CS 193A

Stanford Android Library

Motivation

- Android development is harder than it needs to be.
  - Many common tasks that should be simple aren't.

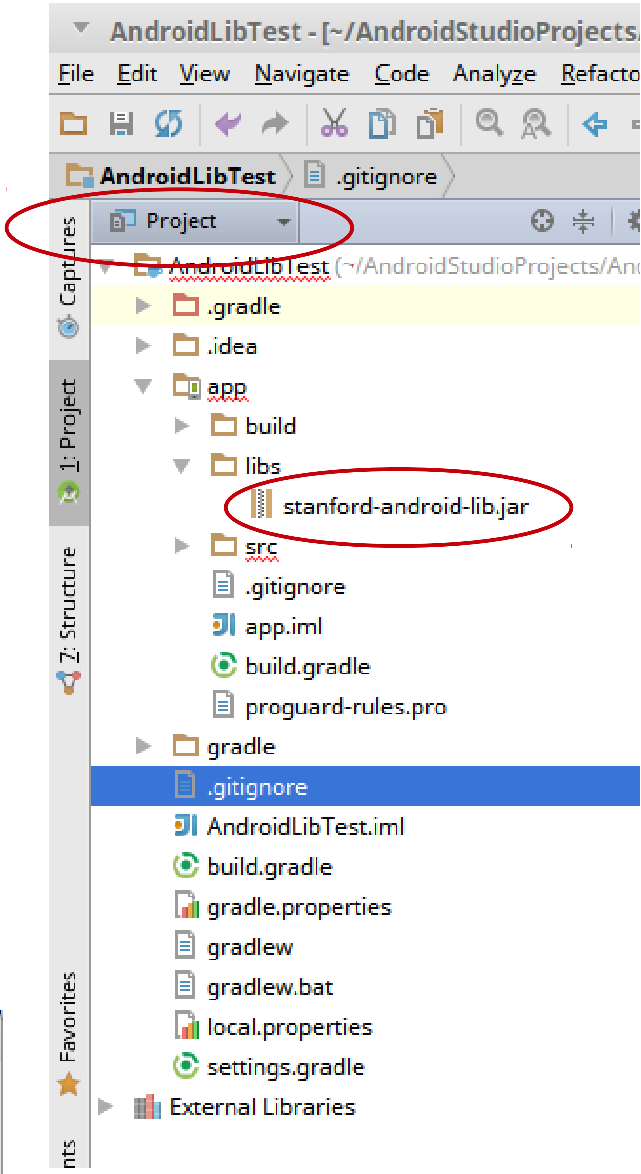
- Stanford (Marty) is creating a library to make it simpler:

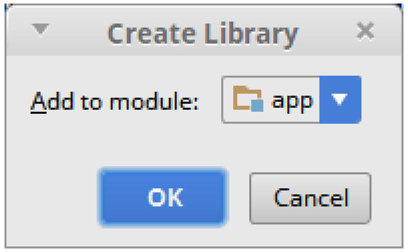
```
~~public class MyActivity extends Activity {~~
public class MyActivity extends SimpleActivity {
```

  - The `SimpleActivity` class provides lots of convenience methods and functionality for simplifying common Android tasks.
  - We will continue to develop the library during the course.
  - We will automatically link the library to future homeworks.

Using the library

- Download library JAR from class web site:
  - http://cs193a.stanford.edu/lib/
- Attach the .JAR file to your project:
  - Put the JAR in your project's app/libs/ folder.
  - In Android Studio:
    - make sure you are in "Project" view mode.
    - scroll down to app/libs/ folder.
    - right-click the JAR.
    - choose "Add as Library" near the bottom.
    - add the lib to your module named "app".





Another way to add library

- Download library JAR from class web site:
  - http://cs193a.stanford.edu/lib/
- Attach the .JAR file to your project:
  - Put the JAR in your project's app/libs/ folder.
  - In Android Studio:
    - Open the **build.gradle** file for your app.
    - Find the section called 'dependencies'.
    - Add the following line inside that section.

```
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    ...
    compile files('libs/stanford-android-lib.jar')
}
```

Accessing widgets by IDs

`findButton(`***`id`***`)`	returns Button for given ID
`findCalendarView, findCheckBox, findDatePicker, findEditText, findFragment, findGridView, findImageButton, findImageView, findListView, findProgressBar, findRadioButton, findRadioGroup, findRatingBar, findScrollView, findSearchView, findSeekBar, findSpace, findSpinner, findStackView, findSwitch, findTextView, findTimePicker, findToggleButton, findToolbar, findZoomButton`	returns widget of given type that has the given ID

`find(`***`id`***`)`
`$(`***`id`***`)`
	alias for `findViewById` but using generics to avoid casts

`$B(`***`id`***`), $CB(`***`id`***`), $ET(`***`id`***`), $IB(`***`id`***`), $IV(`***`id`***`), $LV(`***`id`***`), $RB(`***`id`***`), $TV(`***`id`***`), ...`	alias for `find` but casts to Button, CheckBox, TextView, ...

```
// access widgets by ID without needing to cast
Button button = $B(R.id.mybutton);
ListView list = $LV(R.id.mylist);
TextView text = $(R.id.mytext);
$TV(R.id.mytext).setText("hello!");
...
```

Logging, printing, toasts

Method	Description

`log("`***`message`***`");`
`log(`***`exception`***`);`
`log("`***`message`***`", `***`exception`***`);`
	equivalent to `Log.d`

`println("`***`message`***`");`
`printf("`***`formatStr`***`", `***`args`***`);`
	equivalent to `Log.v`

`toast("`***`message`***`");`
`toast("`***`message`***`", `***`time`***`);`
	equivalent to `Toast.makeText`

```
// slightly easier printing of debug/toast messages
// (these methods are in SimpleActivity)
println("A message from SimpleActivity");
toast("A toast message");
```

The "with" pattern

```
// Many Android libraries use a pattern of
// ClassName.with(this)
//           .methodName();
//
// where 'this' is your Activity

ListView list = $(R.id.mylist);
SimpleList.with(this)
    .setItems(list, "Leo", "Mike", "Don", "Raph");
```

SimpleList

Method	Description
`createAdapter(`***`items`***`)`	create/return an ArrayAdapter

`createAdapter(`***`item1`***`,`
`    `***`item2`***`, ..., `***`itemN`***`)`
	create/return an ArrayAdapter

`getItems(`***`id`***`)`
`getItems(`***`listView`***`)`
	return items as ArrayList

`setItems(`***`id`***`, `***`items`***`);`
`setItems(`***`listView`***`, `***`items`***`);`
	set items from ArrayList

`setItems(`***`listView`***`, `***`item1`***`,`
`    `***`item2`***`, ..., `***`itemN`***`);`
	set items in list view

```
// easy get/set of ListView items
SimpleList.with(this)
    .setItems(R.id.mylist, "Leo", "Mike", "Don", "Raph");
```

Standard list events

```
// normal crappy code to hear list item click events
ListView list = findListView(R.id.mylist);
list.setOnItemClickListener(
    new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent,
                View view, int index, long id) {
            // phew! event handler code goes here :-(
        }
    }
);
```

Easier list events

```
// SimpleActivity code to hear list item click events
ListView list = findListView(R.id.mylist);
list.setOnItemClickListener(this);
...

public void onItemClick(ListView list, int index) {
    // event handler code goes here :-)
}

// also available:
//      - onItemLongClick
//      - onItemSelected
//      - other similar events for other widget types
```

SimpleIO

Method	Description

`openExternalFileBufferedReader("`***`filename`***`")`
`openExternalFileScanner("`***`filename`***`")`
	read file in external storage

`openExternalFilePrintStream(`***`filename`***`)`	write file in external storage

`openInternalFileBufferedReader(`***`id`***`)`
`openInternalFileScanner(`***`id`***`)`
	read file in internal storage

`readFileLines(`***`id`***`)        // internal`
`readFileLines(`***`filename`***`)  // external`
	read file and return its lines as an ArrayList of strings

`readFileText(`***`id`***`)         // internal`
`readFileText(`***`filename`***`)   // external`
	read file and return its text as a String

`writeFileLines(`***`filename, list`***`); // external`
`writeFileText(`***`filename, text`***`);`
	write contents of a list or string to an external file

```
// more easily read and write files
Scanner scan = SimpleIO.with(this)
                .openInternalFileScanner(R.raw.myfile);
while (scan.hasNextLine()) { ... }
```

System directories

Method	Description
`getDocumentsDirectory()`	dir where docs are stored
`getDownloadsDirectory()`	dir where downloads are stored
`getMoviesDirectory()`	dir where movies are stored
`getMusicDirectory()`	dir where music/songs are stored
`getPhotosDirectory()`	dir where pictures are stored

```
// write to a file in the documents directory
File dir = SimpleIO.with(this).getDocumentsDirectory();
PrintStream out = SimpleIO.with(this)
    .openExternalPrintStream(dir, "myfile.txt");
out.println("this is a test");
out.close();
```

SimpleMedia

Method	Description
`play(`***`id`***`);`	play/unpause sound with given ID
`loop(`***`id`***`);`	repeatedly plays sound
`pause(`***`id`***`);`	pause sound if playing
`stop(`***`id`***`);`	stops the given sound if playing
`isPlaying(`***`id`***`)`	returns true if the sound is playing
`isLooping(`***`id`***`)`	returns true if the sound is looping
`getPosition(`***`id`***`)`	returns time index of playing clip in MS
`setPosition(`***`id, ms`***`)`	advances the clip to the given time

```
// convenience methods for playing sounds
SimpleMedia.with(this).play(R.id.cowabunga);
SimpleMedia.with(this).loop(R.id.tmnt_theme);
```

SimpleSpeech

Method	Description
`speak("`***text***`");`	speak a string aloud (text-to-speech)
`textToSpeechSupported()`	returns true if the device supports text-to-speech and the `speak` method
`speechToTextSupported()`	returns true if the device supports speech-to-text
`speechToText("`***prompt***`");`	initiate speech-to-text
`onSpeechToTextReady(`***text***`)`	called when speech-to-text is ready

```
// convenience methods for speech
SimpleSpeech.with(this).speak("Hello, world!");
SimpleSpeech.with(this).speechToText("Say your name");
...
public void onSpeechToTextReady(String theName) { ...
```

SimpleCamera

Method	Description

`takePhoto();`
`takePhoto(`***`filename`***`);`
	initiates taking a photo
	(if filename passed, saves it)

`photoGallery();`	launches photo gallery activity
`cameraExists()`	returns true if device has a camera
`onPhotoReady(`***`bitmap`***`)`	override this to capture the photo after it is taken/chosen

```
// make it easy to take a photo with the camera
SimpleCamera.with(this).takePhoto();
...
public void onPhotoReady(Bitmap bitmap) {
    // write code here to process the photo

}
```

Starting/finishing activities

Method	Description
`startActivity(`***`Class`***`, "`***`paramName1`***`", `***`value1`***`, ..., "`***`paramNameN`***`", `***`valueN`***`);`	start another activity, passing it the given parameters
`startActivityForResult(`***`Class`***`, `***`resultCode`***`, "`***`paramName1`***`", `***`value1`***`, ..., "`***`paramNameN`***`", `***`valueN`***`);`	start an activity that will return a result using the given code
`finish("`***`paramName1`***`", `***`value1`***`, ...);`	end the current activity and pass back parameters
`finish(`***`resultCode`***`, "`***`paramName1`***`", `***`value1`***`, ...);`	end current activity with given code and parameters

```
// more easily launch another activity (examples)
startActivity(MyActivity2.class,
    "userName", myUserName, "id", userID);
...
finish("result", myResult, "details", myDetails);
```

Activity parameters

Method	Description
`getBooleanExtra("`***`name`***`")`	get boolean parameter
`getDoubleExtra("`***`name`***`")`	get double parameter
`getIntExtra("`***`name`***`")`	get integer parameter
`getLongExtra("`***`name`***`")`	get long integer parameter
`getStringExtra("`***`name`***`")`	get string parameter

```
// extracting parameters when an activity is called
// (equiv. to getIntent().getStringExtra)
String email = getStringExtra("emailAddress");
int age = getIntExtra("age");

// each method also has a default-value version
int age = getIntExtra("age", 40);
```

Activity instance state

Method	Description
`saveAllFields(`***`bundle`***`);`	store all fields' values into bundle
`restoreAllFields(`***`bundle`***`);`	load all fields' values from bundle
`@AutoSaveFields`	annotation on top of class to automatically save/restore fields' values when activity is loaded

```
// easily save/load all private instance variables (non-View types)
@Override
protected void onRestoreInstanceState(Bundle bundle) {
    super.onRestoreInstanceState(savedInstanceState);
    restoreAllFields(bundle);
}
...
// or, just put this on top of your class
@AutoSaveFields
public Class MyActivity extends SimpleActivity { ...
```

SimplePreferences

Method	Description
`set("`***name***`", `***value***`);`	sets an app preference
`getBoolean("`***name***`")`	returns an app preference
`getDouble("`***name***`")`	returns an app preference
`getInt("`***name***`")`	returns an app preference
`getLong("`***name***`")`	returns an app preference
`getString("`***name***`")`	returns an app preference

```
// easier version of SharedPreferences object
SimplePreferences.with(this)
    .set("username", "stepp");
...
String username = SimplePreferences.with(this)
    .getString("username");
```

App shared preferences

Method	Description
`setShared("`*`filename`*`", "`*`name`*`", `*`value`*`);`	sets a shared preference
`getSharedBoolean("`*`filename`*`", "`*`name`*`")`	returns a preference
`getSharedDouble("`*`filename`*`", "`*`name`*`")`	returns a preference
`getSharedInt("`*`filename`*`", "`*`name`*`")`	returns a preference
`getSharedLong("`*`filename`*`", "`*`name`*`")`	returns a preference
`getSharedString("`*`filename`*`", "`*`name`*`")`	returns a preference

System services

Method	Description
`dial("`***`phoneNumber`***`");`	launch phone dialer service

`map(`***`lat`***`, `***`lng`***`);`
`map(`***`lat`***`, `***`lng`***`, `***`zoom`***`);`
	launch maps service

`textMessage("`***`phoneNumber`***`");`
`textMessage("`***`phoneNumber`***`", "`***`message`***`");`
	launch SMS messaging service

`webBrowser("`***`url`***`");`	launch default web browser

```
// launch system services
// (these methods are in SimpleActivity)
dial("1-650-555-4444");
webBrowser("http://stanford.edu/");
```

Checking orientation

Method	Description
`isPortrait()`	true if in portrait orientation
`isLandscape()`	true if in landscape orientation

```
~~if (getResources().getConfiguration().orientation ==~~
~~        Configuration.ORIENTATION_LANDSCAPE) {~~
    // we are in landscape orientation
    ...
}

if (isLandscape()) { ... }
```

Accessing resources

Method	Description
`getResourceId(`***name***`, `***type***`)`	return ID for resource of given type, e.g. "drawable"
`getResourceName(`***id***`)`	return resource short name for ID, e.g. R.drawable.foo => "foo"
`getResourceFullName(`***id***`)`	return resource long name for ID, e.g. R.drawable.foo => "R.drawable.foo"

```
// convert between resource IDs and strings easily
// String pika = "pikachu"
String pika = getResourceName(R.drawable.pikachu);

// int id = R.drawable.pikachu
int id = getResourceId("pikachu", "drawable");
```

SimpleFragment

- Accessing fragments from a `SimpleActivity`:

```
Fragment myFrag = findFragmentById(R.id.myId);
```

- If your app uses fragments, you can also have your fragments extend `SimpleFragment`:

~~public class MyFragment extends Fragment {~~
```
public class MyFragment extends SimpleFragment {
```

  – Not a lot of functionality yet, but currently lets you access the `SimpleActivity` containing the fragment.

```
SimpleActivity act = getSimpleActivity();
...
```

Manipulating fragments

Method	Description

`findFragment(`***id***`)`
`findFragmentById(`***id***`)`
	return fragment with the given ID

`addFragment(`***containerID***`, `***fragment***`);`	add a new fragment into the given view as its container
`removeFragment(`***fragment***`);`	remove an existing fragment
`replaceFragment(`***containerID***`, `***fragment***`);`	replace a fragment with a new one
`hideFragment(`***fragment***`);`	make a fragment invisible
`showFragment(`***fragment***`);`	make a fragment visible

```
// convenience methods instead of FragmentManager
MyFragment frag = new MyFragment();
addFragment(R.id.mycontainerid, frag);
```

SimpleDialog

Method	Description
`showAlertDialog("`***`text`***`");`	display a message with OK button
`showCheckboxInputDialog("`***`item1`***`", "`***`item2`***`", ..., "`***`itemN`***`");`	set of checkboxes to choose from
`showConfirmDialog("`***`text`***`");`	display message with Yes/No buttons
`showInputDialog("`***`prompt`***`");`	prompt for input with text box
`showListInputDialog("`***`item1`***`", "`***`item2`***`", ..., "`***`itemN`***`");`	list of tappable items (choose 1)
`showMultiInputDialog("`***`prompt1`***`", "`***`prompt2`***`", ..., "`***`promptN`***`");`	prompt for input with many text boxes
`showRadioInputDialog("`***`item1`***`", "`***`item2`***`", ..., "`***`itemN`***`");`	set of radio buttons (choose 1)
`onAlertDialogClose(`***`dialog`***`)`	called when alert dialog closes
`onDialogCancel(`***`dialog`***`)`	called when any dialog is canceled
`onInputDialogClose(`***`dialog`***`, `***`input`***`)`	called when input / list / radio dialog closes
`onMultiInputDialogClose(`***`dialog`***`, `***`inputs`***`)`	called when checkbox / multi-input closes

** (many methods can accept other parameters to customize their behavior)*

Alert dialog example

```
// example of showInputDialog (in your activity class)
SimpleDialog.with(this).showInputDialog("What's your name?");
...
@Override
public void onInputDialogClose(AlertDialog dialog, String input) {
    toast("The user's name is " + input);
}

// example of showMultiInputDialog (in your activity class)
SimpleDialog.with(this).showMultiInputDialog(
                        "Username", "Email", "Password");
...
@Override
public void onMultiInputDialogClose(AlertDialog dialog, String[] inputs) {
    toast("username: " + inputs[0]);
    toast("email:    " + inputs[1]);
    toast("password: " + inputs[2]);
}
```

More dialog methods

Method	Description
`setDialogsCancelable(`***`boolean`***`);`	whether dialogs should have Cancel button
`setDialogsIcon(`***`id`***`);`	ID of drawable to show as icon on dialogs
`setDialogsTitle("`***`text`***`");`	text to show next to icon as dialogs' title

```
// methods to further customize dialog appearance
SimpleDialog.with(this).setDialogsCancelable(true);
SimpleDialog.with(this).setDialogsIcon(
    android.R.drawable.ic_dialog_alert);
SimpleDialog.with(this).setDialogsTitle("Security Warning");

SimpleDialog.with(this).showConfirmDialog("Unsafe! Continue?");
```

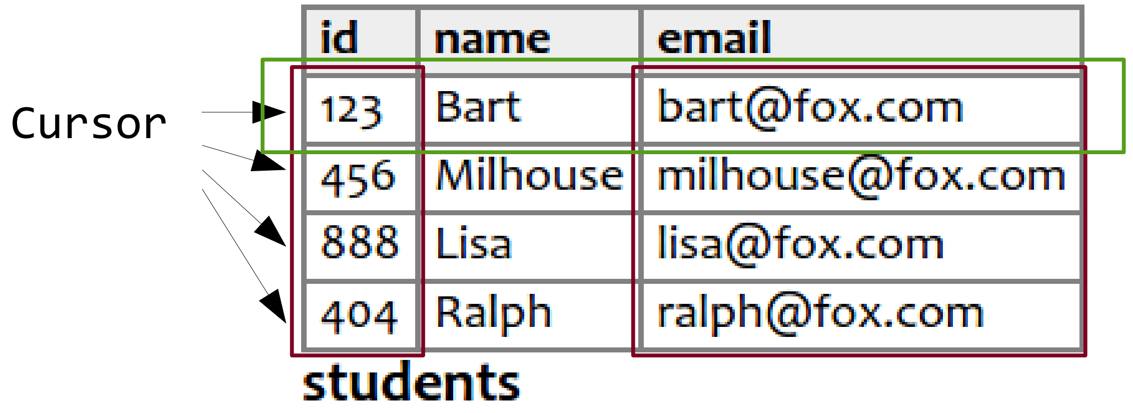
Dialog options in strings.xml

```
<resources>
    ...

    <!-- XML options to customize dialog appearance -->
    <bool name="dialogCancelable">true</bool>
    <string name="dialogDefaultTitle">Security Warning</string>
    <drawable name="dialogIcon">@android:drawable/ic_dialog_alert</drawable>
</resources>
```

Database access

```
// row object has same methods as Cursor and more
String query = "SELECT id, email FROM students";
for (SimpleRow row :
     SimpleDatabase.with(this).query("simpsons", query)) {
    int id = row.get("id");
    String email = row.get("email");
    ...
}
```



id	name	email
123	Bart	bart@fox.com
456	Milhouse	milhouse@fox.com
888	Lisa	lisa@fox.com
404	Ralph	ralph@fox.com

students

Importing a .sql file

- A .sql file contains a sequence of SQL commands.
  - Common format for exporting an entire database and its contents.
  - Used to save a backup or restore db to another server.

- To import a .sql file into an Android app:
  - Put the .sql file into your app's res/raw folder
  - Then use `executeSqlFile` method as shown below to import it!

```
// read file "example.sql" into a database named "example"
SimpleDatabase.with(this)
    .executeSqlFile(db, R.raw.example);

SimpleDatabase.with(this)
    .executeSqlFile("example");
```

Simple graphical canvas

- The library contains a `SimpleCanvas` class that more easily handles drawing and animation.

```
public class MyCanvas extends SimpleCanvas { ...
```

- There is also a `GCanvas` class that replicates much of the functionality of the Stanford Java library from CS 106A.

```
public class MyCanvas extends GCanvas { ...
```

  - `GCanvas` is a subclass of `SimpleCanvas`.

SimpleCanvas methods

Method	Description

`animate(`***framesPerSec***`);`
`animationPause();`
`animationResume();`
`animationStop();`
`isAnimated()`
	animation methods

`onAnimationTick()`	override for code to run on each anim. frame
`createFont(`***name***`, `***style***`)`	create a Typeface
`createPaint(`***red***`, `***green***`, `***blue***`)`	create a Paint

`drawBitmap(`***bmp***`, `***x***`, `***y***`);`
`drawOval(`***x1***`, `***y1***`, `***x2***`, `***y2***`);`
`drawRect(`***x1***`, `***y1***`, `***x2***`, `***y2***`);`
`drawRoundRect(`***x1***`, `***y1***`, `***x2***`, `***y2***`);`
`drawString("`***str***`", `***x***`, `***y***`);`
	draw various shapes and images

`setColor(`***Paint***`);`
`setColor(`***red***`, `***green***`, `***blue***`);`
	sets color for future drawing calls

`setFont(`***name***`, `***style***`, `***size***`);`	sets font for future drawing calls
`setFontSize(`***size***`);`	sets font size for future drawing calls
`setPaintStyle(`***paintStyle***`);`	sets paint style (stroked, filled, both)

GCanvas methods

Method	Description

`add(`***`gobject`***`);`
`add(`***`gobject`***`, `***`x`***`, `***`y`***`);`
	add graphical object to canvas at top of z-order

`contains(`***`gobject`***`)`	true if this graphical object is in canvas
`getElement(`***`index`***`)`	returns graphical object at given index in list
`getElementAt(`***`x`***`, `***`y`***`)`	top object at given pixel, or null if none
`getElementCount()`	returns number of graphical objects
`init()`	override this to write initialization code
`remove(`***`gobject`***`);`	remove graphical object from canvas
`removeAll();`	removes all graphical objects

`sendBackward(`***`gobject`***`);`
`sendForward(`***`gobject`***`);`
`sendToBack(`***`gobject`***`);`
`sendToFront(`***`gobject`***`);`
	adjust object's position in Z-ordering

Types of GObjects

Class	Description
`GColor`	class with many Paint constants including BLACK, BLUE, RED, WHITE, etc.
`GCompound`	container for treating other objects as a group
`GImage`	represents a bitmap image
`GLabel`	a text string drawn in a given font
`GLine`	connection between two points
`GObject`	superclass for other graphical object classes
`GOval`	a circle or ellipse
`GPolygon`	connects arbitrary points to form a polygon
`GRect`	a square or rectangle
`GSprite`	wraps a GObject and adds methods useful for games

- For details on each type of `GObject`, visit the library Javadoc page.
- Many methods and behaviors match the Stanford 106A library .

SimpleActivity game methods

Method	Description
`setWakeLock(`***`boolean`***`);`	set whether wake lock should be on/off
`wakeLockIsEnabled()`	returns true if you called setWakeLock(true); before
`setFullScreenMode(`***`boolean`***`);`	set whether app should go into full screen mode

SimpleLocalization

Method	Description
`with(`***`context`***`)`	get a SimpleLocalization instance
`format(`***`id, args`***`)`	format a resource string
`get(`***`id`***`),  get(`***`id, args`***`)`	look up a resource string

`isLTR(),  isLTR(`***`locale`***`),`
`isRTL(),  isRTL(`***`locale`***`)`
	return whether locale is right-to-left

`date(`***`date`***`),  date(`***`locale`***`)`	format a Date for this locale

`currency(`***`amount`***`),`
`currency(`***`amount, locale`***`)`
	format an amount of money for this locale

`number(`***`n`***`),`
`number(`***`n, locale`***`)`
	format a number for this locale

`parseLocalizedInt/Long/`
`Double/Float(`***`numStr`***`)`
	parse string into a number

`pluralize(`***`id, n, args`***`)`	look up a quantity string

BroadcastReceiver help

- A `SimpleActivity` can act as a broadcast receiver.
  - No need for intent filter or separate broadcast receiver class.
  - Just override the `onBroadcastReceived` method.

```
public class ActivityClassName extends SimpleActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // register for any broadcasts you want to receive
        // (no need for IntentFilter or BroadcastReceiver class)
        registerReceiver("action1", "action2", ..., "actionN");
    }

    @Override
    public void onBroadcastReceived(Intent intent) {
        ...
    }
}
```

SimpleNotification

- Stanford library class `SimpleNotification` extends `Notification.Builder` with convenience methods:

`send()`	- combines build() with NotificationManager
`setIntent(...)`	- simpler syntax for a pending intent
`addAction(...)`	- simpler syntax for an action

```
// example
SimpleNotification.with(this)
        .setContentTitle("title")
        .setContentText("text")
        .setSmallIcon(R.drawable.icon)
        .setIntent(MyActivity.class, parameters)
        .addAction(iconID1, "title1", MyActivity1.class, params)
        .addAction(iconID2, "title2", MyActivity2.class, params)
        .send();
```