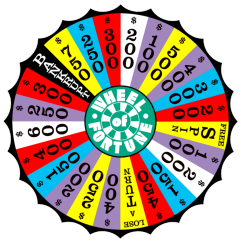
CSC 472 / 372
Mobile Application Development for Android

Prof. Xiaoping Jia
School of Computing, CDM
DePaul University
xjia@cdm.depaul.edu
@DePaulSWEng

Outline

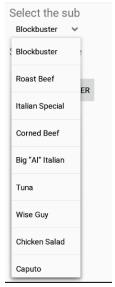
- *Adapters* and *Adapter Views*
- *Spinners*
- *List Views* and *List Activities*
- Customize *List Views* and *Adapters*
- Master-detail apps
- Implement *Parcelable*

Spinners



Spinners

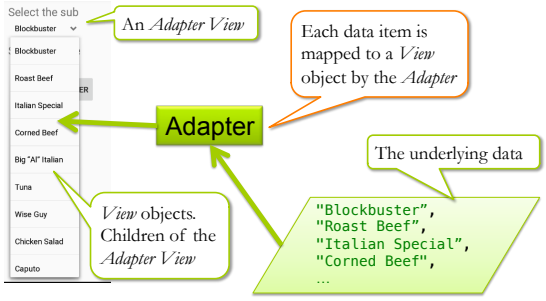
- A drop-down list that allows users to select one value from a list.
  - The drop-down list is scrollable if necessary
- Class: `android.widget.Spinner`
  - Subclass of: `AdapterView`
  - The items are populated using an *adapter*
- XML element: `<Spinner>`



Adapter and Adapter View

- An *Adapter View* is a *View Group*, whose children are provided by an *adapter*.
  - Including *Spinners*, *List Views*
- An *Adapter* is a bridge between an *Adapter View* object and the underlying data of the view.
  - Manages a list/array of data items
    - The data can be static or dynamic
  - Provides access to the date items
  - Creates a *View* object for each data item
    - Creates *View* objects on-demand
    - May reuse *View* objects

The Role of an Adapter



Array Adapter

- A commonly used simple adapter
- The data source
  - an array of strings or arbitrary objects (with `toString()` method defined)
- Maps each item to a *Text View* object
  - Display the `toString()` value of each item
- It is possible to extend the class to
  - Use other View objects, e.g., *Image View*
  - Display other data besides `toString()`

Will see an example later.

DePaul University

Create an Array Adapter

```
new ArrayAdapter<T>(context, resId, array);
```

- *T*: the type of the contents, i.e., data items
- *context*: the app context, the hosting *Activity*
- *resId:* the layout resource id, either
  - a *Text View* for displaying a string, or
  - a container that contains a *Text View* for displaying a string
- *array:* the data array, items are of type *T*

The basic use

Allow customization

DePaul University

Handle Selection for Spinners

- Listener interface:
  `AdapterView.OnItemSelectedListener`
  - Action methods

```
public void onItemSelected(AdapterView parent,
                           View view, int pos, long id)
```

When an item is selected. The selected item can be retrieved

```
parent.getItemAtPosition(pos)
```

```
public void onNothingSelected(AdapterView parent)
```

When no item is selected

DePaul University

"Order My Sub" App

"I'd Like to Order My Sub"

- A simple app to order a sub sandwich
- Two *Spinners*
  - Spinner 1: select the sandwich type
  - Spinner 2. select the sandwich size
- "Place Your Order" *Button*
- Display a *Toast* of your order

DePaul University

"Order My Sub" – The Layout: activity_main.xml

```
<LinearLayout … >
  <TextView … android:text="Select the sub" />
  <Spinner
     android:id="@+id/spinner1"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"/>
  <TextView … android:text="Select the size"/>
  <Spinner
     android:id="@+id/spinner2"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"/>
  <Button … android:id="@+id/button"
    android:text="Place Your Order"/>
</LinearLayout>
```

DePaul University

"Order My Sub" Activity – The Spinner Data

```
public class MainActivity extends Activity {
  …
  static final String[] SUBS = {
       "Blockbuster",
       "Roast Beef",
       …
       "American"
  };
  static final String[] SIZES = {
       "6 inch",
       "8 inch",
       …
       "3 foot"
  };
}
```

The data arrays for *Array Adapter*

"Order My Sub" Activity – Initialize the Spinners

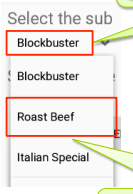
```
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);

  final Spinner s1 = (Spinner) findViewById(R.id.spinner1);
  ArrayAdapter<String> adapter =
    new ArrayAdapter<String>(this,
         android.R.layout.simple_spinner_item, SUBS);
  adapter.setDropDownViewResource(
      android.R.layout.simple_spinner_dropdown_item);
  s1.setAdapter(adapter);
  …
}
```

Create a *Array Adapter*
Set the adapter for spinner

"Order My Sub" Activity – The Layout Resources for Spinners

```
new ArrayAdapter<String>(this,
    android.R.layout.simple_spinner_item, SUBS);
```

Layout resource for displaying the spinner



Layout resource for displaying the drop-down list of the spinner

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


"Order My Sub" Activity – Handle Selection

```
protected void onCreate(Bundle savedInstanceState) {
  …
  s1.setOnItemSelectedListener(
    new AdapterView.OnItemSelectedListener() {
      public void onItemSelected(AdapterView<?> parent,
                   View view, int position, long id) {
         Log.d(TAG, "Spinner1: " + SUBS[position] +
                  " position=" + position + " id=" + id);
      }
      public void onNothingSelected(AdapterView<?> parent) {
         Log.d(TAG, "Spinner1: unselected");
      }
  });
  …
}
```

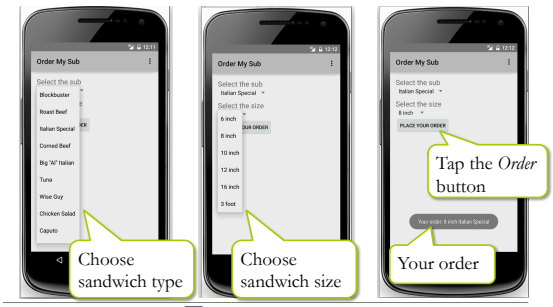
Log messages only. No action.

"Order My Sub" Activity – Button Action

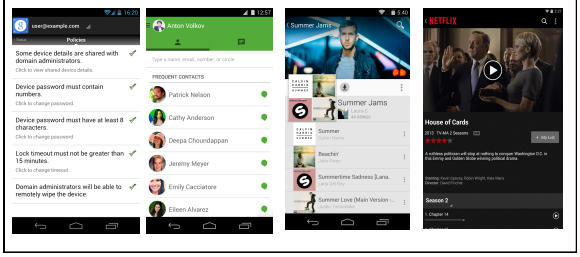
```
protected void onCreate(Bundle savedInstanceState) {
  …
  Button button = (Button) findViewById(R.id.button);
  button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
      Toast.makeText(MainActivity.this, "Your order: " +
        s2.getSelectedItem() + " " + s1.getSelectedItem(),
        Toast.LENGTH_SHORT).show();
    }
  });
}
```

Retrieve the selected items from the spinners.

"Order My Sub" App



List Views



List View

- A *View Group* that displays a list of scrollable items/rows.
  - Typically occupies the whole screen, or most of a screen
- The row contents are provided by an *adapter*
  - Bridge between the data source and the view objects
  - The *View* objects for rows are inflated dynamically
- The data source can be
  - Static, e.g., an array, or
  - Dynamic, e.g., a database

DePaul University

List Activity

- An *Activity* that hosts a *List View* should extend *List Activity*
  - A subclass of *Activity*
- Provide a default layout
  - A full-screen *List View*
  - No layout file is necessary if the default layout is used
- The *screen layout* and the *row layout* for the items displayed in the *List View* are customizable

Will see examples of both.

DePaul University

Handle List View Actions

- Option 1: Listener interface: `AdapterView.OnItemClickListener`

Compare with *Spinner*: `OnItemSelectedListener`

  - Action method

```
public void onItemClick(AdapterView parent,
                        View view, int pos, long id)
```

When an item is clicked. The selected item can be retrieved

```
parent.getItemAtPosition(pos)
```

- Option 2: Action method in *List Activity*

```
protected void onItemClick(ListView list,
                        View view, int pos, long id)
```

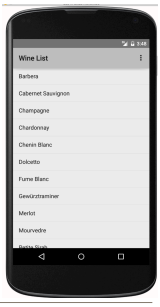
When an item is clicked.

DePaul University

The Wine List App

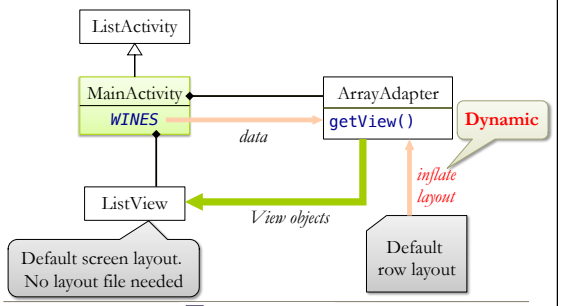
The "Wine List" App

- A simple list app displays a list of wine grapes, names only
- Use *List View* and *List Activity*
  - No layout file needed
- Use an *Array Adapter*
- Static data provided as an array
- Use the default row layout, i.e., a simple *Text View*



DePaul University

The Wine List App – Architecture Overview



DePaul University

Wine List Activity

The Superclass

No layout

The *Adapter*

The data

```
public class MainActivity extends ListActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //setContentView(R.layout.activity_main);
    setListAdapter(new ArrayAdapter<String>(this,
      android.R.layout.simple_list_item_1, WINES));
  }
  private static final String[] WINES = {
    "Barbera", "Cabernet Sauvignon", …
    "Zinfandel"
  };
}
```

DePaul University

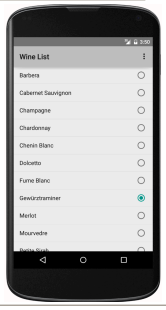
Wine List Activity – Handle List View Action

```
public class MainActivity extends ListActivity  {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //setContentView(R.layout.activity_main);
    setListAdapter(new ArrayAdapter<String>(this,
      android.R.layout.simple_list_item_1, WINES));
  }
  @Override
  protected void onListItemClick(ListView l, View v,
                                 int position, long id) {
    Log.d(TAG, "onListItemClick position=" + position +
          " id=" + id + " " + WINES[position]);
  }
}
```

Logs a message when an item is clicked

The Wine List App (Alt 1) – Single Selection Mode

- A *List View* with single selection mode, i.e., exclusive selection
- Each row contains a clickable *Radio Button*



DePaul University

Wine List (Alt 1) Activity – Single Selection Mode

```
public class MainActivity extends ListActivity  {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //setContentView(R.layout.activity_main);
    final ListView listView = getListView();
    listView.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
    setListAdapter(new ArrayAdapter<String>(this,
      android.R.layout.simple_list_item_single_choice,
      WINES));
  }
  …
}
```

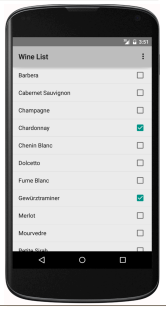
Set single choice mode

Use a row layout with a *Text View* and a *Radio Button*

DePaul University

The Wine List App (Alt 2) – Multiple Selection Mode

- A *List View* with multiple selection mode, i.e., non-exclusive selection
- Each row contains a clickable *Check Box*



DePaul University

Wine List (Alt 2) Activity – Multiple Selection Mode

```
public class MainActivity extends ListActivity  {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //setContentView(R.layout.activity_main);
    final ListView listView = getListView();
    listView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
    setListAdapter(new ArrayAdapter<String>(this,
      android.R.layout.simple_list_item_multile_choice,
      WINES));
  }
  …
}
```

Set multiple choice mode

Use a row layout with a *Text View* and a *Check Box*

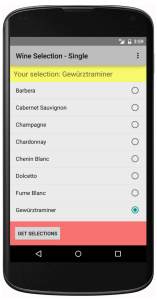
DePaul University

The Wine Selection App

✧ *Bacchus* by Caravaggio, c. 1595.
Oil on canvas. 95 cm × 85 cm.
Original in Uffizi Gallery, Firenze, Italy.
Image in Public Domain (US).

The Wine Selection App – Single Selection

- A variation of the *Wine List* app
- Customized screen layout
- Single selection mode
- Handle selection
  - Display the selection in the *Text View* at the top



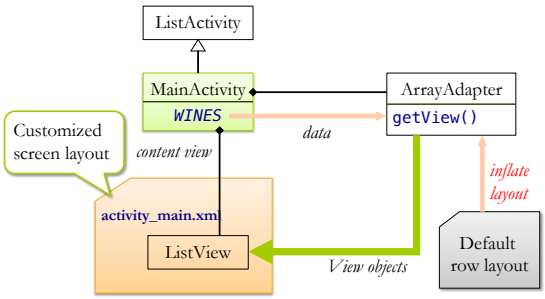
DePaul University

Screen Layout with a List View

- You can customize the screen layout of a *List Activity*
- Create a layout that
  - Must contain a *List View* object
  - The id of the *List View* object must be `"@android:id/list"` (or `list` in Java code)
- Set the layout with `setContentView()` in `onCreate()`

DePaul University

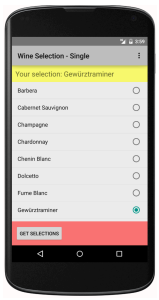
Wine Selection – Single – Architecture Overview



DePaul University

Wine Selection – Single – The Screen Layout

```
<LinearLayout … >
  <TextView android:id="@+id/text" … />
  <ListView
    android:id="@android:id/list"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"/>
  <LinearLayout … >
    <Button android:id="@+id/select" … />
  </LinearLayout>
</LinearLayout>
```



DePaul University

Wine Selection – Single – The List Activity

```
public class MainActivity extends ListActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main
    );
    final ListView listView = getListView();
    listView.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
    setListAdapter(new ArrayAdapter<String>(this,
      android.R.layout.simple_list_item_single_choice,
      WINES));
    …
  }
}
```

Set the screen layout

Set single selection mode

DePaul University

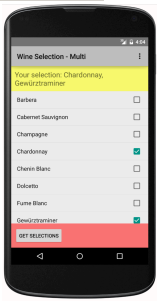
Wine Selection – Single – Display the Selected Item

```
protected void onCreate(Bundle savedInstanceState) {
  …
  final TextView textView =
     (TextView) findViewById(R.id.text);
  Button select = (Button) findViewById(R.id.select);
  select.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view) {
      int pos = listView.getCheckedItemPosition();
      if (pos >= 0) {
        textView.setText("Your selection: " + WINES[pos]);
      } else {
        textView.setText("No selection");
      }
    }
  });
}
```

The name of the selected wine

The Wine Selection App – Multiple Selections

- Another variation of the *Wine List* app
- Customized screen layout
  - Similar to the Single Selection app
- Multiple selection mode
- Handle selections
  - Display the selections in the *Text View* at the top



DePaul University

Wine Selection – Multiple – The List Activity

```
public class MainActivity extends ListActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    final ListView listView = getListView();
    listView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
    setListAdapter(new ArrayAdapter<String>(this,
      android.R.layout.simple_list_item_multiple_choice,
      WINES));
    …
  }
}
```

Set multiple selection mode

DePaul University

Wine Selection – Multiple – Display the Selected Items

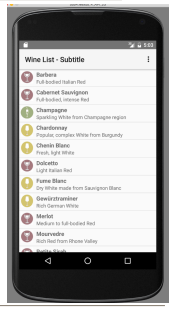
```
protected void onCreate(Bundle savedInstanceState) {
  …
  select.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view) {
      SparseBooleanArray selected = listView.getCheckedItemPositions();
      if (listView.getCheckedItemCount() > 0) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < selected.size(); i++) {
          if (selected.valueAt(i)) {
            if (sb.length() > 0) sb.append(", ");
            sb.append(WINES[selected.keyAt(i)]);
          }
        }
        textView.setText("Your selection: " + sb);
      } else { textView.setText("No selection"); }
    }
  });
}
```

Sparse Boolean Array:
{ 1 → *true*,
4 → *true*,
7 → *false*,
9 → *true* }

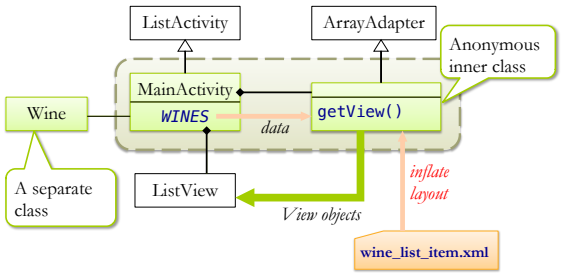
The Wine List Apps

The Wine List - Subtitle App

- Another variation of the *Wine List* app
- Customize the row layout in *List View*
  - Two lines of text and an icon
- Use a customized *Array Adapter*
  - *Array Adapter* handles a single string object as the content for each row in the list, by default.



The Wine List - Subtitle App – Architecture Overview



The Wine Class

```
public class Wine {
  enum Type { Red, White, Rosé, Sparkling }
  String name;
  Type type;
  String shortDescription;
  String longDescription;
  public Wine(String name, Type type, String
              shortDescription, String longDescription) {
    this.name = name;
    this.type = type;
    this.shortDescription = shortDescription;
    this.longDescription = longDescription;
  }
  public String toString() { return name; }
  …
}
```

The `toString()` method is required for *Array Adapter*.

Getter and setter for each field.

The Main Activity – The WINES Array

```
private static final Wine[] WINES = {
  new Wine("Barbera",
           Wine.Type.Red,
           "Full-bodied Italian Red",
           "Barbera is a red wine grape ..."),
  new Wine("Cabernet Sauvignon",
           Wine.Type.Red,
           "Full-bodied, intense Red",
           "Cabernet Sauvignon is a red wine ..."), … ,
  new Wine("Zinfandel",
           Wine.Type.Red,
           "Medium to full-bodied Red",
           "Zinfandel is a medium to full-bodied ..."),
};
```


Array Adapter with a View

- The basic use of an *Array Adapter*
  - Map each item's `toString()` value to a *Text View*
- It is possible to map any object to a *View Group*
  - The layout of the *View Group* is user defined
  - The *View Group* must include a *Text View*
  - You must extend *Array Adapter* and override the `getView()` method
    - Return an instance of the user defined *View Group*, for a specific row/position in the *List View*
    - Set the values to the *View* objects in the *View Group*

Array Adapter with a View

- Use the following constructor of *Array Adapter*

```
new ArrayAdapter<T>(context, resId, textId, array);
```

  - *T*: the type of the contents, i.e., data items
  - *context*: the app context, the hosting *Activity*
  - *resId*: the layout resource id of the user-defined *View Group* that contains a *Text View*
  - *textId*: the layout resource id of the *Text View* contained in the *View Group*
  - *array*: the data array, items are of type *T*

An extra parameter

Customize Row Layout – wine_list_itme.xml

```
<LinearLayout android:layout_width="match_parent"
              android:layout_height="wrap_content">
  <ImageView android:id="@+id/image" … />
  <LinearLayout android:orientation="vertical" … >
    <TextView android:id="@+id/text1"
              android:textSize="16sp"
              android:textStyle="bold" … />
    <TextView android:id="@+id/text2"
              android:textSize="14sp" … />
  </LinearLayout>
</LinearLayout>
```

Chardonnay
Popular, complex White from Burgundy

Place image files for icons in **res/drawable-***

Layout for a single row in *List View*

DePaul University

The Main Activity – Customize Array Adapter

```
public class MainActivity extends ListActivity {
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setListAdapter(
      new ArrayAdapter<Wine>(this,
        R.layout.wine_list_item, R.id.text1, WINES) {
        @Override
        public View getView(int position,
          View convertView, ViewGroup parent) { … }
      });
  }
}
```

An anonymous inner class that extends *Array Adapter*

DePaul University

The Main Activity – Customize Array Adapter

```
public class MainActivity extends ListActivity {
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setListAdapter(
      new ArrayAdapter<Wine>(this,
        R.layout.wine_list_item, R.id.text1, WINES) {
        @Override
        public View getView(int position,
          View convertView, ViewGroup parent) { … }
      });
  }
}
```

This method overrides the getView() method in *Array Adapter*

Will show the implementation later

DePaul University

The Main Activity – Customize Array Adapter

```
public class MainActivity extends ListActivity {
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setListAdapter(
      new ArrayAdapter<Wine>(this,
        R.layout.wine_list_item, R.id.text1, WINES) {
        @Override
        public View getView (int position,
          View convertView, ViewGroup parent) { … }
      });
  }
}
```

Calling the constructor of the superclass *Array Adapter*

DePaul University

The Main Activity – Customize Array Adapter

```
public class MainActivity extends ListActivity {
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setListAdapter(
      new ArrayAdapter<Wine>(this,
            R.layout.wine_list_item,
            R.id.text1,
            WINES)
        @Override
        public View getView (int posi
                             ViewGroup par
```

The item type

The customized row layout

The id of the *Text Field* in the row layout for the content: the toString() value of each item.

The data array. The toString() value of each item is used as the row content.

DePaul University

The Main Activity – Customize Array Adapter

```
setListAdapter(new ArrayAdapter<Wine>(this,
    R.layout.wine_list_item, R.id.text1, WINES) {
  @Override
  public View getView(int position, View convertView, ViewGroup parent) {
    View row = super.getView(position, convertView, parent);

    return row;
  }
});
```

3. Find the *View* objects in row

4. Set values to the *View* objects

Call the superclass getView() method.
1. Inflate the row layout.
2. Set the `toString()` value of the item to the specified *Text View*.

Return a *View* object for the row at *position*

The Main Activity – Customize Array Adapter

```
setListAdapter(new ArrayAdapter<Wine>(this,
    R.layout.wine_list_item, R.id.text1, WINES) {
  @Override
  public View getView(int position, View convertView, ViewGroup parent) {
    View row = super.getView(position, convertView, parent);
    ImageView icon = (ImageView) row.findViewById(R.id.image);
    TextView description = (TextView) row.findViewById(R.id.text2);

    return row;
  }
});
```

3. Find the *View* objects in row

4. Set values to the *View* objects

The Main Activity – Customize Array Adapter

```
setListAdapter(new ArrayAdapter<Wine>(this,
    R.layout.wine_list_item, R.id.text1, WINES) {
  @Override
  public View getView(int position, View convertView, ViewGroup parent) {
    View row = super.getView(position, convertView, parent);
    ImageView icon = (ImageView) row.findViewById(R.id.image);
    TextView description = (TextView) row.findViewById(R.id.text2);

    Wine wine = WINES[position];
    description.setText(wine.getShortDescription());
    switch (wine.getType()) {
    case White: icon.setImageResource(R.drawable.white); break;
    case Red: icon.setImageResource(R.drawable.red); break;
    case Rosé: icon.setImageResource(R.drawable.rose); break;
    case Sparkling: icon.setImageResource(R.drawable.sparkling); break;
    }

    return row;
  }
});
```

4. Set values to the *View* objects

The Main Activity – Customize Array Adapter

```
setListAdapter(new ArrayAdapter<Wine>(this,
    R.layout.wine_list_item, R.id.text1, WINES) {
  @Override
  public View getView(int position, View convertView, ViewGroup parent) {
    View row = super.getView(position, convertView, parent);
    ImageView icon = (ImageView) row.findViewById(R.id.image);
    TextView description = (TextView) row.findViewById(R.id.text2);
    Wine wine = WINES[position];
    description.setText(wine.getShortDescription());
    switch (wine.getType()) {
    case White: icon.setImageResource(R.drawable.white); break;
    case Red: icon.setImageResource(R.drawable.red); break;
    case Rosé: icon.setImageResource(R.drawable.rose); break;
    case Sparkling: icon.setImageResource(R.drawable.sparkling); break;
    }
    return row;
  }
});
```

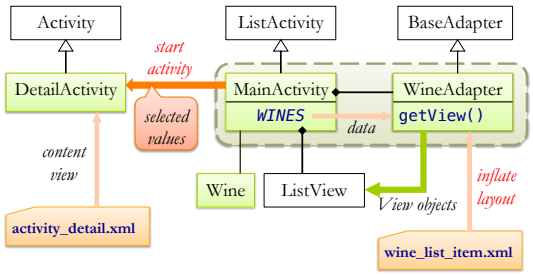
The Wine List – Details App

The Wine List – Details App

- An enhanced version of the *Wine List* app
  - The master-detail pattern
  - The master view is a *List View*
  - Touch an item in the master view brings up a detailed view for the item selected
- Passing data from the master list to the detail view
- Craft a more efficient list adapter

DePaul University

Wine List – Details – Architecture Overview



DePaul University

The Wine Class – A New Static Method

```
public class Wine {
  enum Type { Red, White, Rosé, Sparkling }
  String name;
  Type type;
  String shortDescription;
  String longDescription;
  …
  public static int getIconResource(Wine.Type type) {
    switch (type) {
      case White: return R.drawable.white;
      case Red:   return R.drawable.red;
      case Rosé: return R.drawable.rose;
      case Sparkling: return R.drawable.sparkling;
    }
    return -1;
  }
}
```

A static method to get the resource id of the icon for each wine type.

Extend the Base Adapter

- *Base Adapter* is the superclass of *Array Adapter*
  - An abstract class. Nearly complete
  - Easily extensible for any type of data and *View* objects
- Methods must be overridden

`int getCount()`	Number of items in the list
`Object getItem(int i)`	The *i*-th item
`long getItemId(int i)`	The unique id of the *i*-th item
`View getView(int i, View convertView, ViewGroup parent)`	The View object for the *i*-th item

DePaul University

Layout Inflation

- Convert an XML layout resource into a hierarchy of *View* objects
  - For performance reason, XML resources are preprocessed at build time
  - Not possible to inflate layout from XML loaded at run time
- Layout inflation is carried out by a *Layout Inflator*
  - Never directly created
  - Retrieve an instance as a system service

```
LayoutInflater inflater = (LayoutInflater)
  context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
```

DePaul University

Layout Inflation

- Inflation methods

```
inflate(int resource, ViewGroup root)
inflate(int resource, ViewGroup root,
        boolean attachToRoot)
```

`attachToRoot: true`

- `resource`: the resource id of the layout to be inflated
- `root`: the root container (parent) to attach the inflated *View* hierarchy
- `attachToRoot`: whether the inflated View hierarchy should be attached to the root container
  - Even when `attachToRoot` is false, `root` should not be null

DePaul University

The Wine Adapter – An Inner Class of the Main Activity

```
class WineAdapter extends BaseAdapter {
  private LayoutInflater inflater;

  @Override
  public int getCount() { return WINES.length; }
  @Override
  public Object getItem(int i) { return WINES[i]; }
  @Override
  public long getItemId(int i) { return i; }
  @Override
  public View getView(int position,
    View convertView, ViewGroup parent) { … }
}
```

Used in getView()

Data are in *WINES* array.

# of items

The *i*-th item

The unique id of the *i*-th item

Next slide

DePaul University

The Wine Adapter – The getView() Method

```
public View getView(int position,
                    View convertView, ViewGroup parent) {

}
```

1. Inflate the row layout
2. Find the *View* objects in row
3. Set values to the *View* objects

DePaul University

The Wine Adapter – The getView() Method

```
public View getView(int position,
                    View convertView, ViewGroup parent) {
 if (inflater == null) inflater = (LayoutInflater)
   MainActivity.this.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
 View row = inflater.inflate(R.layout.wine_list_item, parent, false);

}
```

1. Inflate the row layout
2. Find the *View* objects in row
3. Set values to the *View* objects

DEPAUL UNIVERSITY 124

The Wine Adapter – The getView() Method

```
public View getView(int position,
                    View convertView, ViewGroup parent) {
  if (inflater == null) inflater = (LayoutInflater)
    MainActivity.this.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
  View row = inflater.inflate(R.layout.wine_list_item, parent, false);

  ImageView icon = (ImageView) row.findViewById(R.id.image);
  TextView name = (TextView) row.findViewById(R.id.text1);
  TextView description = (TextView) row.findViewById(R.id.text2);

}
```

2. Find the *View* objects in row
3. Set values to the *View* objects

DEPAUL UNIVERSITY

The Wine Adapter – The getView() Method

```
public View getView(int position,
                    View convertView, ViewGroup parent) {
  if (inflater == null) inflater = (LayoutInflater)
    MainActivity.this.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
  View row = inflater.inflate(R.layout.wine_list_item, parent, false);

  ImageView icon = (ImageView) row.findViewById(R.id.image);
  TextView name = (TextView) row.findViewById(R.id.text1);
  TextView description = (TextView) row.findViewById(R.id.text2);

  Wine wine = WINES[position];
  name.setText(wine.getName());
  description.setText(wine.getShortDescription());
  icon.setImageResource(Wine.getIconResource(wine.getType()));

  return row;
}
```

3. Set values to the *View* objects

DEPAUL UNIVERSITY

The Wine Adapter – Can We Improve This?

```
public View getView(int position,
                    View convertView, ViewGroup parent) {
 if (inflater == null) inflater = (LayoutInflater)
   MainActivity.this.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
 View row = inflater.inflate(R.layout.wine_list_item, parent, false);

 …
 return row;
}
```

1. Inflate the row layout
2. Find the *View* objects in row
3. Set values to the *View* objects

DEPAUL UNIVERSITY

Recycle View Objects in Adapter

```
View getView(int i, View convertView,
             ViewGroup parent)
```

- The `convertView` parameter
  - A recycled *View* object available for reuse, if it is not null
  - Need to ensure it is possible to convert the *View* object to display the correct data.
  - Heterogeneous lists can specify the view type for each row, so that this *View* is always of the right type
    - Use `getItemViewType(i)`

DEPAUL UNIVERSITY

A More Efficient Approach – Recycle View Objects

```
public View getView(int position,
                    View convertView, ViewGroup parent) {
 View row = convertView;
 if (convertView == null) {
   if (inflater == null) inflater = (LayoutInflater)
       MainActivity.this.getSystemService(
       Context.LAYOUT_INFLATER_SERVICE);
   row = inflater.inflate(R.layout.wine_list_item, parent, false);
 }

 return row;
}
```

1. Inflate the row layout, or reuse
2. Find the *View* objects in row
3. Set values to the *View* objects

DEPAUL UNIVERSITY

The Master Activity

```
public class MainActivity extends ListActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setListAdapter(new WineAdapter());
  }
  @Override
  protected void onListItemClick(ListView l,
                    View v, int position, long id) { … }
  class WineAdapter extends BaseAdapter { … }
  private static final Wine[] WINES = { … }
}
```

Next slide

DePaul University

Passing Data to the Detail View

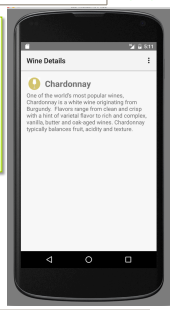
```
protected void onListItemClick(ListView l,
                View v, int position, long id) {
  Intent intent =
      new Intent(MainActivity.this, DetailActivity.class);
  intent.putExtra("WineName",
         WINES[position].getName());
  intent.putExtra("WineDescription",
         WINES[position].getLongDescription());
  intent.putExtra("WineIcon",
         Wine.getIconResource(WINES[position].getType()));
  startActivity(intent);
}
```

Can we pass the Wine object?

DePaul University

The Detail Activity – The Layout

```
<LinearLayout … >
  <LinearLayout … >
    <ImageView … android:id="@+id/image" />
    <TextView … android:id="@+id/text1" />
  </LinearLayout>
  <TextView … android:id="@+id/text2" />
</LinearLayout>
```



DePaul University

The Detail Activity

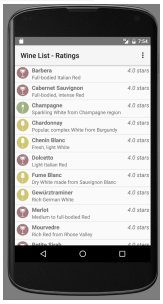
```
public class DetailActivity extends Activity {
  protected void onStart() {
    super.onStart();
    Intent intent = getIntent();
    if (intent != null) {
      TextView name = (TextView) findViewById(R.id.text1);
      TextView description = (TextView) findViewById(R.id.text2);
      ImageView icon = (ImageView) findViewById(R.id.image);
      name.setText(intent.getCharSequenceExtra("WineName"));
      description.setText(
         intent.getCharSequenceExtra("WineDescription"));
      icon.setImageResource(intent.getIntExtra("WineIcon", -1));
    }
  }
  …
}
```

DePaul University

The Wine List – Ratings App

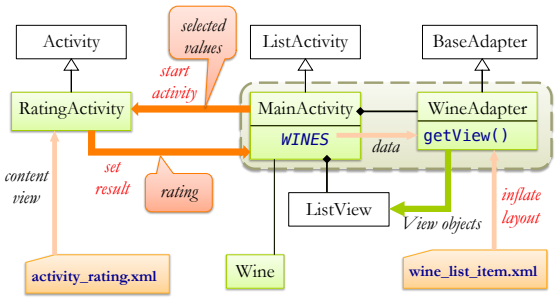
The Wine List – Ratings App

- Another enhancement of the *Wine List* App
- A rating is associated with each wine
  - The master list includes ratings
- A *Rating Bar* is added to the detail view
- A rating changes are reflected in the master list, when user navigate back to the master list.



DePaul University

The Wine List – Ratings App – Architecture Overview



DePaul University

The Wine Class – A New Field: rating

```
public class Wine {
   enum Type { Red, White, Rosé, Sparkling }
   String name;
   Type type;
   String shortDescription;
   String longDescription;
   float rating = 4.0f;

   public Wine(String name, Type type,
            String shortDescription, String longDescription) { … }
   …
  public String toString() { return name; }
  public static int getIconResource(Wine.Type type) { … }
}
```

Default rating is set to 4 stars

Getter and setter for each field.

DePaul University

Updated Row Layout

```
<LinearLayout … >
  <ImageView android:id="@+id/image" … />
  <LinearLayout android:orientation="vertical" … >
    <LinearLayout … >
      <TextView … android:id="@+id/text1" />
      <Space … android:layout_weight="1" />
    <TextView … android:id="@+id/text3" />
    </LinearLayout>
    <TextView … android:id="@+id/text2" />
  </LinearLayout>
</LinearLayout>
```

The rating displayed on the master list

Chardonnay
Popular, complex White from Burgundy
4.0 stars

DePaul University

A More Efficient Wine Adapter – Cache Bitmaps

```
static class WineAdapter extends BaseAdapter {
  private LayoutInflater inflater;
  private Map<Wine.Type, Bitmap> icons;
  WineAdapter(Context context) {
    inflater = (LayoutInflater)
      context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    icons = new HashMap<Wine.Type, Bitmap>();
    icons.put(Wine.Type.Red,
      BitmapFactory.decodeResource(context.getResources(),
                                   R.drawable.red));
    icons.put(Wine.Type.Rosé, … );
    icons.put(Wine.Type.Sparkling, … );
    icons.put(Wine.Type.White, … );
  }
  …
```

A static inner class

The drawable resources are loaded once only and cached as bitmaps

DePaul University

A More Efficient Wine Adapter – Cache Bitmaps (Alt)

```
static class WineAdapter extends BaseAdapter {
  private LayoutInflater inflater;
  private Map<Wine.Type, Bitmap> icons;
  WineAdapter(Context context) {
    inflater = (LayoutInflater)
      context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    icons = new HashMap<Wine.Type, Bitmap>();
    for (Wine.Type type : Wine.Type.values()) {
      icons.put(type,
        BitmapFactory.decodeResource(
            context.getResources(),
            Wine.getIconResource(type)));
    }
  }
  …
```

Iterate over each value of the **enum** type

DePaul University

The Wine Adapter – Can We Improve This?

```
public View getView(int position,
                    View convertView, ViewGroup parent) {
  View row = convertView;
  if (row == null) {
    row = inflater.inflate(R.layout.wine_list_item, parent, false);
  }
  ImageView icon = (ImageView) row.findViewById(R.id.image);
  TextView name = (TextView) row.findViewById(R.id.text1);
  TextView description = (TextView) row.findViewById(R.id.text2);

}
```

1. Inflate the row layout, or reuse

2. Find the *View* objects in row, even when the *View* is reused

3. Set values to the *View* objects

DePaul University

A More Efficient Wine Adapter – The Holder Pattern

```
static class WineAdapter extends BaseAdapter {
  private LayoutInflater inflater;
  private Map<Wine.Type, Bitmap> icons;
  WineAdapter(Context context) { … }
  @Override public int getCount() { return WINES.length; }
  @Override public Object getItem(int i) { return WINES[i]; }
  @Override public long getItemId(int i) { return i; }
  @Override public View getView(int position,
              View convertView, ViewGroup parent) { … }
  static class ViewHolder {
    TextView name;
    TextView description;
    TextView rating;
    ImageView icon;
  }
}
```

Store references to *View* objects inside each row container

A More Efficient Wine Adapter – The Holder Pattern

```
public View getView(int position, View convertView,
                    ViewGroup parent) {
  ViewHolder holder;
  View row = convertView;
  if (row == null) {
    row = inflater.inflate(R.layout.wine_list_item, parent, false);
    holder = new ViewHolder();
    holder.icon = (ImageView) row.findViewById(R.id.image);
    holder.name = (TextView) row.findViewById(R.id.text1);
    holder.description = (TextView)
    row.findViewById(R.id.text2);
    holder.rating = (TextView) row.findViewById(R.id.text3);
    row.setTag(holder);
  } else {
    holder = (ViewHolder) row.getTag();
  }
  …
```

Use the tag of *View* objects

A More Efficient Wine Adapter – The Holder Pattern

```
public View getView(int position, View convertView,
                    ViewGroup parent) {
  ViewHolder holder;
  View row = convertView;
  …
  Wine wine = WINES[position];
  holder.name.setText(wine.getName());
  holder.description.setText(wine.getShortDescription());
  holder.icon.setImageBitmap(icons.get(wine.getType()));
  holder.rating.setText(wine.getRating() + " stars");
  return row;
}
```

Use bitmap rather than resource

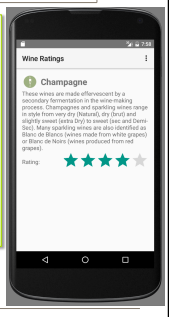
Set the rating

Passing Data to Rating Activity

```
private Wine selectedWine;
protected void onListItemClick(ListView l,
                View v, int position, long id) {
  selectedWine = WINES[position];
  Intent intent = new Intent(MainActivity.this,
                             RatingActivity.class);
  intent.putExtra("WineName", selectedWine.getName());
  intent.putExtra("WineDescription",
                  selectedWine.getLongDescription());
  intent.putExtra("WineIcon",
          Wine.getIconResource(selectedWine.getType()));
  intent.putExtra("WineRating", selectedWine.getRating());
  startActivityForResult(intent, RATING);
}
```


The Rating Activity – The Layout

```
<LinearLayout … >
  <LinearLayout … >
    <ImageView … android:id="@+id/image" />
    <TextView … android:id="@+id/text1" />
  </LinearLayout>
  <TextView … android:id="@+id/text2" />
  <LinearLayout … >
    <TextView … android:text="Rating:" />
    <RatingBar … android:id="@+id/rating" />
  </LinearLayout>
</LinearLayout>
```



The Rating Activity

```
public class RatingActivity extends Activity {
  @Override protected void onStart() {
    super.onStart();
    Intent intent = getIntent();
    if (intent != null) {
      TextView name = (TextView) findViewById(R.id.text1);
      TextView description = (TextView) findViewById(R.id.text2);
      ImageView icon = (ImageView) findViewById(R.id.image);
      RatingBar rating = (RatingBar) findViewById(R.id.rating);
      name.setText(intent.getCharSequenceExtra("WineName"));
      description.setText(
        intent.getCharSequenceExtra("WineDescription"));
      icon.setImageResource(intent.getIntExtra("WineIcon", -1));
      rating.setRating(intent.getFloatExtra("WineRating", 4));
    }
  }
  …
```

Return the Rating Result – Option A: the First Try

```
public class RatingActivity extends Activity
     implements RatingBar.OnRatingBarChangeListener {
  …
  @Override protected void onStart() {
    …
    rating.setOnRatingBarChangeListener(this);
  }

  @Override public void onRatingChanged(RatingBar ratingBar,
                                        float v, boolean b) {
    Intent ratingResult = new Intent();
    ratingResult.putExtra("WineRating", ratingBar.getRating());
    setResult(RESULT_OK, ratingResult);
  }
}
```

DePaul University

Return the Rating Result – Option B: the Better Option

```
public class RatingActivity extends Activity {
  …
  @Override
  public void finish() {
    Intent ratingResult = new Intent();
    RatingBar ratingBar =
      (RatingBar) findViewById(R.id.rating);
    ratingResult.putExtra("WineRating",
                          ratingBar.getRating());
    setResult(RESULT_OK, ratingResult);
    super.finish();
  }
}
```

DePaul University

Refresh List View

- If the underlying data of a list view is changed, you should call

```
adapter.notifyDataSetChanged();
```

- This will notify the list view that the data has been changed, and it should refresh itself.

DePaul University

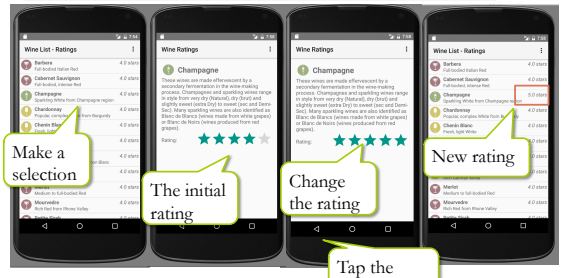
Back to the Master List – Update the Rating

```
public class MainActivity extends ListActivity {
  private Wine selectedWine;
  @Override protected void onListItemClick( … ) {
    selectedWine = WINES[position];
    …
    startActivityForResult(intent, RATING);
  }
  @Override protected void onActivityResult(int requestCode,
                                            int resultCode, Intent data) {
    if (requestCode == RATING) {
      if (resultCode == RESULT_OK && data != null) {
        selectedWine.setRating(data.getFloatExtra("WineRating", 4));
        ((WineAdapter) getListAdapter()).notifyDataSetChanged();
      }
    }
  }
  …
}
```

Selected wine was set before starting activity

1. Update the selected wine
2. Refresh the list view

The Wine List – Rating App



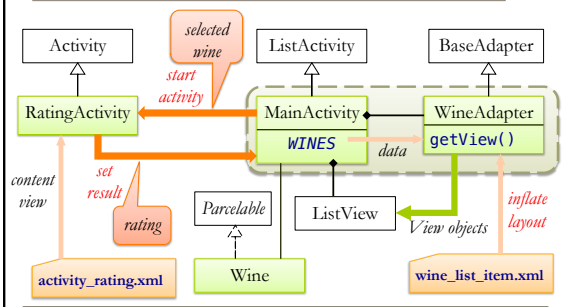
DePaul University

The Wine List – Parcel App

The Wine List – Parcel App

- One more enhancement to the *Wine List – Rating* app
  - Identical function and behavior
- Passing a *Wine* object to the detail view
- The *Wine* class need to be made *Parcelable*

The Wine List – Parcel – Architecture



The Parcel Class

- Container for messages and data to be passed among different components or apps
- Data are flattened. Must be *Parcelable*.
- Not a general-purpose serialization mechanism.
  - Designed for high performance.
  - Compact in size. Minimum redundancy.
  - Less fault-tolerant. Maybe unreadable after changes.
- Methods
  - Read/write data of common types

Implement Parcelable

- Implement the methods

```
int describeContents()
void writeToParcel(Parcel out, int flags)
```

- Provide a static field CREATOR, which implements Parcelable.Creator
  - Methods of

```
MyParcelable createFromParcel(Parcel in)
MyParcelable[] newArray(int size)
```


The Parcelable Wine Class

```
public class Wine implements Parcelable {
  enum Type {Red, White, Rosé, Sparkling}
  String name;
  Type type;
  String shortDescription;
  String longDescription;
  float rating = 4.0f;
  public Wine(String name, Type type,
    String shortDescription, String longDescription) { … }
  public String toString() { return name; }
```

Getter and setter for each field.

Implementation of Parcelable

Next 3 slides

```
}
```


The Parcelable Wine Class – Implement writeToParcel()

```
public class Wine implements Parcelable {
  …
  @Override
  public int describeContents() { return 0; }
  @Override
  public void writeToParcel(Parcel out, int flags) {
    out.writeString(name);
    out.writeInt(type.ordinal());
    out.writeString(shortDescription);
    out.writeString(longDescription);
    out.writeFloat(rating);
  }
  …
}
```

Flags for "special contents"

Write each field to the *Parcel*

The Parcelable Wine Class – Implement Creator

```
public class Wine implements Parcelable   {
  …
  public static final Parcelable.Creator<Wine> CREATOR
       = new Parcelable.Creator<Wine>() {
    public Wine createFromParcel(Parcel in) {
      return new Wine(in);
    }
    public Wine[] newArray(int size) {
      return new Wine[size];
    }
  };
  …
}
```

Create a *Wine* object. Delegate to a new *Wine* constructor

Create a *Wine* array

The Parcelable Wine Class – A New Private Constructor

```
public class Wine implements Parcelable   {
  …
  private Wine(Parcel in) {
    name = in.readString();
    type = Type.values()[in.readInt()];
    shortDescription = in.readString();
    longDescription = in.readString();
    rating = in.readFloat();
  }
}
```

Read each field from the *Parcel*, in the same order they were written.

Passing Data to Rating Activity – Wine is Not Parcelable

```
private Wine selectedWine;
protected void onListItemClick(ListView l,
                View v, int position, long id) {
  selectedWine = WINES[position];
  Intent intent = new Intent(MainActivity.this,
                             RatingActivity.class);
  intent.putExtra("WineName", selectedWine.getName());
  intent.putExtra("WineDescription",
                  selectedWine.getLongDescription());
  intent.putExtra("WineIcon",
          Wine.getIconResource(selectedWine.getType()));
  intent.putExtra("WineRating", selectedWine.getRating());
  startActivityForResult(intent, RATING);
}
```


Passing Data to Rating Activity – Wine is Parcelable

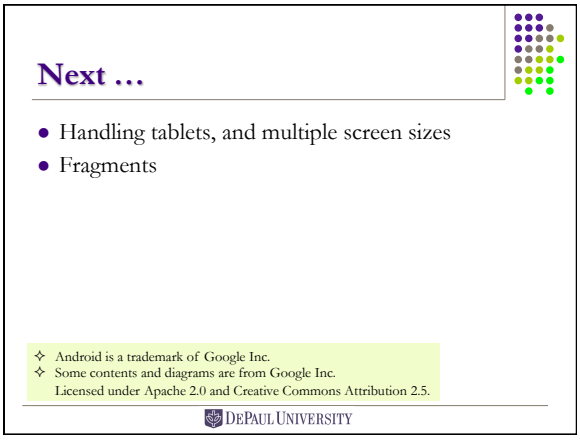
```
private Wine selectedWine;
protected void onListItemClick(ListView l,
                View v, int position, long id) {
  selectedWine = WINES[position];
  Intent intent = new Intent(MainActivity.this,
                             RatingActivity.class);
  intent.putExtra("Wine", selectedWine);




  startActivityForResult(intent, RATING);
}
```


The Rating Activity – Retrieve Unflattened Wine Objects

```
public class RatingActivity extends Activity {
  @Override protected void onStart() {
    super.onStart();
    Intent intent = getIntent();
    if (intent != null) {
      TextView name = (TextView) findViewById(R.id.text1);
      TextView description = (TextView) findViewById(R.id.text2);
      ImageView icon = (ImageView) findViewById(R.id.image);
      RatingBar rating = (RatingBar) findViewById(R.id.rating);
      Wine wine = intent.getParcelableExtra("Wine");
      name.setText(wine.getName());
      description.setText(wine.getLongDescription());
      icon.setImageResource(Wine.getIconResource(wine.getType()));
      rating.setRating(wine.getRating());
    }
  }
  …
```


The Sample Code

- The sample apps in this lecture are available in D2L
  - OrderMySub.zip
  - WineList.zip
  - WineSelection-Single.zip
  - WineSelection-Multi.zip
  - WineList.zip
  - WineList-Details.zip
  - WineList-Ratings.zip
  - WineList-Parcel.zip


Next …
Handling tablets, and multiple screen sizes
Fragments
Android is a trademark of Google Inc.
Some contents and diagrams are from Google Inc.
Licensed under Apache 2.0 and Creative Commons Attribution 2.5.
DePaul University