText("");
        }
    };

    // builds the dialog box for distance

    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Distance");
    builder.setView(textEntryView);
    builder.setPositiveButton("OK", OKListener);
    builder.setNegativeButton("Cancel",CancelListener);

    // launch dialog

    return builder.create();
```

Construct the dialog for distance and render

data persistence

SharedPreference for key/value pairs

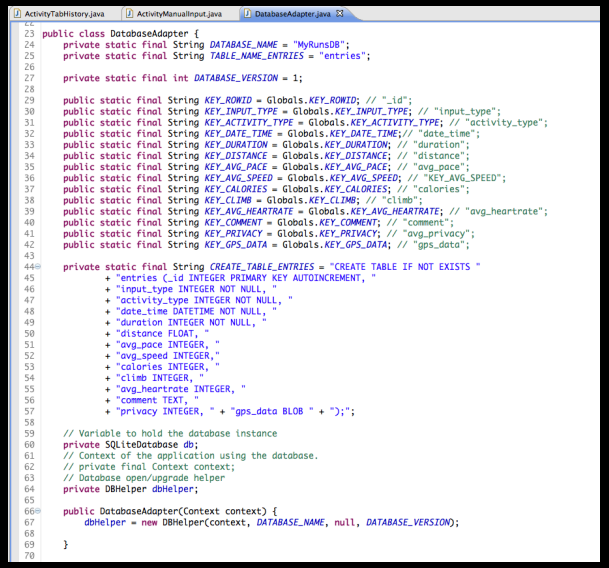
Files (internal/external -- SD card

Database (SQLite)

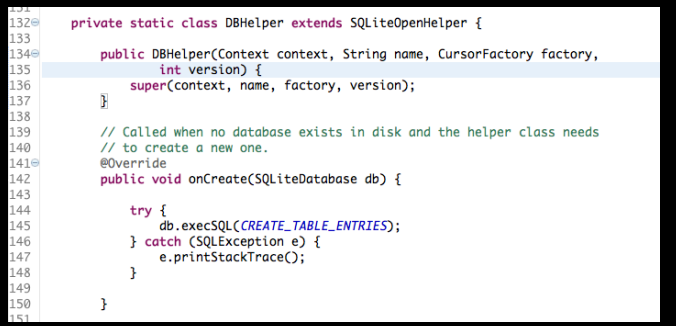
DataBaseAdapter.java

```
                mHour, mMinute);
        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



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



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

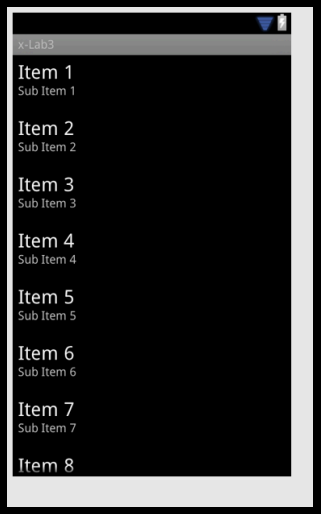
```
    public DatabaseAdapter(Context context) {
        dbHelper = new DBHelper(context, DATABASE_NAME, null, DATABASE_VERSION);

    }

    public DatabaseAdapter open() throws SQLException {
        db = dbHelper.getWritableDatabase();
        return this;
    }

    public void close() {
        db.close();
    }

    public long insertEntry(int inputType, int activityType, GregorianCalendar time, long durationInSeconds
            int calories, String comment){

        ContentValues value = new ContentValues();

        value.put(KEY_INPUT_TYPE, inputType);
        value.put(KEY_ACTIVITY_TYPE, activityType);
        value.put(KEY_DATE_TIME, time.getTimeInMillis()/1000);
        value.put(KEY_DURATION, durationInSeconds);
        value.put(KEY_DISTANCE, distanceInMeters);
        value.put(KEY_CALORIES, calories);
        value.put(KEY_COMMENT, comment);

        return db.insert(TABLE_NAME_ENTRIES, null, value);
    }
```

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

```
    public Cursor fetchEntries() {
        return db.query(TABLE_NAME_ENTRIES, new String[] { KEY_ROWID,
                KEY_ACTIVITY_TYPE, KEY_DATE_TIME, KEY_DURATION, KEY_DISTANCE },
                null, null, null, null, null);
    }
```

fetchEntries()

ActivityTabHistory is extended from ListActivity



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

ActivityTabHistory.java | ActivityManualInput.java | DatabaseAdapter.java | history.xml | display_entry.xml

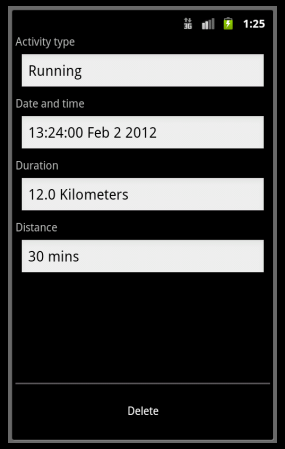
```
public class ActivityTabHistory extends ListActivity {

    private DatabaseAdapter mDb;
    private Context mContext;
    private ActivityEntriesCursorAdapter mAdapter;
    private Cursor mActivityEntryCursor;

    private int mRowIdIndex;
    private int mActivityIndex;
    private int mTimeIndex;
    private int mDurationIndex;
    private int mDistanceIndex;

    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.history);
        mContext = this;

        mDb = new DatabaseAdapter(this);
        mDb.open();
        mActivityEntryCursor = mDb.fetchEntries();

        mRowIdIndex = mActivityEntryCursor.getColumnIndex(DatabaseAdapter.KEY_ROWID);
        mActivityIndex = mActivityEntryCursor.getColumnIndex(DatabaseAdapter.KEY_ACTIVITY_TYPE);
        mTimeIndex = mActivityEntryCursor.getColumnIndex(DatabaseAdapter.KEY_DATE_TIME);
        mDurationIndex = mActivityEntryCursor.getColumnIndex(DatabaseAdapter.KEY_DURATION);
        mDistanceIndex = mActivityEntryCursor.getColumnIndex(DatabaseAdapter.KEY_DISTANCE);

        startManagingCursor(mActivityEntryCursor);
        mAdapter = new ActivityEntriesCursorAdapter(
                this, mActivityEntryCursor);
        setListAdapter(mAdapter);
    }
```

History uses a cursor and adapter to manage data

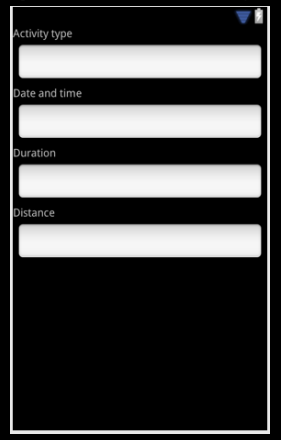
```
@Override
protected void onListItemClick(ListView l, View v, int position, long id) {

    String activityTypes[] = getResources().getStringArray(R.array.spinnerItemActivityType);

    mActivityEntryCursor.moveToPosition(position);

    int i = mActivityEntryCursor.getInt(mActivityEntryCursor
            .getColumnIndex(Globals.KEY_ACTIVITY_TYPE));

    Bundle extras = new Bundle();

    extras.putLong(Globals.KEY_ROWID, mActivityEntryCursor.getLong(mRowIdIndex));
    extras.putString(Globals.KEY_ACTIVITY_TYPE, parseActivityType(mActivityEntryCursor.getInt(mActivityIndex)));
    extras.putString(Globals.KEY_DATE_TIME, parseTime(mActivityEntryCursor.getLong(mTimeIndex), false));
    extras.putString(Globals.KEY_DURATION, parseDistance(mActivityEntryCursor.getDouble(mDistanceIndex)));
    extras.putString(Globals.KEY_DISTANCE, parseDuration(mActivityEntryCursor.getInt(mDurationIndex)));

    Intent intent = new Intent(mContext, ActivityDisplayEntry.class);
    intent.putExtras(extras);

    startActivity(intent);

```

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



ActivityDisplayEntry



display_entries.xml

Let's look at the code