Dialogs

Creating Dialogs

- The Android API supports the following types of Dialog objects:
  - AlertDialog
    A dialog that can manage zero, one, two, or three buttons, and/or a list of selectable items that can include checkboxes or radio buttons.
  - ProgressDialog
    A dialog that displays a progress wheel or progress bar. Because it's an extension of the AlertDialog, it also supports buttons.
  - DatePickerDialog
  - TimePickerDialog

Showing Dialog

- showDialog(int)
  - Shows a Dialog. Accepts an integer which uniquely identifies the Dialog.
- onCreateDialog(int)
  - Called when the Dialog is created for the first time.
  - This is the method where you should actually create a Dialog .
- onPrepareDialog(int, Dialog)
  - This method is called before the Dialog is displayed every time.
  - Define this method if you want to change any properties of the dialog each time it is opened.

Example

```
static final int DIALOG_PAUSED_ID = 0;
static final int DIALOG_GAMEOVER_ID = 1;
protected Dialog onCreateDialog(int id) {
        Dialog dialog;
        switch(id) {
        case DIALOG_PAUSED_ID:
           // do the work to define the pause Dialog
           break;
        case DIALOG_GAMEOVER_ID:
           // do the work to define the game over Dialog
           break;
        default:
           dialog = null;
        }
        return dialog;
    }
```

Dismissing a Dialog

- dismiss()
  - Called on the Dialog object itself to dismiss itself.
- dismissDialog(int)
  - Called from an Activity when a Dialog box is to be dismissed.
- removeDialog(int)
  - Removes the state of the Dialog, which is maintained by the onCreateDialog() method.

Dialog dismiss listener

- If the application need to perform some procedures when the dialog is dismissed, you should attach an on-dismiss listener to your Dialog.
- First define the DialogInterface.OnDismissListener interface. This interface has just one method, onDismiss(DialogInterface), which will be called when the dialog is dismissed.
  Then simply pass your OnDismissListener implementation to setOnDismissListener().
- A Dialog will be cancelled when user presses the back key or call the cancel() method.
- In case of cancel, the onDismisslistener will be still notified.
  If it is needed to listen to Dialog cancel event, DialogInterface.onCancelListener() can be used.

Creating an AlertDialog

- An AlertDialog is an extension of the Dialog class.
- An AlertDialog could be used when we need
  - A title
  - A text message
  - One, two, or three buttons
  - A list of selectable items
- To create an AlertDialog, use the AlertDialog.Builder subclass.
- Get a Builder with AlertDialog.Builder(Context) and then use the class's public methods to define all of the AlertDialog properties.
  After you're done with the Builder, retrieve the AlertDialog object with create().

Alert Dialog - Example

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setMessage("Are you sure you want to exit?")
        .setCancelable(false)
        .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
           public void onClick(DialogInterface dialog, int id) {
              MyActivity.this.finish();
           }
        })
        .setNegativeButton("No", new DialogInterface.OnClickListener() {
           public void onClick(DialogInterface dialog, int id) {
              dialog.cancel();
           }
        });
    AlertDialog alert = builder.create();
```

Alert Dialog – Adding a list

```
final CharSequence[] items = {"Red", "Green", "Blue"};

    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Pick a color");
    builder.setItems(items, new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int item) {
            Toast.makeText(getApplicationContext(), items[item],
    Toast.LENGTH_SHORT).show();
        }
    });
    AlertDialog alert = builder.create();
```

To create a list of multiple-choice items (checkboxes) or single-choice items (radio buttons) inside the dialog, use the setMultiChoiceItems() and setSingleChoiceItems() methods, respectively.

Progress Dialog

- A ProgressDialog is an extension of the AlertDialog class that can display a progress animation in the form of a spinning wheel, for a task with progress that's undefined, or a progress bar, for a task that has a defined progression.
- The dialog can also provide buttons, such as one to cancel.
- Opening a progress dialog can be as simple as calling ProgressDialog.show().

```
ProgressDialog dialog = ProgressDialog.show(MyActivity.this, "",
                "Loading. Please wait...", true);
```

- The first parameter is the application Context, the second is a title for the dialog (left empty), the third is the message, and the last parameter is whether the progress is indeterminate

Showing a progress Dialog

```
ProgressDialog progressDialog;
progressDialog = new ProgressDialog(mContext);
progressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
progressDialog.setMessage("Loading...");
progressDialog.setCancelable(false);
```

Creating Custom Dialog

- you can create your own layout for the dialog window with layout and widget elements.
- After you've defined your layout, pass the root View object or layout resource ID to setContentView(View).

- *Refer the example CustomDialogDemo*