EPFL

Lab on apps development for tablets, smartphones and smartwatches

Week 4: Fragments and User Interface

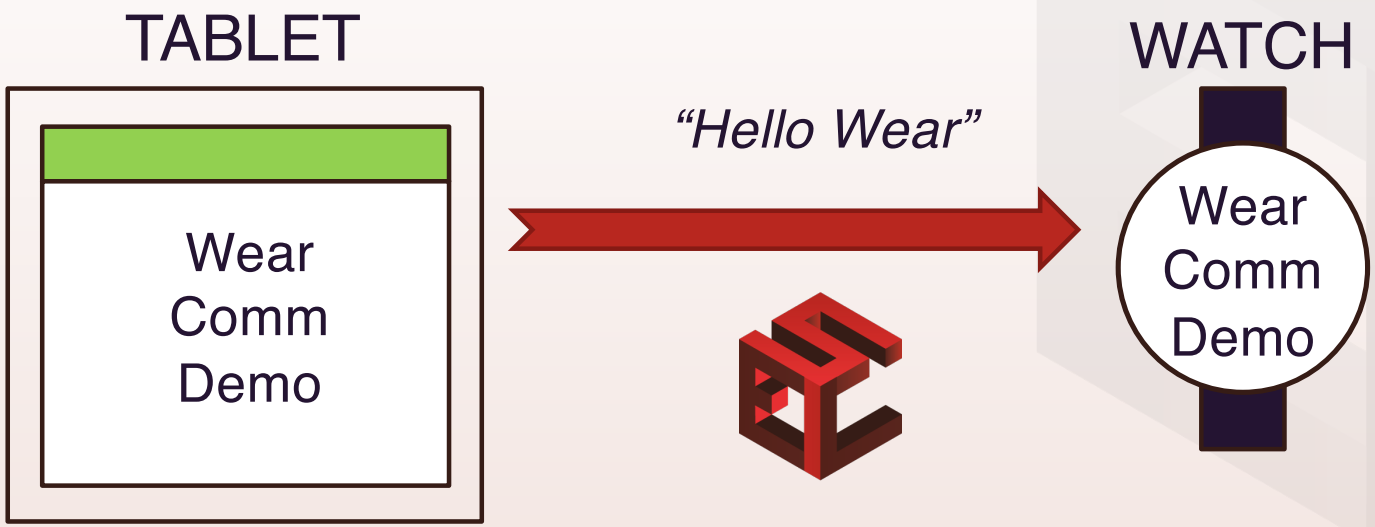
Dr. Giovanni Ansaloni, Prof. David Atienza

Ms. Halima Najibi, Ms. Farnaz Forooghifar,
Mr. Renato Zanetti, Mr. Saleh Baghersalimi, Mr. Alireza Amirshahi

Institute of Electrical Engineering (IEL) – Faculty of Engineering (STI)

Announcements

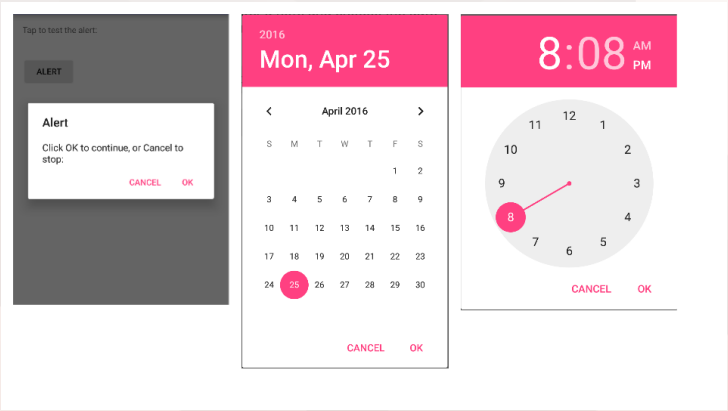
- WearCommunicationDemo available on Moodle
  - Message API: string
  - Data API: Bitmap image
  - Explicit intents on sender between Activity and WearService
  - Implicit intents on receiver side between WearService and Activity



Outline of the class

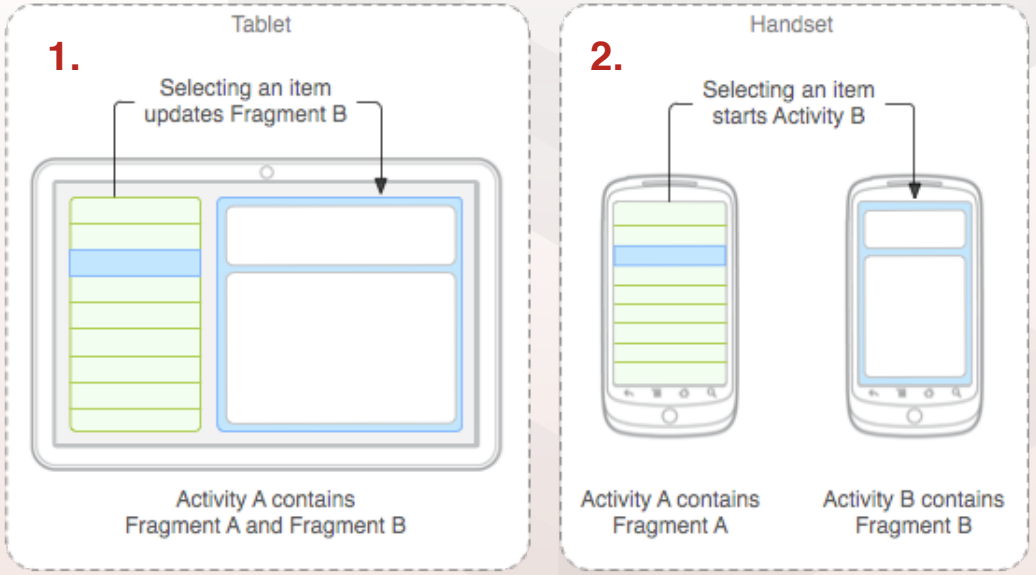
- **Fragments**

- User interaction
  - Buttons, text fields and spinners
  - Dialogs
  - Toasts
  - Menus



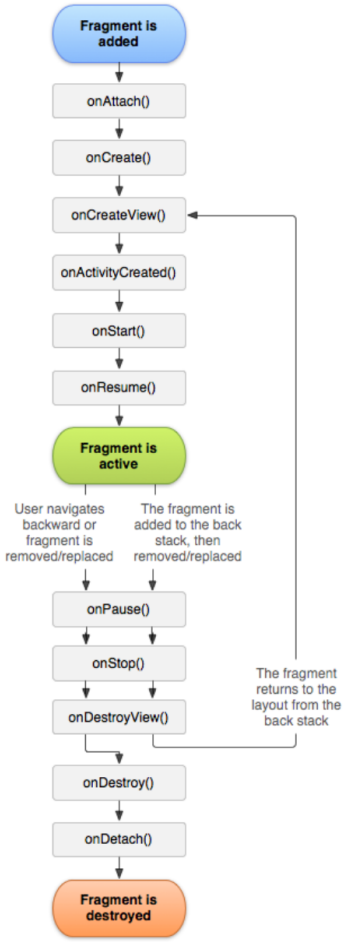
What is a Fragment?

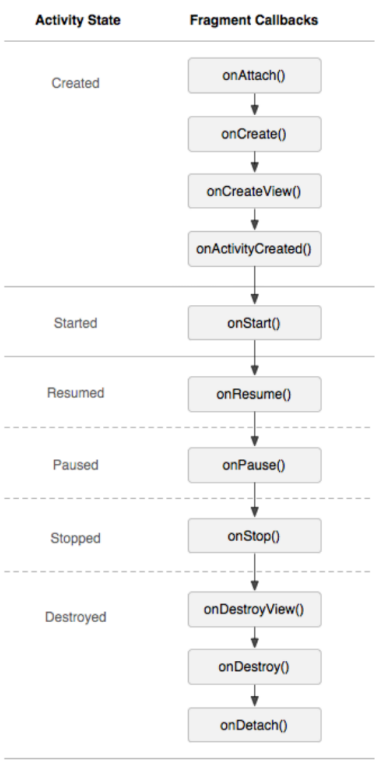
- A **Fragment** is a portion of a user activity
- Multiple fragments can be combined into a single activity in a multi-pane UI
- Fragments can be reused in more than one activity
- Introduced in Android 3 (API 11) to support more dynamic and flexible UIs



Lifecycle of a Fragment

- A fragment has its own lifecycle
  - But its lifecycle is affected by the activity
    - When the activity is paused → fragments are paused
    - When the activity is destroyed → fragments are destroyed
- Usually, we should implement:
  - onCreate() → initialize essential components
  - onCreateView() → called when it draws the UI for the first time (returns a View)
  - onPause() → to commit changes





Adding a fragment to an activity

- Two ways of adding a fragment to an activity:
  1. Via the activity layout XML file
     - The <android:name> clause specifies the Fragment subclass to instantiate
     - Each fragment requires a unique identifier that the system can use to restore the fragment if the activity is restarted.
       - android:id or android:tag

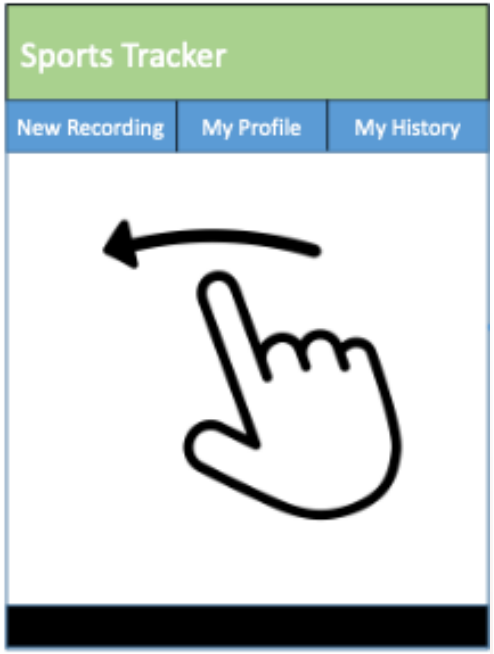
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment android:name="com.example.news.ArticleListFragment"
            android:id="@+id/list"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
    <fragment android:name="com.example.news.ArticleReaderFragment"
            android:id="@+id/viewer"
            android:layout_weight="2"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
</LinearLayout>
```

  2. Programmatically, adding the fragment to a ViewGroup using FragmentManager
     - Using the FragmentTransaction to add/remove/replace a Fragment

```
FragmentManager fragmentManager = getFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

ExampleFragment fragment = new ExampleFragment();
fragmentTransaction.add(R.id.fragment_container, fragment);
fragmentTransaction.commit();
```

Managing fragment within containers(1)

- Example: Viewpager (as in Lab4)
  1. Create the fragment modules (layout and java files)
  2. Create a class extending FragmentStatePageAdapter
     - Helper class incapsulating FragmentManager
  3. Add a ViewPager element in the layout of the activity containing the fragments




```
<androidx.viewpager.widget.ViewPager
    xmlns:android="http://schemas.android.com/apk/res/and
    android:id="@+id/mainViewPager"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.viewpager.widget.PagerTabStrip
        android:id="@+id/pagerTabStrip"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="top"
        android:background="#20B2AA"
        android:paddingTop="15dp"
        android:paddingBottom="15dp"
        android:textColor="#fff" />

</androidx.viewpager.widget.ViewPager>
```

Managing fragment within containers(2)

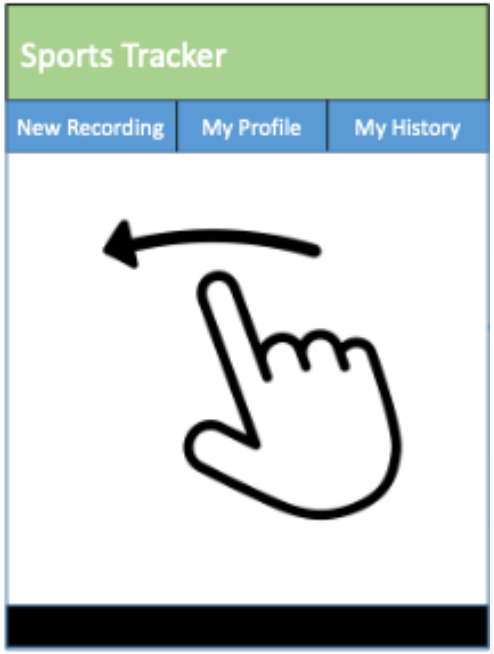
- Example: Viewpager (continued)
  4. Setup ViewPager and Fragments in containing activity

```
public class MainActivity extends AppCompatActivity implements NewRecordingFragment
        .OnFragmentInteractionListener, MyProfileFragment.OnFragmentInteractionListener,
        MyHistoryFragment.OnFragmentInteractionListener {

    private NewRecordingFragment newRecFragment;
    private MyProfileFragment myProfileFragment;
    private MyHistoryFragment myHistoryFragment;
    private SectionsStatePagerAdapter mSectionStatePagerAdapter;
    protected void onCreate(Bundle savedInstanceState) {
        //...
        mSectionStatePagerAdapter = new SectionsStatePagerAdapter(getSupportFragmentManager());
        myProfileFragment = new MyProfileFragment();
        newRecFragment = new NewRecordingFragment();
        myHistoryFragment = new MyHistoryFragment();

        ViewPager mViewPager = findViewById(R.id.mainViewPager);
        mSectionStatePagerAdapter.addFragment(myProfileFragment, getString(R.string.tab_title_my_profile));
        mSectionStatePagerAdapter.addFragment(newRecFragment, getString(R.string.tab_title_new_recording));
        mSectionStatePagerAdapter.addFragment(myHistoryFragment, getString(R.string.tab_title_history));

        mViewPager.setAdapter(mSectionStatePagerAdapter);
        mViewPager.setCurrentItem(mSectionStatePagerAdapter.getPositionByTitle("New Recording"));
        //
```

Implement listeners for each fragment

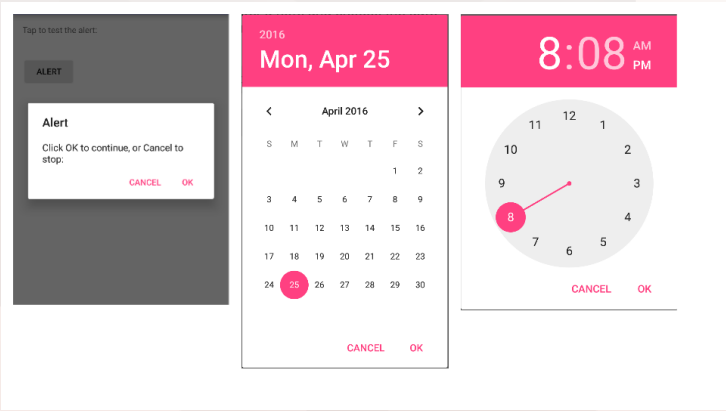
Instantiate fragments, adapter

Add fragments to adapter



Outline of the class

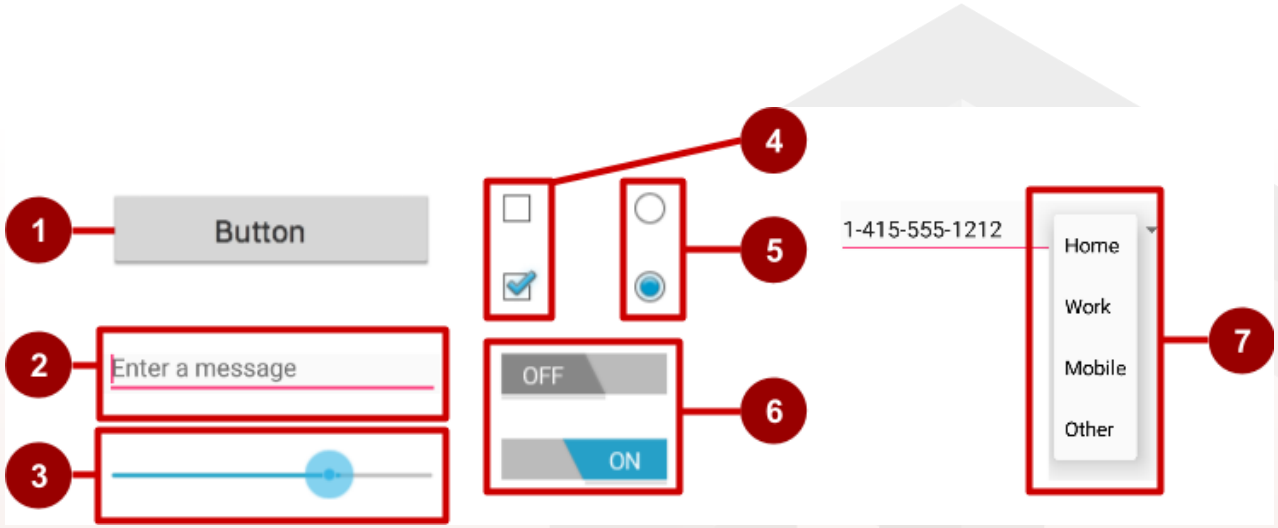
- Fragments
- User interaction
  - **Buttons, text fields and spinners**
  - Dialogs
  - Toasts
  - Menus



User input controls

- User input controls:

1. Button
2. Text field
3. Seek bar
4. Checkboxes
5. Radio buttons
6. Toggle
7. **Spinner**

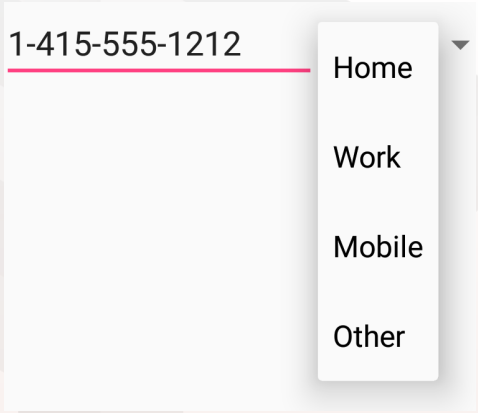


- For further usage details:
  https://docs.google.com/presentation/d/1YY40Zc_44-LnuRc_S849WNabjQDmt9NTgsC4iAFAyrs

Spinners

- Spinner: Quick way to select value from a set
  - Drop-down list of all values, users can select only one

- Implementing Spinners:
  1. Create Spinner UI element in the XML layout
  2. Define spinner choices in an array
  3. Instantiate Spinner in activity
  4. Create an adapter with default spinner layouts
  5. Attach the adapter to the spinner
  6. Implement onItemSelectedListener method



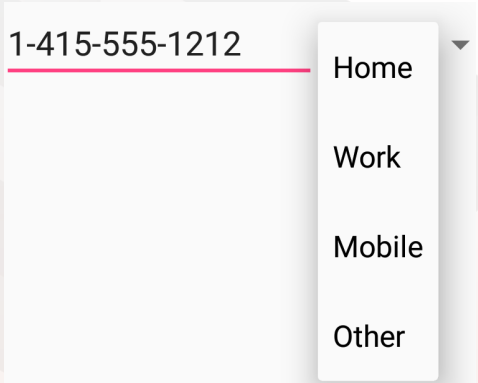
Instantiating a spinner (1)

1. In layout XML file create spinner element

```
<Spinner
   android:id="@+id/label_spinner"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content">
</Spinner>
```

2. In arrays.xml resource file define choices

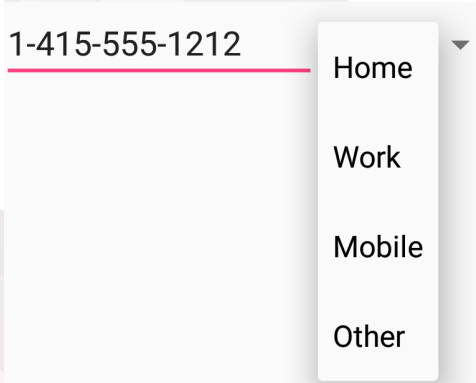
```
<string-array name="labels_array">
    <item>Home</item>
    <item>Work</item>
    <item>Mobile</item>
    <item>Other</item>
</string-array>
```



Instantiating a spinner (2)

3. Instantiate spinner

```
// In onCreate()
//...
Spinner spinner = (Spinner) findViewById(R.id.label_spinner);
if (spinner != null) {
      spinner.setOnItemSelectedListener(this);
//...
```

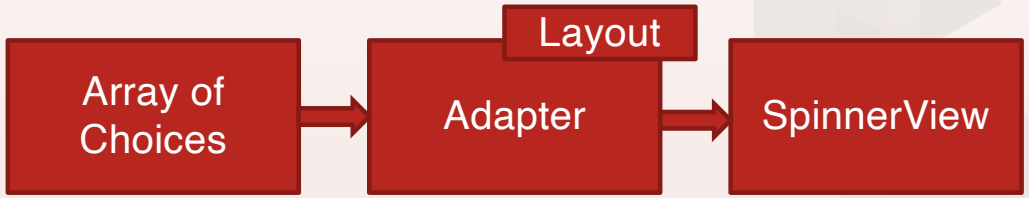


4. Create adapter for the spinner

```
ArrayAdapter<CharSequence> adapter =
  ArrayAdapter.createFromResource(
        this, R.array.labels_array,
        // Layout for each item
        android.R.layout.simple_spinner_item);
```

What is an adapter?

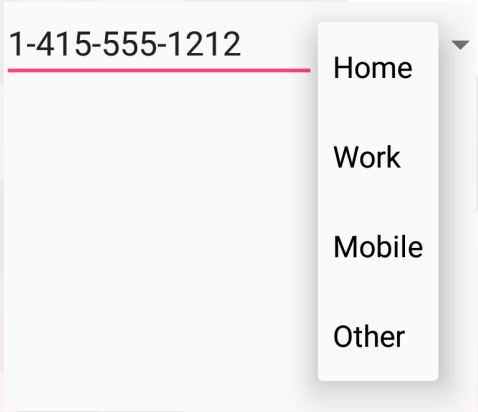
- An **adapter** is a bridge between data sources and UI components
- It pulls content from a source such as an array and converts each item result into a view that's placed into the layout
- When the content for your layout is dynamic or not pre-determined, the items are automatically inserted to the layout using an adapter



Instantiating a spinner (3)

5. Attach the adapter to the spinner

```
adapter.setDropDownViewResource(
    android.R.layout.simple_spinner_dropdown_item);
spinner.setAdapter(adapter);
```

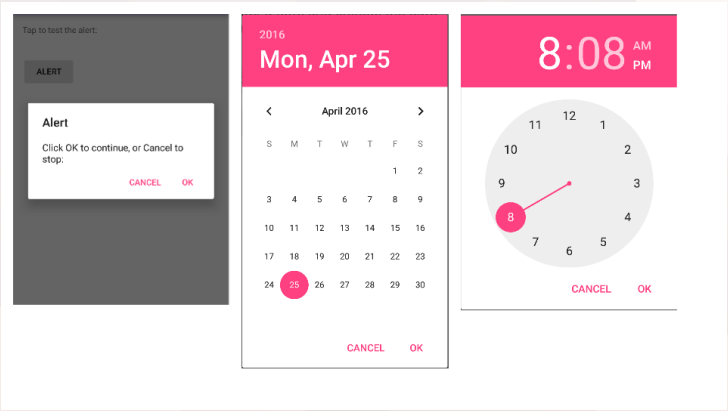


6. Implement onItemSelectedListener method

```
public class MainActivity extends AppCompatActivity
        implements AdapterView.OnItemSelectedListener

    public void onItemSelected(AdapterView<?> adapterView,
        View view, int pos, long id) {
            String spinner_item =
                adapterView.getItemAtPosition(pos).toString();
            // Do something here with the item
    }
```

Outline of the class

- Fragments
- User interaction
  - Buttons, text fields and spinners
  - **Dialogs**
  - **Toasts**
  - Menus

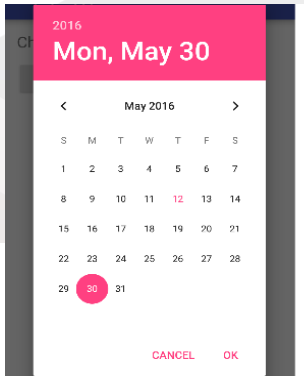


Dialogs

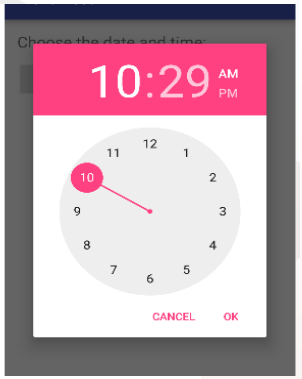
- Dialogs appear on top, interrupting the flow of the activity, and require an action to be dismissed
- Different types:
  - Alert dialog, date picker, time picker
- AlertDialog can show:
  1. Title
  2. Content area
  3. Action buttons

```
public void onClickShowAlert(View view) {
    AlertDialog.Builder alertDialog = new
                        AlertDialog.Builder(MainActivity.this);
    alertDialog.setTitle("Connect to Provider");
    alertDialog.setMessage(R.string.alert_message);
    ...
```

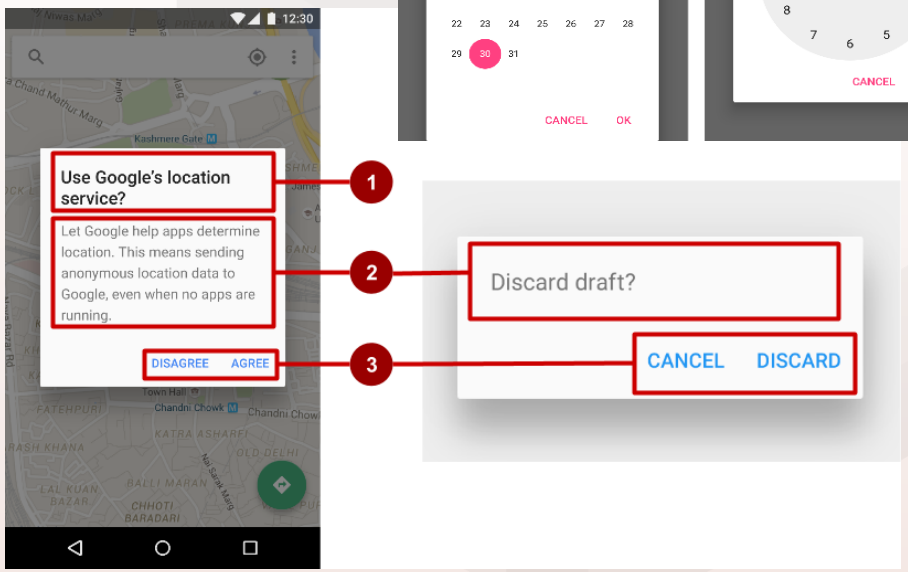
DatePicker



TimePicker



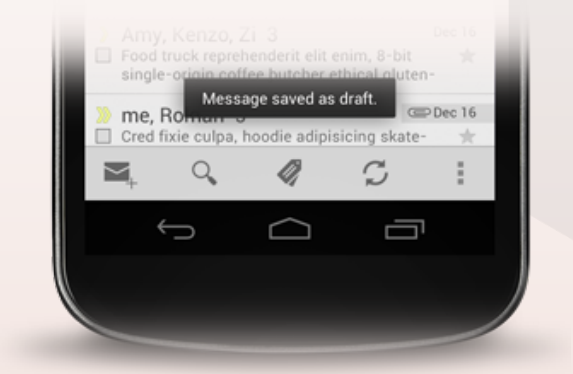
AlertDialog



Toasts

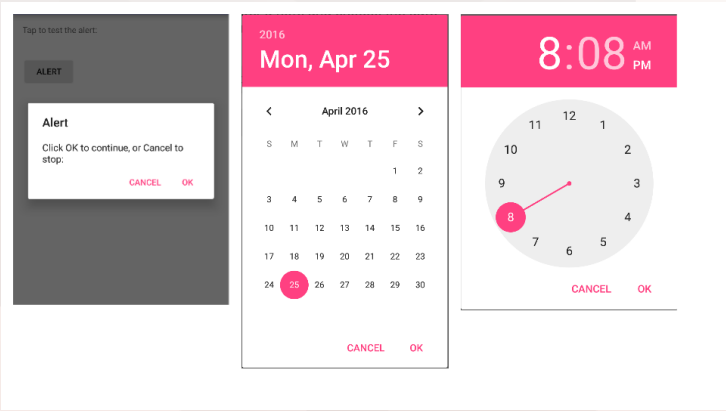
- Tiny messages over the Activity
- Used to signal to the user some confirmation, error, etc.
- Can control the duration of the Toast
- As simple as:

```
Toast msg = Toast.makeText(this, “Toast!”, Toast.LENGTH_SHORT).show();
```



Outline of the class

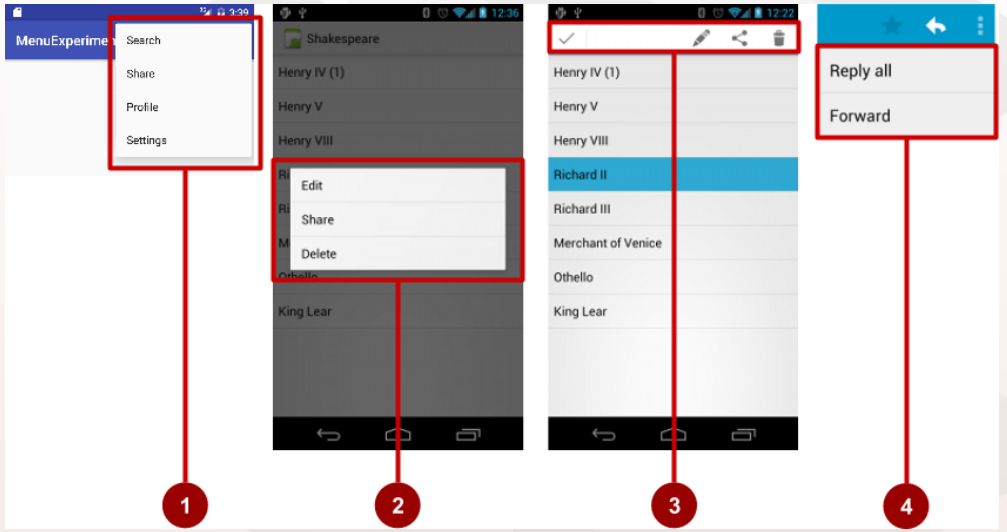
- Fragments
- User interaction
  - Buttons, text fields and spinners
  - Dialogs
  - Toasts
  - **Menus**



Menus

- They appear whenever the user presses the menu button
- Useful for giving different options without leaving the current Activity
  - Your projects should have menus!! → at least one, please!

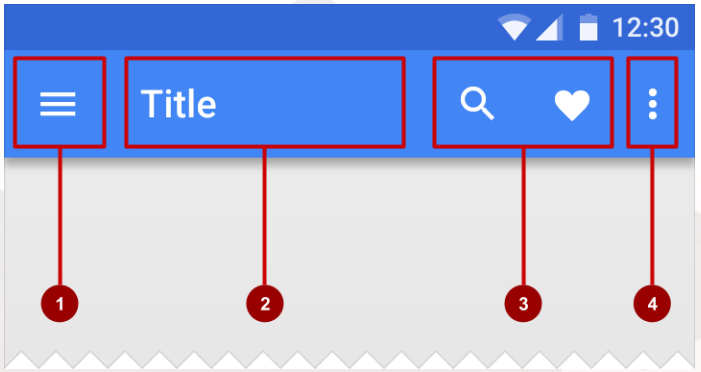
- Types of menus
  1. Application bar with options menus
  2. Contextual menu
  3. Contextual action mode
  4. Popup menu



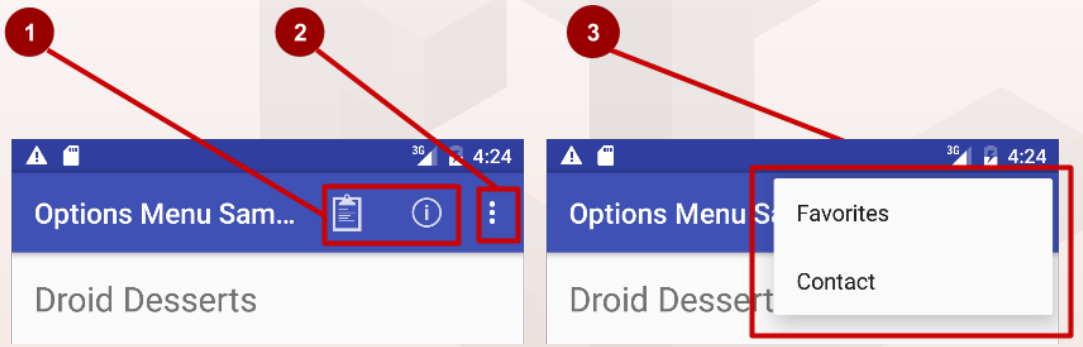
- Documentation: *https://developer.android.com/guide/topics/ui/menus*

Application bar with options menu

- Bar at the top of each screen, usually the same for all screens
  1. Navigation icon to open navigation drawer
  2. Title of the current activity
  3. Icons for **options menu** items
  4. Action overflow button for rest of options

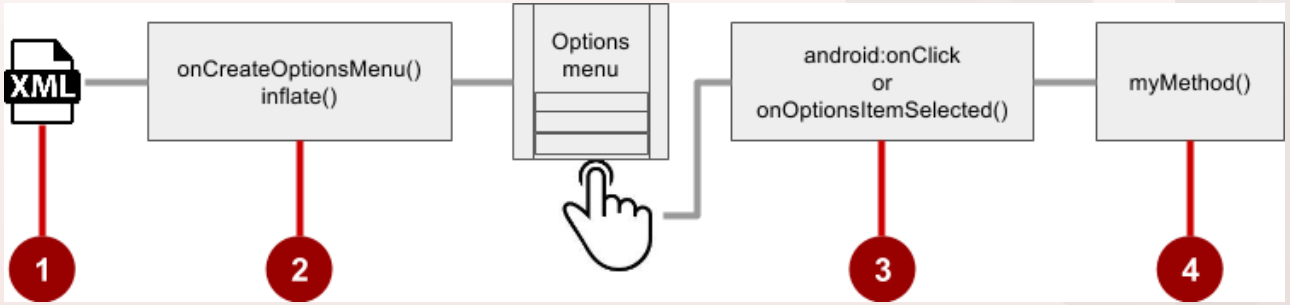


- What is the options menu?
  - Actions for important items (1)
  - By tapping the overflow part (2) you get more options



Steps to implement options menu

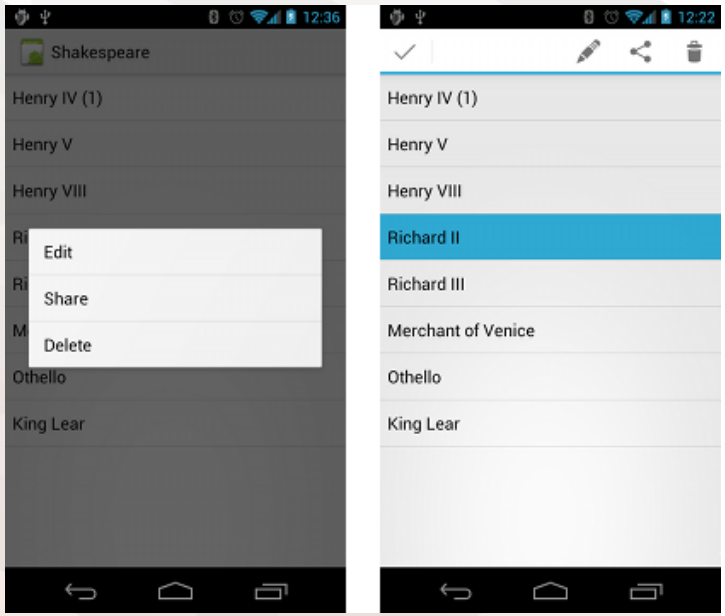
- As always, we develop the menu in XML and Java:
  1. XML menu resource (menu_main.xml)
     - Placing new file inside “res/menu”
  2. onCreateOptionsMenu() to inflate the menu inside the activity
  3. onClick attribute or onOptionsItemSelected()
  4. Method to handle item click



Contextual menus

- Allow users to perform an action on a selected view or content
- Can be deployed on any View object

- Two types:
  - Floating context menus
    - Floating list of menu items
    - Users can modify the View element or use it
    - Users perform a contextual action
  - Contextual action mode
    - Temporary action bar in place of the app bar



Floating context menus

- Allow users to perform an action on a selected view or content

1. Register View:

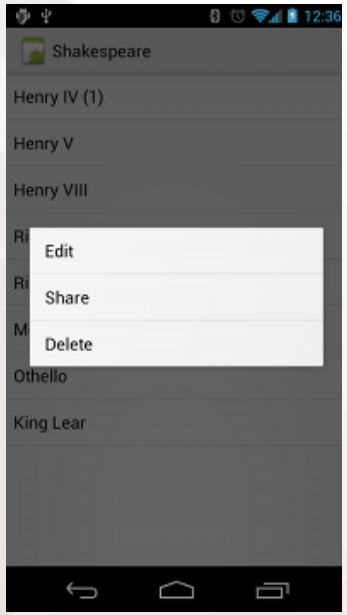
```
registerForContextMenu(View)
```

2. Link the menu item (from XML) with the menu

```
public void onCreateContextMenu(ContextMenu menu, View v,
                                ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.context_menu, menu);
}
```

3. State the action to be done when an item is selected

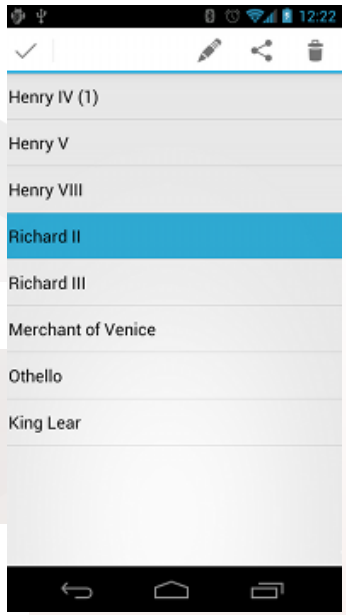
```
public boolean onContextItemSelected(MenuItem item) {
 ...
```



Contextual action mode

- Similar to floating context menu
  - `onCreateActionMode(), onActionItemClicked()`
  - Methods defined inside an object extending the `ActionMode.Callback()` interface

```
actionModeCallback = new ActionMode.Callback(){ …
```

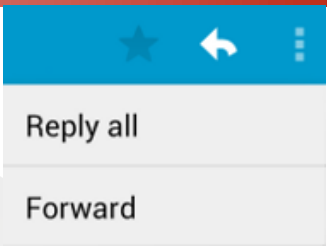


- StartActionMode() to enable the contextual action

```
someView.setOnLongClickListener(new View.OnLongClickListener() {
    public boolean onLongClick(View view) {
       actionMode = getActivity().startActionMode(actionModeCallback);
       view.setSelected(true);
       return true;} };
```

Popup menu

- A list of items anchored to a view (visible icon)
  - For example, in an email app, Reply All and Forward



Show the menu (called e.g. when a button is pressed)

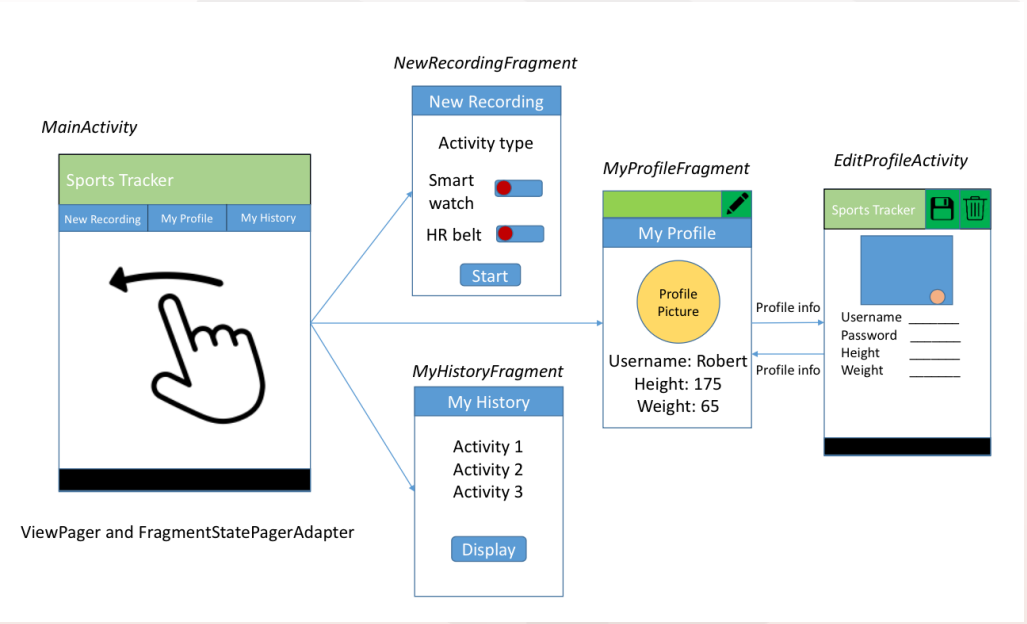
```
public void showMenu(View v) {
    PopupMenu popup = new PopupMenu(this, v);
    popup.setOnMenuItemClickListener(this);
    popup.inflate(R.menu.actions);
    popup.show();
}
```

Do something with selected item

```
@Override
public boolean
            onMenuItemClick(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.archive:
            archive(item);
            return true;
        case R.id.delete:
            delete(item);
            return true;
        default:
            return false;
    }
}
```

Today's lab

- Adding **fragments** to our sports tracker app
  - The ViewPager layout
  - Moving contents from MainActivity to ViewPager
- UI: Toasts, **menus**, dialogs…
  - Adding an action bar menu



Questions?