JÖNKÖPING UNIVERSITY

School of Engineering

ANDROID LISTVIEW

Peter Larsson-Green

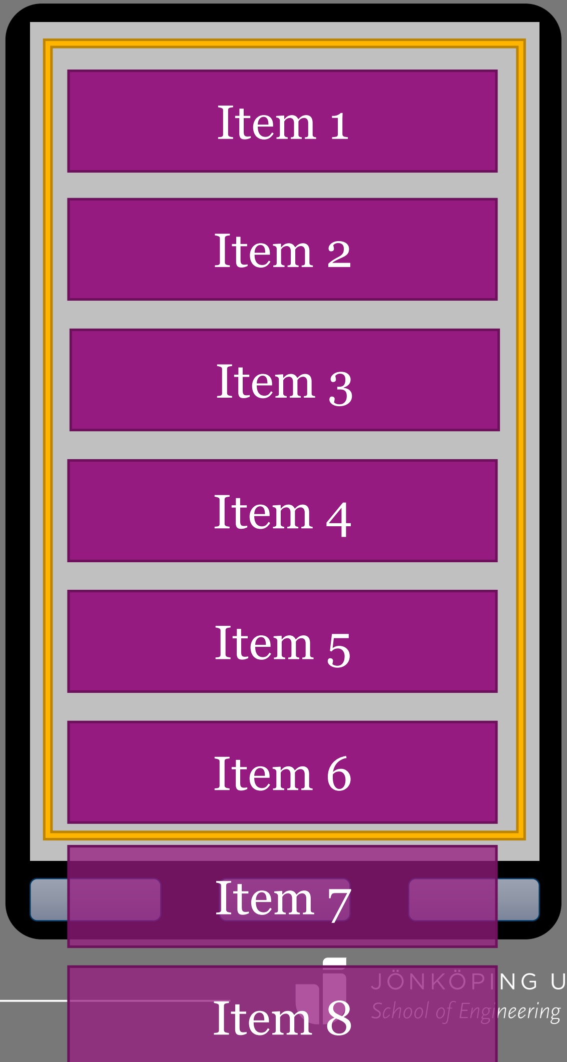
Jönköping University

Spring 2020

VERTICAL LISTS OF VIEWS

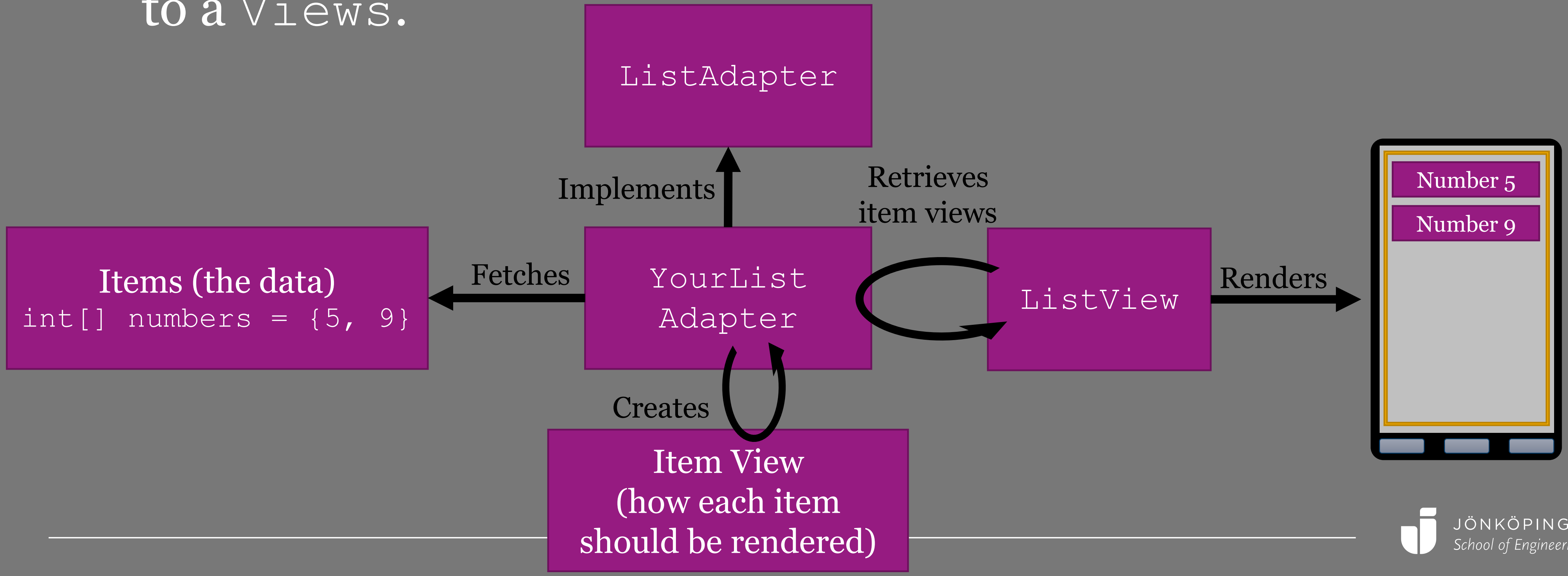
Showing a vertical list of items.

- `ScrollView` + `LinearLayout`?
  - 1000 items → 1000 item views created.
  - Only ~7 shown on the screen at the same time → a lot of memory wasted.
- `ListView` to the rescue.
  - Will only create ~7 item views.
  - When an item view scrolls out at the top, it will be inserted at the bottom for another item (the item view is re-used for another item).



USING `ListView`

- The class `ListView` is responsible for rendering the list.
- The interface `ListAdapter` is used by `ListView` to map items to a `Views`.



USING `ListView`

- The class `ListView` is responsible for rendering the list.
- The interface `ListAdapter` is used by `ListView` to map items to a `Views`.
- Additional things good to know:
  - `ListView` can be used to select one or multiple items.
  - Different item views can be used for different items.
  - Individual items can be marked as disabled (not selectable).

USING ListView

```java
public class MainActivity extends AppCompatActivity {
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ListView listView = (ListView) findViewById(R.id.list_view);
    listView.setAdapter(new MyListAdapter());
  }
}
```

layout/activity_main.xml

```xml
<ListView
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:id="@+id/list_view">
</ListView>
```

USING `ListView`

```
public class MyListAdapter implements ListAdapter {
  private int[] numbers = {5, 9, 4, 1, 6};

  public boolean areAllItemsEnabled() {
    return true;
  }

}
```

- Return `false` if some items should not be selectable, e.g.:
  - If it should only be possible to pick even numbers.

USING `ListView`

```java
public class MyListAdapter implements ListAdapter {
  private int[] numbers = {5, 9, 4, 1, 6};

  public boolean isEnabled(int position) {
    return true;
  }

}
```

- If `areAllItemsEnabled` returned `false`,
  this will be used to figure out which items that are selectable.

USING `ListView`

```
public class MyListAdapter implements ListAdapter {
  private int[] numbers = {5, 9, 4, 1, 6};

  public int getCount() {
    return numbers.length;
  }

}
```

- Should return the total number of items.

USING `ListView`

```java
public class MyListAdapter implements ListAdapter {
  private int[] numbers = {5, 9, 4, 1, 6};

  public boolean isEmpty() {
    return getCount() == 0;
  }

}
```

- Should return `true` if there are no items, otherwise `false`.

USING ListView

```
public class MyListAdapter implements ListAdapter {
  private int[] numbers = {5, 9, 4, 1, 6};

  public Object getItem(int position) {
    return numbers[position];
  }


}
```

USING `ListView`

```java
public class MyListAdapter implements ListAdapter {
  private int[] numbers = {5, 9, 4, 1, 6};

  public long getItemId(int position) {
    return position;
  }

}
```

USING `ListView`

```
public class MyListAdapter implements ListAdapter {
  private int[] numbers = {5, 9, 4, 1, 6};

  public boolean hasStableIds() {
    return true;
  }

}
```

USING ListView

```
public class MyListAdapter implements ListAdapter {
  private int[] numbers = {5, 9, 4, 1, 6};

  public int getViewTypeCount() {
    return 1;
  }


}
```

USING `ListView`

```
public class MyListAdapter implements ListAdapter {
  private int[] numbers = {5, 9, 4, 1, 6};

  public int getItemViewType(int position) {
    return 0;
  }

}
```

USING ListView

```java
public class MyListAdapter implements ListAdapter {
  private int[] numbers = {5, 9, 4, 1, 6};
  public View getView(int position, View convertView, ViewGroup parent){
    LayoutInflater inflater = LayoutInflater.from(aContext);
    View rootView = inflater.inflate(R.layout.item, parent, false);
    TextView textView = (TextView) rootView.findViewById(R.id.textView);
    textView.setText("Number "+numbers[position]);
    return rootView;
  }
}
```

We should reuse this one if it's not null!

layout/item.xml

```xml
<TextView
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/textView"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"/>
```

USING `ListView`

```java
public class MyListAdapter implements ListAdapter {
  private int[] numbers = {5, 9, 4, 1, 6};
  public View getView(int position, View convertView, ViewGroup parent){
    if(convertView == null){
      LayoutInflater inflater = LayoutInflater.from(aContext);
      convertView = inflater.inflate(R.layout.item, parent, false);
    }
    TextView textView = (TextView) convertView.findViewById(R.id.textView);
    textView.setText("Number "+numbers[position]);
    return convertView;
  }
}
```

Is costly when the item layout contains many views. Should be cached!

USING ListView

```java
public class ViewHolder{
  public TextView textView;
}
```

```java
public class MyListAdapter implements ListAdapter {
  private int[] numbers = {5, 9, 4, 1, 6};
  public View getView(int position, View convertView, ViewGroup parent){
    if(convertView == null){
      LayoutInflater inflater = LayoutInflater.from(aContext);
      convertView = inflater.inflate(R.layout.item, parent, false);
      ViewHolder viewHolder = new ViewHolder();
      viewHolder.textView = (TextView) convertView.findViewById(R.id.textView);
      convertView.setTag(viewHolder);
    }
    ((ViewHolder)convertView.getTag()).textView.setText("Number "+numbers[position]);
    return convertView;
  }
}
```

USING `ListView`

Android comes with some pre-defined list adapters we can use:

- `ArrayAdapter<T>` for arrays and `List`s.
- `CursorAdapter` for cursors (e.g. when reading from database).