학생 여러분 반갑습니다.

다른 친구들이 입장할 때까지 조금 기다려 주십시오.

곧 모바일 프로그래밍 수업을 시작합니다.

음소거( )가 되었는지 확인 바랍니다.

모바일 프로그래밍
화목(1,2교시)/ 화목(3,4교시)
정윤현 (AI/소프트웨어학부)

Activity ⇒ 위젯들
Service

Mobile Programming

Android Programming

Chap 5-1. Widgets & Event (basic)

Prof. Younhyun Jung

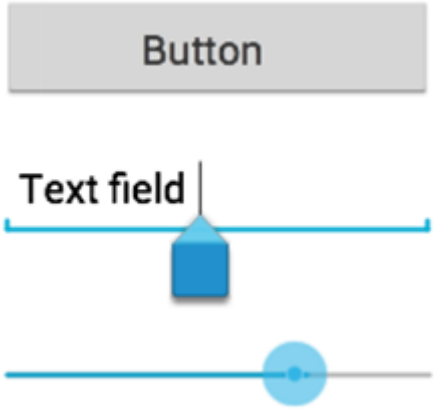
Email) younhyun.jung@gachon.ac.kr

가천대학교 소프트웨어학과
Gachon University

View (par) → 상속

Widgets

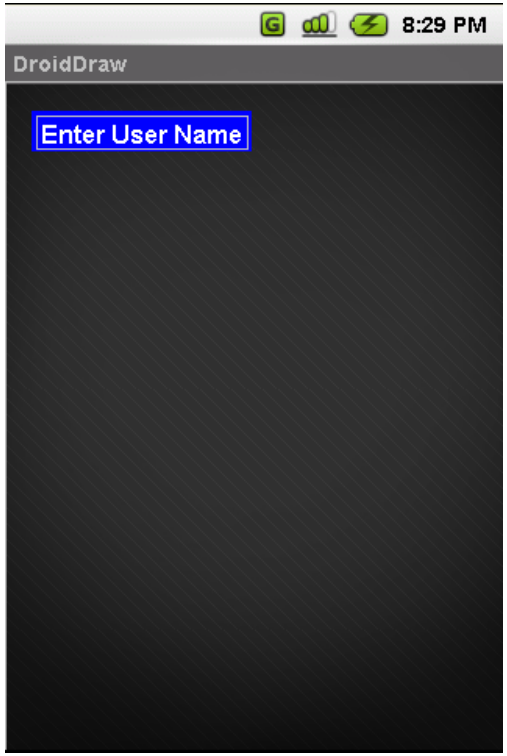
(child)



- **Label**
  - TextView
- **Input Controls**
  - Button
    - Represents a push-button widget.
  - EditText
    - A subclass of the TextView that allows users to edit its text content
  - ImageButton
    - Similar to the Button view, except that it also displays an image.
  - CheckBox
    - A special type of button that has two states: checked or unchecked.
  - More…
    - RadioButton, ToggleButton, etc.

Widgets - TextView

- A label is called in android a **TextView**. Label
- TextViews are typically used to display a caption.
- TextViews are *not* editable, therefore they take no input.

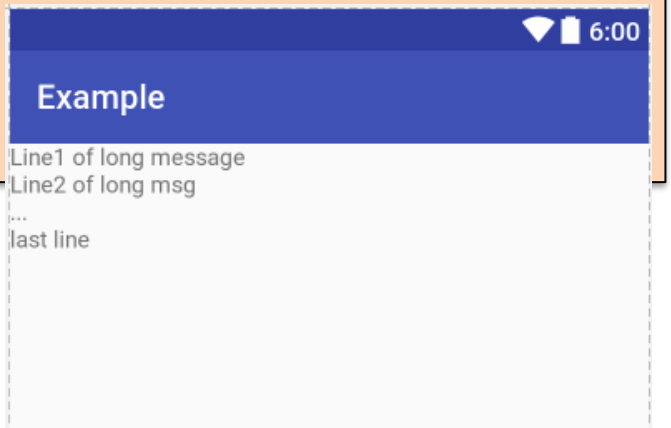


Widgets - TextView

- Example

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical" >
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/hello_there" />
</LinearLayout>
```

String 변수 설정할 수 없음
→ 유지보수 ??



Good Programming Style:

Add to the res/values/string.xml the entry

```
<string name="hello_there">Line1 of long message\nLine2 of long msg\n...\nlast line</string>
```

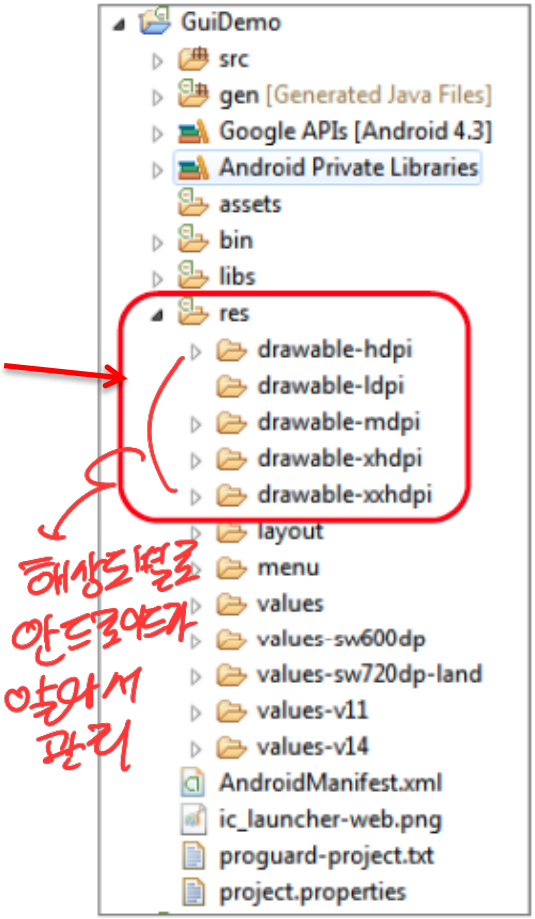
Widgets: Images

- **ImageView** and **ImageButton** are two Android widgets that allow embedding of images in your applications.
  - ImageView : ImageButton = TextView : Button
- Two attributes to specify what picture to use :
  - **android:src** 앞
  - **android:background** 뒤

  그 뒤에서 구분.
- Pictures usually reference a drawable resource.
  - @drawable/ID
- In a Java code, you can also set the image via setImageURI() method.
- **ImageButton**, is a subclass of ImageView. It adds the standard Button behavior for responding to clickevents.

  이미지 + 버튼

  - Supported file types: PNG (preferred), JPG (acceptable), and GIF (discoraged) for icons, logos, or other graphics



해상도별로 안드로이드가 알아서 관리

Cont.

```
<ImageButton
        android:id="@+id/myImageBtn1"
        android:src="@drawable/ic_launcher"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
>
</ImageButton>
<ImageView
        android:id="@+id/myImageView1"
        android:src="@drawable/flower1"
        android:layout_width="500px"
        android:layout_height="300px"
        android:scaleType="fitXY"
>
</ImageView>
```

ImageView크기에 image 사이즈를 맞춤

이미지를 맞추는 방법은

Widgets: Buttons

- A **Button** widget allows the simulation of a clicking action on a GUI.
- Button is a subclass of TextView. Therefore formatting a Button's face is similar to the way to set a TextView.

TextView + click event
Button

```
...
<Button
android:id="@+id/btnExitApp"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/buttonExit_caption"
android:textSize="16sp"
android:textStyle="bold"
>
</Button>
```

Widgets: ImageButtons

ImageView에
Button 아님!?
SetText() X

- **Combining Images & Text!**
- A **Button** widget can consist of **text and/or an icon**.
  - With text and an icon, using the Button class with the android:drawableLeft attribute:

```
<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:drawableLeft="@drawable/button_icon"
   ... />
```

setImageURI()


```
<ImageButton
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:src="@drawable/button_icon"
   ... />
```

Buttons - Combining Images & Text

- A common **Button** widget could display text and a simple image as shown below


```
<LinearLayout
. . .
    <Button
        android:layout_width= "wrap_content"
        android:layout_height= "wrap_content"
        android:drawableLeft= "@drawable/ic_launcher"
        android:gravity= "left|center_vertical"
        android:padding= "15dp"
        android:text= "Click me" />
</LinearLayout>
```

Interact with Your UI components By Attaching Listeners

- To Capture the events from the specific View object that the user interacts with.
  - The View class provides the means to do so.
- For various UI events, View classes notice several public callback methods.
- Example:
  - When a View (such as a Button) is clicked, the onClick () method is called on that object.
  - You can intercept this by extending the class and overriding the method!!
  - You can do this by attaching new listeners to the Widgets and handle your events !!

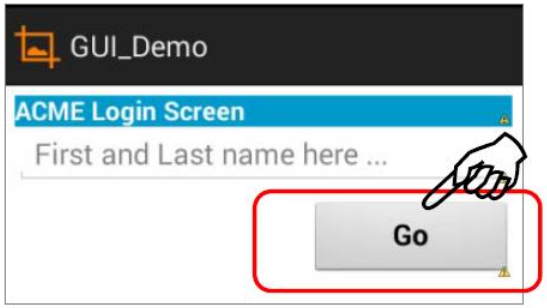
UI 상호작용
↔ Listener?

→register (override) a new "**Event Listener**" to the View
: use setOnXXXXListener() method of View class

Skip

Cont. : Attaching Listeners

- **Attaching Listeners to the Widgets**
  - The button of our example could now be used, for instance a listener for the click event could be written as:




```
   Button button = (Button)findViewById(R.id.myButton);
   button.setOnClickListener( new OnClickListener() {
        @Override
        public void onClick(View v) {
                    updateTime();
        }
    } );
// …
    private void updateTime() {
                   button.setText(new Date().toString());
    }
```

Note: Other common 'listeners' watch for events such as: textChanged, tap, long-press, select, focus, etc

Event Listener

- **An event listener**
  - An **interface** in the View class that contains a single callback method.
  - called by the Android framework when the respective action occurs on that object, i.e., the listener that has been registered is triggered by user interaction with the item in the UI.
- Example (callback) methods:
  - OnClickListener : onClick()
    - called when the user either touches the item
  - OnLongClickListener : onLongClick()
    - called when the user either touches and holds the item
  - OnTouchListener : onTouch()
  - OnFocusChangeListener: onFocusChange()
  - onKey() …

Exercise

- **Create a new Project**
- In **activity_main.xml**
  - **Add the** following element **for making a button**

```
<Button
    android:id="@+id/myButton"
    android:text="Press Me"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
/>
```

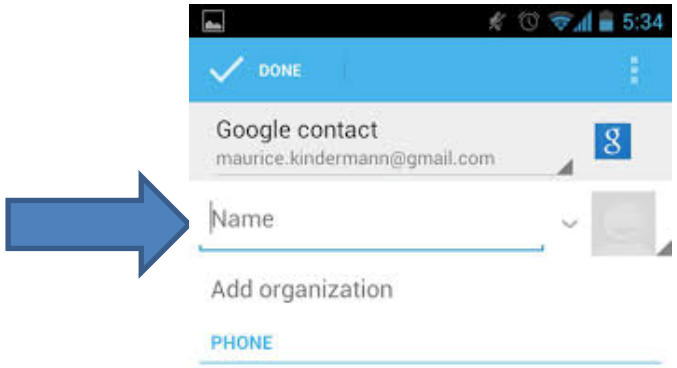
Exercise (cont.)

```java
package com.example.myapplication;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import java.util.Date;

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

        button = findViewById(R.id.myButton);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                updateTime();
            }
        });

    }

    void updateTime() {
        button.setText(new Date().toString());
    }
}
```

RUN !!

Widgets: EditText Boxes

- The **EditText** (or textBox) widget is an extension of TextView that allows updates.
- The control configures itself to be *editable.*
- View Class: EditText
  - http://developer.android.com/reference/android/widget/EditText.html
- Important Java I/O methods are:



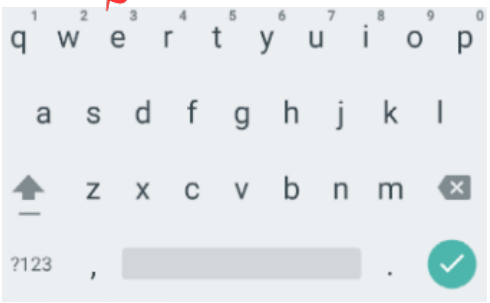

```
txtBox.setText("someValue");

txtBox.getText().toString();
```

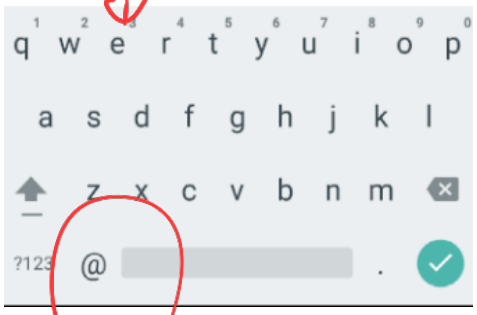
Widgets: EditText

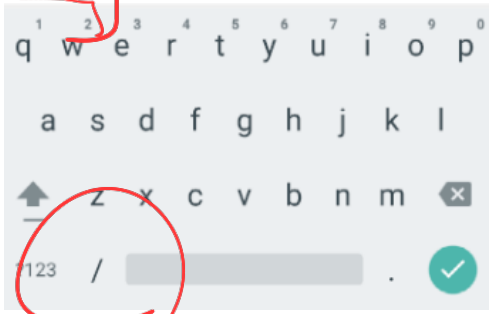
- Specifying the Keyboard Type
  - android:inputType attribute
    - "text" : Normal text keyboard.
    - "textEmailAddress": Normal text keyboard with the @ character.
    - "textUri" : Normal text keyboard with the / character.

```
<EditText
    android:id="@+id/email_address"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/email_hint"
    android:inputType="textEmailAddress" />
```

키보드 타입 설정 가능?



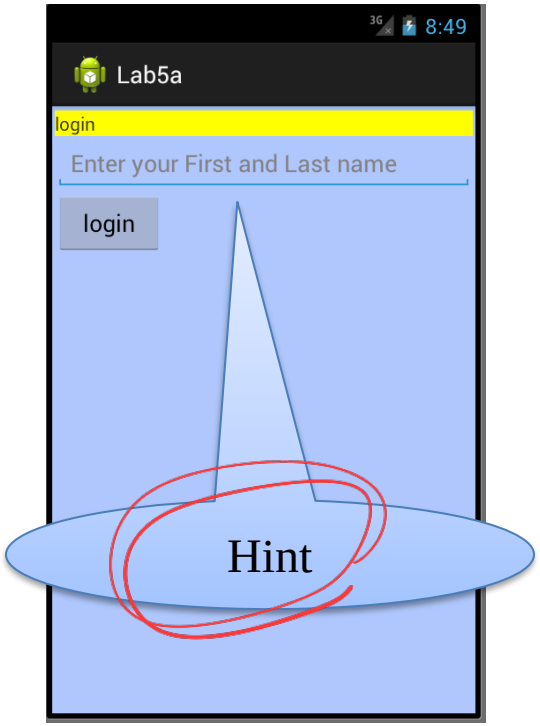


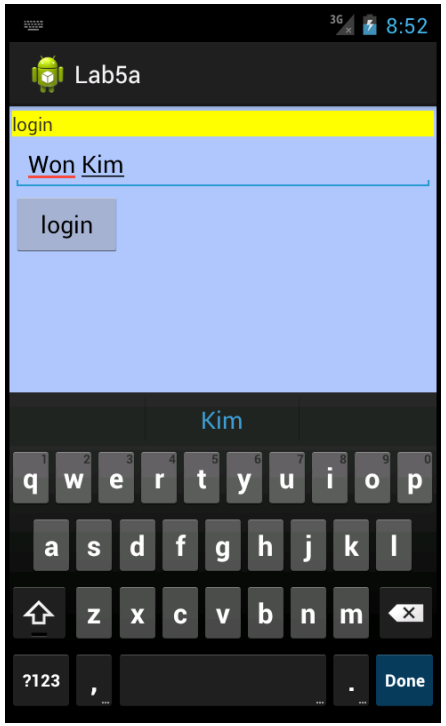


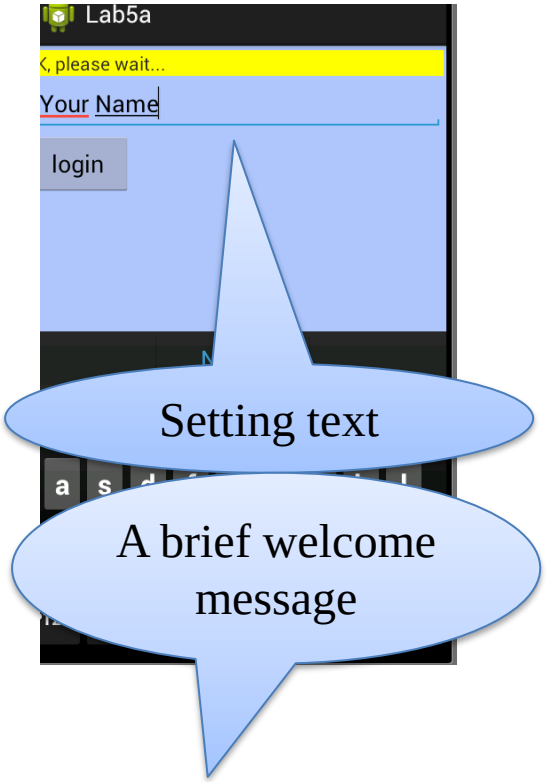
Exercise

In this little example we will use an **LinearLayout** holding a label(TexView), a textBox(EditText), and a Button.

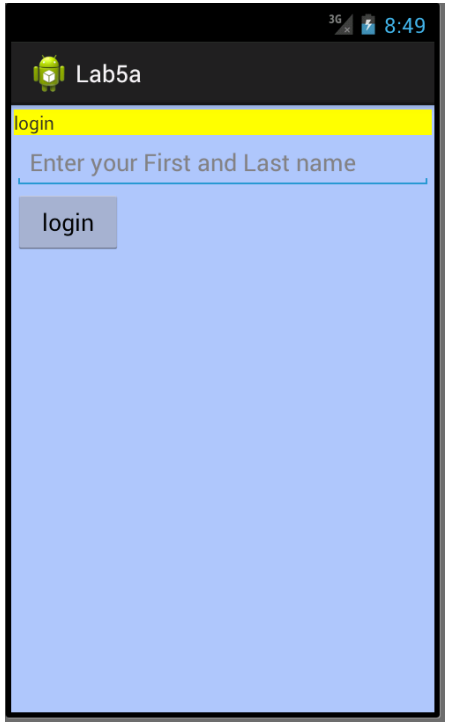
We will use the view as a sort of simplified login screen.







Exercise (cont.)



- Layout: LinearLayout
  - Orientation : vertical
  - Background color: #886495ed

- Three Widgets
  - Label (TextView)
    - Background color : yellow
  - TextBox (EditText)
    - Hint property
    - inputType: textCapWords + textAutoCorrect
  - Button

Exercise (cont.)

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#886495ed"
    android:orientation="vertical"
    android:padding="2dp" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="1dp"
        android:background="#ffffff00"
        android:text="@string/ACME_Corp_Caption" />

    <EditText
     android:id="@+id/txtUserName"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:layout_marginTop="1dp"
     android:hint="@string/Enter_your_First_and_Last_name"
     android:textSize="18sp" >
</EditText>
```

오탈자 주의!

```
  <Button
     android:id="@+id/button1"
     android:layout_width="82dp"
     android:layout_height="wrap_content"
     android:layout_marginTop="1dp"
     android:text="@string/login" />

</LinearLayout>
```

Exercise (cont.)

- Java code

```
package com.example.myapplication;
import android.content.Context;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.util.Date;

public class MainActivity extends AppCompatActivity {
    TextView labelUserName;
    EditText txtUserName;
    Button btnBegin;
    private Context context;
    private int duration = Toast.LENGTH_SHORT;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        context = getApplicationContext();

        labelUserName=findViewById(R.id.textView1);
        txtUserName=findViewById(R.id.txtUserName);
        btnBegin=findViewById(R.id.button1);
```

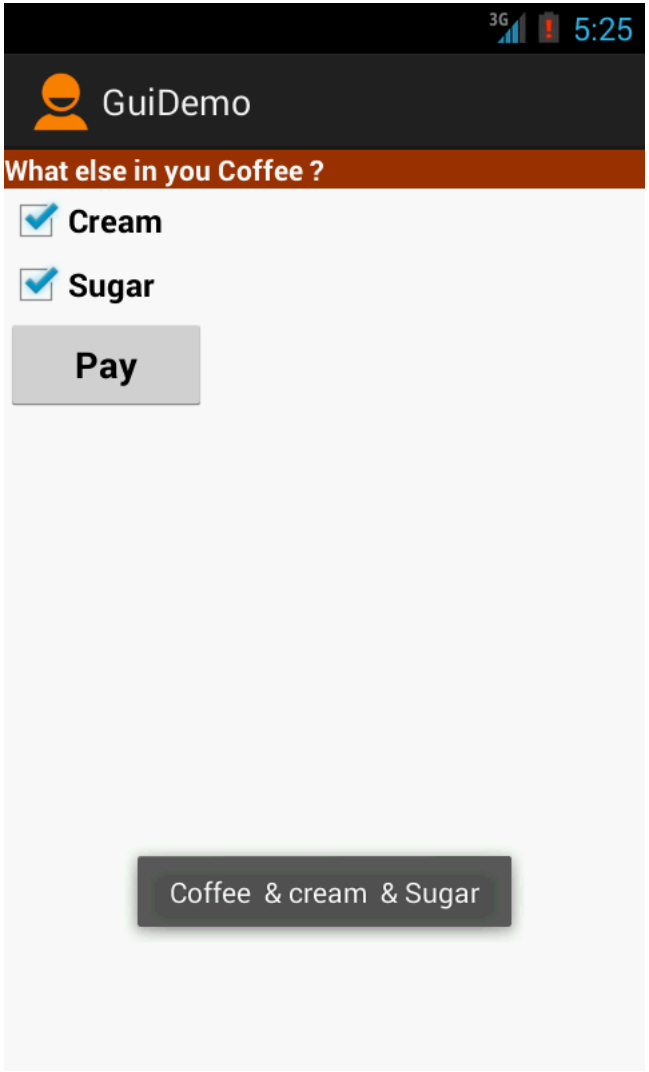
Exercise (cont.)

```
        btnBegin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String userName = txtUserName.getText().toString();
                if(userName.compareTo("Younhyun Jung")==0){
                    labelUserName.setText("OK, Please wait..");
                    Toast.makeText(context, "Hi!, Prof."+userName, duration).show();
                }
                else{
                    Toast.makeText(context, userName+ " is not a valid User", duration).show();
                }
            }
        });

    }
}
```

String 비교

Widgets: CheckBox

- A checkbox is a specific type of button with two-states
  - either *checked* or *unchecked*.
- An example usage of a checkbox :
  - (two check buttons)



Widgets: CheckBox

- XML element

```
<CheckBox
    android:id="@+id/chkCream"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Cream"
    android:textStyle="bold"
    >
</CheckBox>
```

- Java method
  - `import android.widget.CheckBox;`
  - `CheckBox chkCream;`
  - `chkCream = (CheckBox) findViewById(R.id.chkCream);`
  - `chkCream.isChecked()`
    - Observe/report method - isChecked()

Exercise

- **Resources: res/values/strings**

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">MyApplication2</string>
    <string name="sugar">Sugar</string>
    <string name="cream">Cream</string>
    <string name="coffee_addons">What else in your coffee?</string>
    <string name="pay">Pay</string>
</resources>
```

Exercise

- **Layout: res/layout/activity_main.xml**

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="6dp"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/labelCoffee"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff993300"
        android:text="@string/coffee_addons"
        android:textColor="@android:color/white"
        android:textStyle="bold" />
    <CheckBox
        android:id="@+id/chkCream"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/cream"
        android:textStyle="bold" />
```

```xml
 <CheckBox android:id="@+id/chkSugar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/sugar"
        android:textStyle="bold" />
    <Button android:id="@+id/btnPay"
        android:layout_width="153dp"
        android:layout_height="wrap_content"
        android:text="@string/pay"
        android:textStyle="bold" />
</LinearLayout>
```

Exercise

- **Java: MainActivity**

```java
package com.example.myapplication;
import android.content.Context;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.util.Date;

public class MainActivity extends AppCompatActivity {
    CheckBox chkCream;
    CheckBox chkSugar;
    Button btnPay;

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

        chkCream = findViewById(R.id.chkCream);
        chkSugar = findViewById(R.id.chkSugar);
        btnPay = findViewById(R.id.btnPay);
```

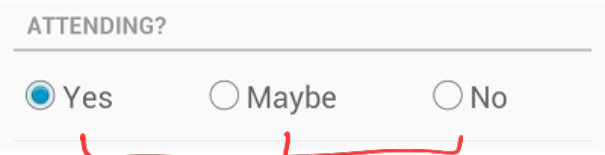
```java
        btnPay.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String msg = "Coffee ";
                if (chkCream.isChecked()){
                    msg += " & cream ";
                }
                if (chkSugar.isChecked()){
                  msg += " & Sugar";
                }
                Toast.makeText(getApplicationContext(), msg,
Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

Checkbox에 event Listener를 다는 형태는 아님.

Exercise

- **RUN!!!**

Widgets: RadioButtons



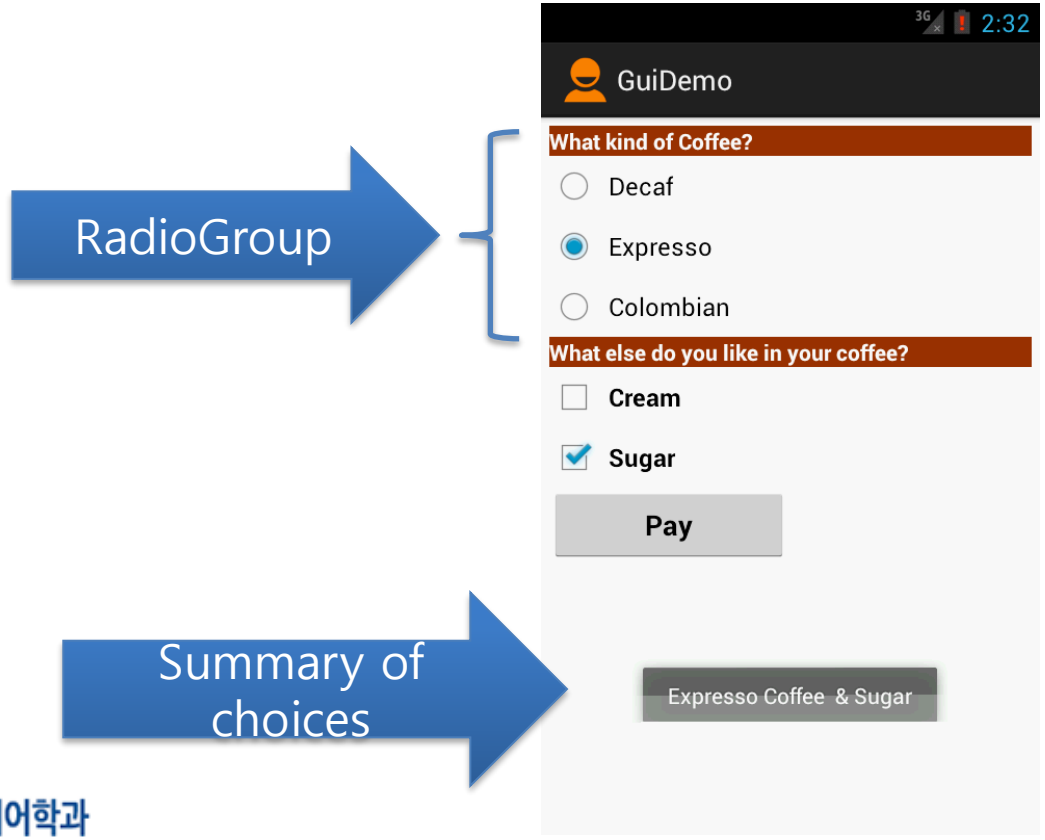
- A radio button is a two-states button
  - ***checked*** *or* ***unchecked.***
  - When the radio button is unchecked, the user can press or click it to check it.

- Radio buttons are normally used together in a **RadioGroup**.
  - When several radio buttons live inside a radio group,
    it allows only one RadioButton to be checked within the Radiogroup,
    (checking one radio button ***unchecks*** all *the others.)*

- **isChecked() -** method to check whether it is checked or not
  - Similarly, you can call **isChecked()** on a RadioButton to see if it is selected, like you can with a CheckBox

Widgets: RadioButtons

- We will extend the previous example (CheckBox) by adding a *RadioGroup and three RadioButtons*



Exercise

- Add XML elements for the new parts into previous exercise

```
<RadioGroup
    android:id="@+id/radGroupCoffeeType"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <TextView
        android:id="@+id/labelCoffeeType"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff993300"
        android:text="What type of coffee?"
        android:textStyle="bold">
    </TextView>

    <RadioButton
        android:id="@+id/radDecaf"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Decaf">
    </RadioButton>
```

```
    <RadioButton
        android:id="@+id/radExpresso"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Expresso"
        >
    </RadioButton>
    <RadioButton
        android:id="@+id/radColombian"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Colombian">
    </RadioButton>
</RadioGroup>
```

Exercise (cont.)

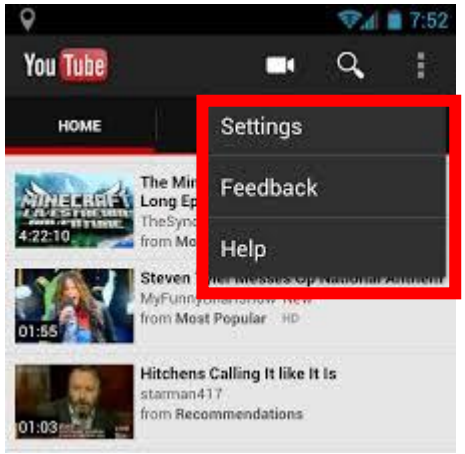
MainActivity

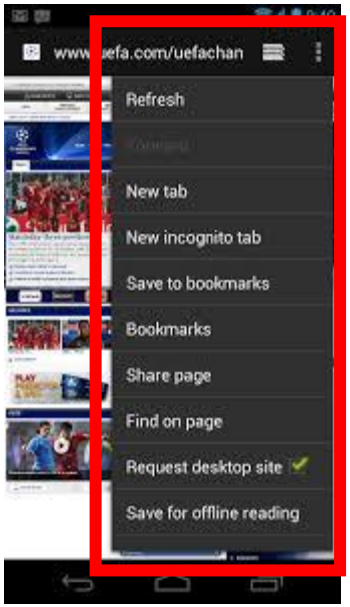
```java
package com.example.myapplication;
import android.content.Context;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;
import java.util.Date;

public class MainActivity extends AppCompatActivity {
    CheckBox chkCream;
    CheckBox chkSugar;
    Button btnPay;
    RadioGroup radCoffeeType;
    RadioButton radDecaf;
    RadioButton radEspresso;
    RadioButton radColombian;

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

        chkCream = findViewById(R.id.chkCream);
        chkSugar = findViewById(R.id.chkSugar);
        btnPay = findViewById(R.id.btnPay);
        radCoffeeType = findViewById(R.id.radGroupCoffeeType);
        radDecaf = findViewById(R.id.radDecaf);
        radEspresso = findViewById(R.id. radExpresso);
        radColombian = findViewById (R.id.radColombian);
```

```java
btnPay.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String msg = "Coffee ";
                if (chkCream.isChecked()){
                    msg += " & cream ";
                }
                if (chkSugar.isChecked()){
                  msg += " & Sugar";
                }
                int radiold =
radCoffeeType.getCheckedRadioButtonId();
                if (radDecaf.getId()==radiold)
                    msg = "Decaf " + msg;
                if (radColombian.getId()==radiold)
                    msg = "Colombian " + msg;
                if (radEspresso.isChecked())
                    msg = "Espresso " + msg;

                Toast.makeText(getApplicationContext(), msg,
Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

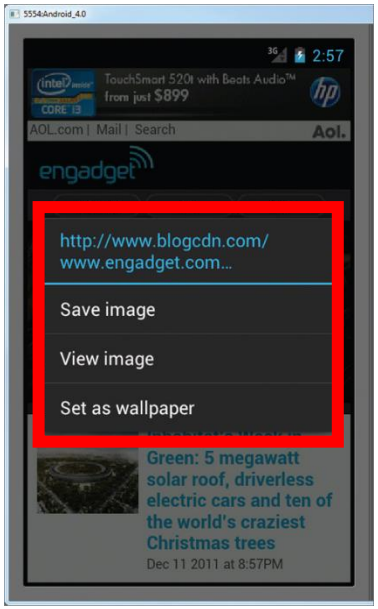
radioId를 체크해서
radiobutton Id와
비교!

Widgets : Menus

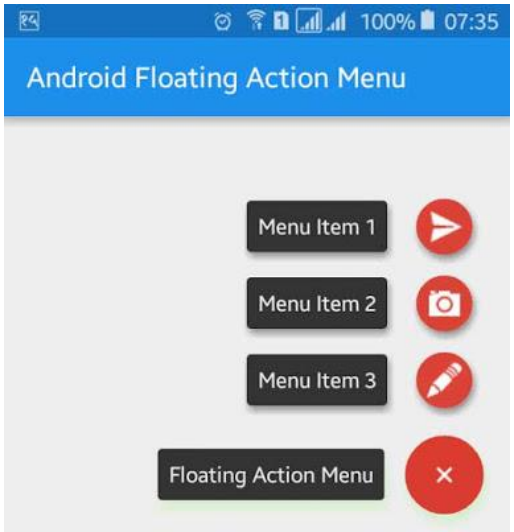




options menu



context menu



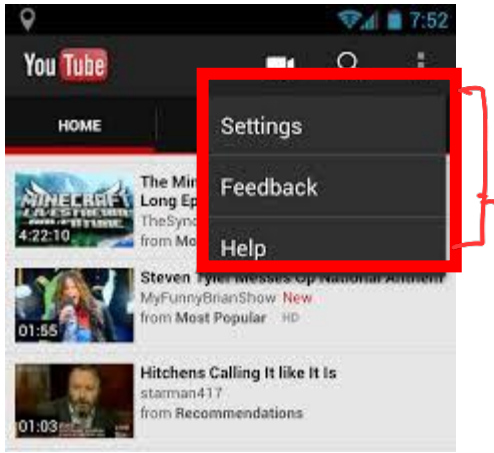
Floating action buttons(menu)

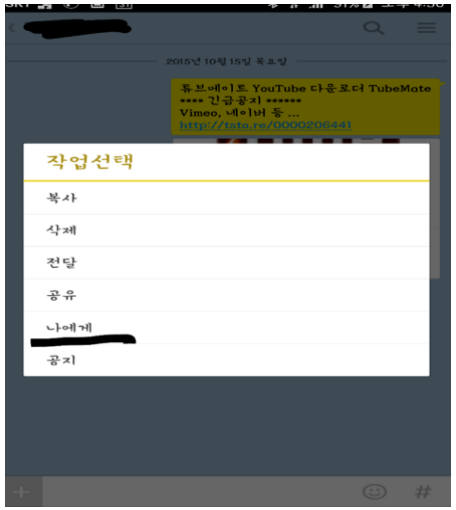
Widgets : Menus

- **Menus**
  - **a common user interface component in many types of applications.**
  - useful for displaying additional options that are not directly visible on the main UI of an application
  - not consuming 'much' view space

    필요할 때만 나오게 해서 공간 절약할 수 있다?
    - When displayed, a menu is shown as an overlapping layer on top of the current UI. After making a selection, the exposed Menu layer disappear

Widgets : Menus





- Two types : **options menu** and **context menu**.
  - **Options menu**

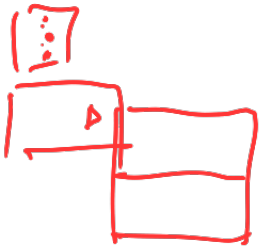
    On Android 3.0 and higher, items from the options menu are presented by the action bar as a combination of on-screen action items and overflow options.

  - **Contextual menu (or Context menu)**

    A context menu is a floating menu that appears when the user performs a long-click on an element.

Defining a Menu in XML

- **res/menu/** directory
  - To define the menu, create an XML file inside your project's **res/menu/** directory and build the menu with the following elements:
  - <menu>
    - Defines a Menu, which is a container for menu items. A <menu> element must be the root node for the file and can hold one or more <item> and <group> elements.
  - <item>
    - Creates a Menu Item, which represents a single item in a menu. This element may contain a nested <menu> element in order to create a submenu.
  - <group>
    - An optional, invisible container for <item> elements. It allows you to categorize menu items so they share properties such as active state and visibility.



game_menu.xml:

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:id="@+id/file"
        android:title="@string/file" >
      <!-- "file" submenu -->
      <menu>
         <item android:id="@+id/create_new"
              android:title="@string/create_new" />
         <item android:id="@+id/open"
              android:title="@string/open" />
      </menu>
   </item>
</menu>
```

별도 xml로 구현.

Sub menu 구현 가능?

Defining a Menu in XML

- Example : **res/menu/ game_menu.xml**

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/new_game"
          android:icon="@drawable/ic_new_game"
          android:title="@string/new_game"
          android:showAsAction="ifRoom"/>
    <item android:id="@+id/help"
          android:icon="@drawable/ic_help"
          android:title="@string/help" />
</menu>
```

R.id.new_game

- Attributes
- android:id
  - A resource ID
- android:icon
  - the item's icon.
- android:title
- android:showAsAction
  - Define How appear in ActionBar

ifRoom, withText, never, always

Create Menu in Java

아이템을 항상(always) 앱바(App Bar)의 액션으로 표시.
never와 ifRoom보다 우선되며, 공간이 없으면 표시되지 않음.

아이템을 앱바(App Bar)의 액션으로 표시하지 않고(never) 오버플로우 메뉴에 바로 표시.

만약 공간이 있다면(ifRoom), 앱바(App Bar)의 액션으로 표시하고, 공간이 없다면 오버플로우 메뉴에 표시.

아이템을 앱바(App Bar)의 액션으로 표시할 때 텍스트와 같이(withText) 표시.
단, 아이콘과 텍스트를 같이 표시할 공간이 있는 경우에만 텍스트 표시.

Handling click events

- **Handling click events**
  - When the user selects an item from the options menu (including action items in the action bar), the system calls your activity's onOptionsItemSelected() method.
    - This method passes the MenuItem selected.

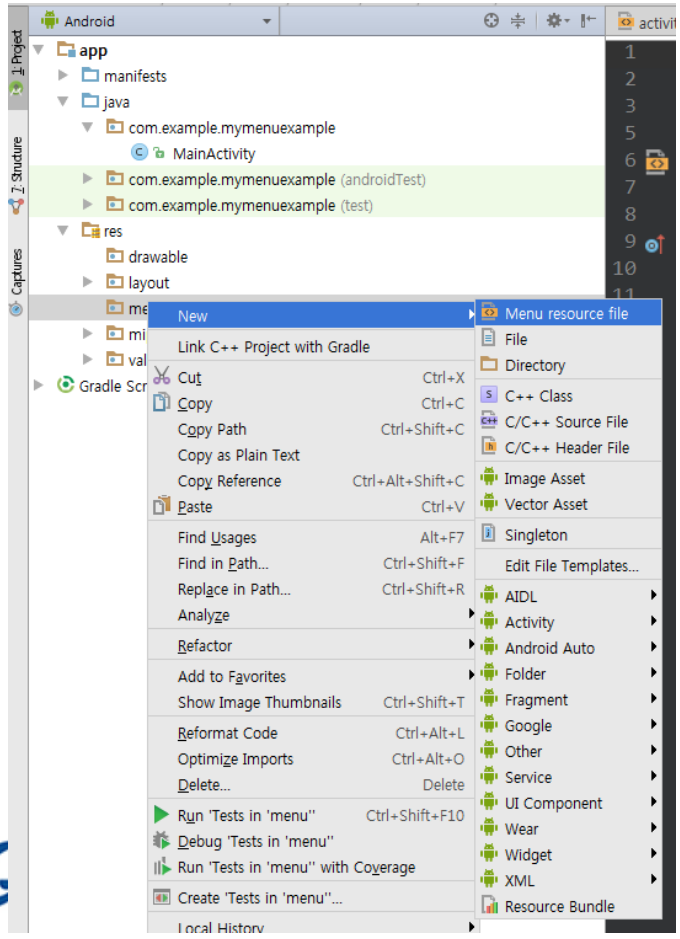
```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle item selection
    switch (item.getItemId()) {
        case R.id.new_game:
            newGame();
            return true;
        case R.id.help:
            showHelp();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

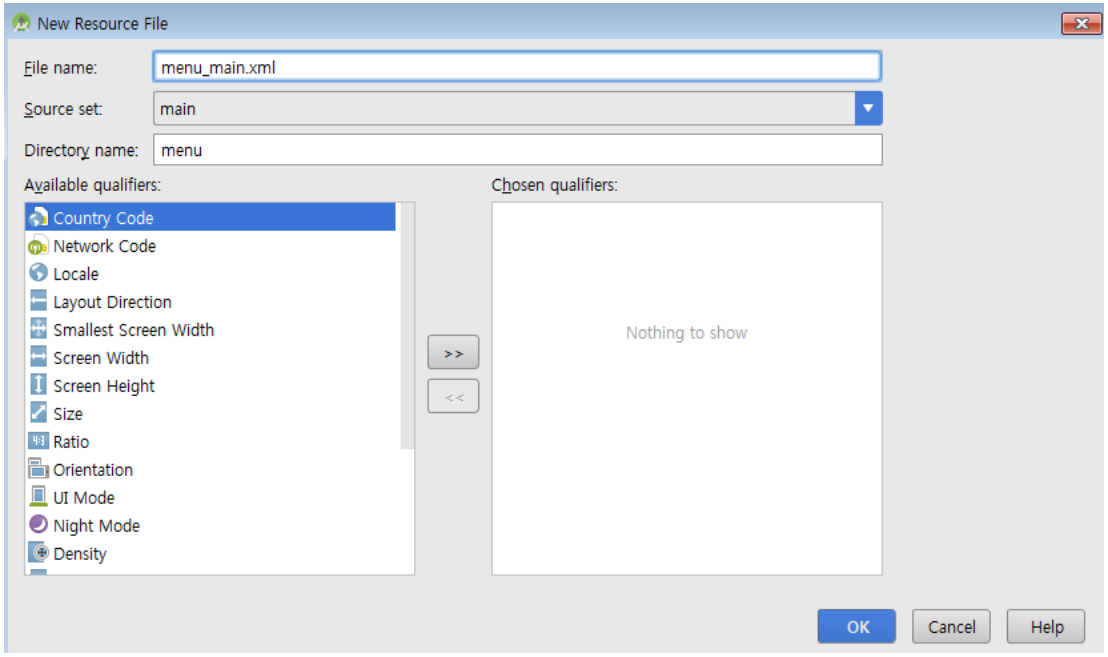
item의 id를 반환

You can identify the item by calling getItemId(), which returns the unique ID for the menu item

Exercise

- The New > '**Android Application Project**' wizard allows you to choose a "Empty Activity".
- Make menu folder in "res" folder →make menu_main.xml file.





Exercise (cont.)

- Example : **res/menu/menu_main.xml**

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item android:id="@+id/menu_refresh"
        android:title="새로고침"
        app:showAsAction="withText"/>
    <item android:id="@+id/menu_search"
        android:title="검색"
        app:showAsAction="withText"/>
    <item android:id="@+id/menu_settings"
        android:title="설정"
        app:showAsAction="withText"/>
</menu>
```

Exercise (cont.)

- Add override methods in MainActivity.java

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int curId = item.getItemId();
    switch(curId){
        case R.id.menu_refresh:
            Toast.makeText(this, "새로고침 메뉴가 선택 됨", Toast.LENGTH_SHORT).show();
            break;
        case R.id.menu_search:
            Toast.makeText(this, "검색 메뉴가 선택 됨", Toast.LENGTH_SHORT).show();
            break;

        case R.id.menu_settings:
                Toast.makeText(this, "설정 메뉴가 선택 됨", Toast.LENGTH_SHORT).show();
                break;
    }
    return super.onOptionsItemSelected(item);
}
```

Exercise (cont.)

- Our example produces the following snapshots: