Lecture 3

- Inheriting a View
- ListView and Array Adapter
- Custom row in ListView
- Fragments in static layout
- Fragments in dynamic layout
- Workshop
- Switching visible Fragments dynamically during execution

Inheriting a View

It is possible to inherit a View, for example, a TextView. In the example a TextView component, which always shows the text that rövarspråket.

```
public class TRLTextView extends TextView {
    private static String consonants = "bcdfghjklmnpqrstvwxzBCDFGHJKLMNPQRSTVWXZ";

    public TRLTextView(Context context) {
        super(context);
    }

    public TRLTextView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public TRLTextView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    public void setText(CharSequence text, TextView.BufferType type) {
        StringBuilder str = new StringBuilder();
        for(int i=0; i<text.length(); i++) {
            if(consonants.indexOf(text.charAt(i)) >=0) {
                str.append(text.charAt(i));
                str.append('o');
                str.append(text.charAt(i));
            } else {
                str.append(text.charAt(i));
            }
        }
        super.setText(str.subSequence(0,str.length()),type);
    }
}
```

Contructors

Override setText.

- Build a new String
- Call super.setText()

An inherited View in a Layout

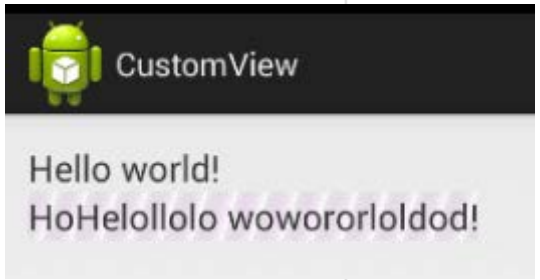
When the new component must be placed in a layout, you must use the full (advanced) class name, e.g. se.mah.tsroax.f3.TRLTextView

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".MyActivity">

    <TextView
        android:text="@string/hello_world"
        android:textSize="20sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <se.mah.tsroax.f3.TRLTextView
        android:text="@string/hello_world"
        android:textSize="20sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
```

Inheriting a View

A little color in the background is always fun

```
public class TRLTextView extends TextView {
    private static String consonants = "bcdfghjklmnpqrstvwxzBCDFGHJKLMNPQRSTVWXZ";
    private static Paint paint;

    public TRLTextView(Context context) {
        super(context);
        init();
    }

    public TRLTextView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    public TRLTextView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        init();
    }

    public void setText(CharSequence text, TextView.BufferType type) {...}

    private void init() {
        if(paint==null) {
            paint = new Paint();
            paint.setShader(new LinearGradient(0, 0, 30, 10, 0xD0A0D0, Color.WHITE, Shader.TileMode.REPEAT));
        }
    }

    @Override
    protected void onDraw(Canvas canvas) {
        canvas.drawPaint(paint);
        super.onDraw(canvas);
    }
}
```



Initiating drawing tools

Override onDraw

- The background is painted and then super.onDraw is called.

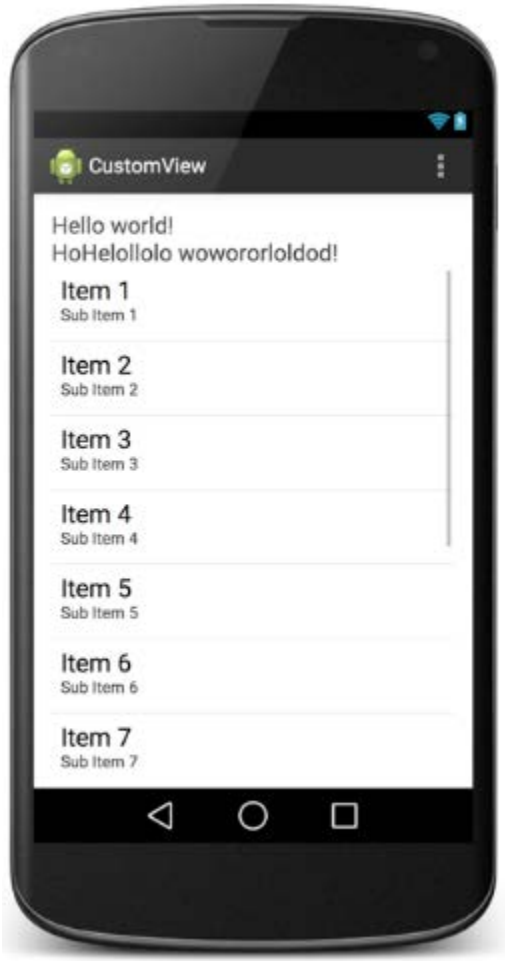
ListView and ArrayAdapter

ListView component is used extensively in applications to display information in list form.

ListView component needs a List adapter to fill the rows in the list of information.

List Adapter is an interface with many methods.

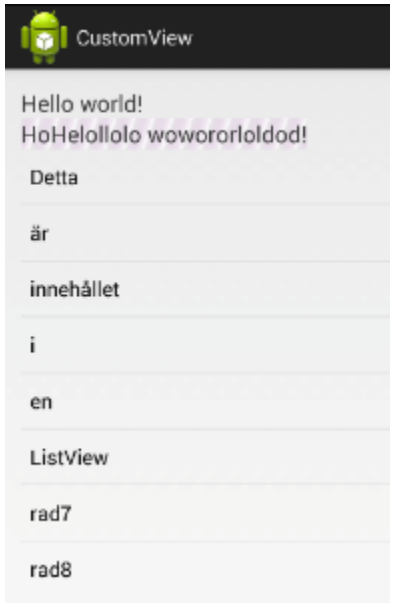
Array Adapter implements List Adapter and is very useful for this.



ListView and ArrayAdapter

A ListView must be declared in the layout file.

You have to create an ArrayAdapter (with any kind of content) and connect the adapter to the List View component.




```
        android:layout_height="wrap_content" />

    <se.mah.tsroax.f3.TRLTextView
        android:text="@string/hello_world"
        android:textSize="20sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <ListView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/lvTrl" />

</LinearLayout>
```

```
public class MyActivity extends Activity {
    private String[] content = {"Detta","är","innehållet","i","en","ListView",

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

    private void initializeComponents() {
        ListView lvTrl = (ListView)findViewById(R.id.lvTrl);
        lvTrl.setAdapter(new ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,content));
    }
```

setAdapter passes the selected adapter to the List View.

3 arguments :

- Context (this)
- Layout for every line in the list
- Data (array or list)

ListView and Array Adapter

OnItemClickListener handles the clicks on every row.

```
private void initializeComponents() {
    ListView lvTrl = (ListView)findViewById(R.id.lvTrl);
    lvTrl.setAdapter(new ArrayAdapter<String>(this,R.layout.trl_row,content));
    lvTrl.setOnItemClickListener(new ListViewListener());
}
```

Create a short internal class (e.g. ListViewListener) an override the onItemClick method.

```
private class ListViewListener implements android.widget.AdapterView.OnItemClickListener {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        TextView tv = (TextView)view;
        Log.d("OICL",content[position] + " - " + tv.getText());
    }
}
```

Arguments:

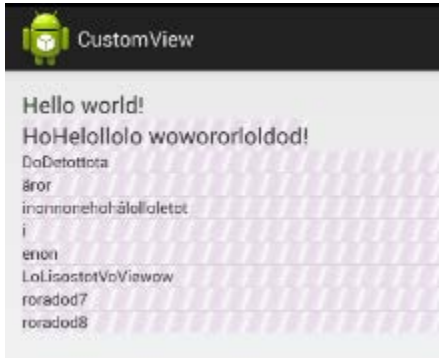
- The parent View (where the click happened)
- This View (the clicked row).
- The position of the clicked row in the list.
- The id of the clicked row.

Access the content of the clicked row as follows:

```
TextView tv = (TextView)view;
tv.getText();
```

ListView and Array Adapter

If you have a TextView per line in the list, you can make your own layout using the Array Adapter.



Save it as trl_row.xml

```
<?xml version="1.0" encoding="utf-8"?>
<se.mah.tsroax.f3.TRLTextView xmlns:android="http://schemas.android.com/apk
    android:layout_width="match_parent"
    android:layout_height="match_parent">

</se.mah.tsroax.f3.TRLTextView>
```

Then you specify the layout when the Array Adapter object is created.

```
public class MyActivity extends Activity {
    private String[] content = {"Detta","är","innehållet","i","en","ListView", "rad7","rad8"};

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

    private void initializeComponents() {
        ListView lvTrl = (ListView)findViewById(R.id.lvTrl);
        lvTrl.setAdapter(new ArrayAdapter<String>(this,R.layout.trl_row,content));
    }
```

Specify the layout file trl_row for the ArrayAdapter when it is instantiated.

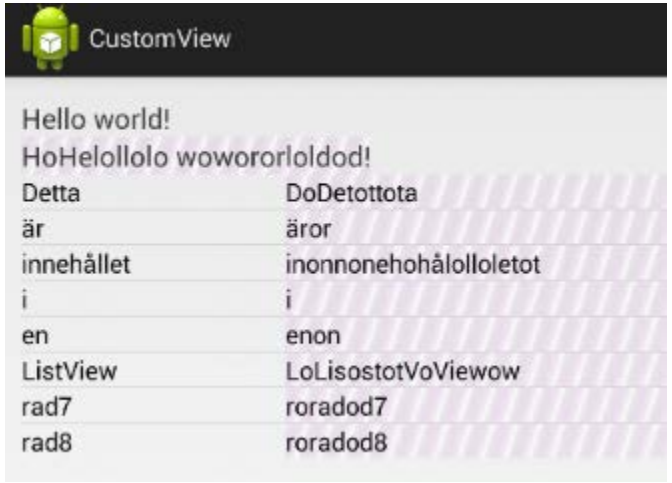
Several components in a row

A layout file must be created for the line

You have to create an Array Adapter (with any kind of content) and connect the adapter to the List View component.

```
<?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">

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_weight="1"
        android:text="Medium Text"
        android:id="@+id/tvNormal" />

    <se.mah.tsroax.f3.TRLTextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_weight="2"
        android:text="Medium Text"
        android:id="@+id/ttvTrl" />
</LinearLayout>
```

Saved as listview_item.xml



Several components in a row

Create your own Adapter by extending ArrayAdapter

```
public class TRLAdapter extends ArrayAdapter<String> {
    private LayoutInflater inflater;

    public TRLAdapter(Context context, String[] objects) {
        super(context, R.layout.listview_item, objects);
        inflater = (LayoutInflater)context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        TextView tvNormal;
        TRLTextView tvTrl;
        if(convertView==null) {
            convertView = (LinearLayout)inflater.inflate(R.layout.listview_item, parent, false);
        }
        tvNormal = (TextView)convertView.findViewById(R.id.tvNormal);
        tvTrl = (TRLTextView)convertView.findViewById(R.id.ttvTrl);
        tvNormal.setText(this.getItem(position));
        tvTrl.setText(this.getItem(position));
        return convertView;
    }
}
```

Arguments:
- Context
- Layout for the row
- Data (array or List)

LayoutInflater “inflates” the objects from the layout.

You must reuse rows, otherwise the app will consume lots of resources (battery killer!)
If the current View is empty, then we inflate it from the layout.
Otherwise, we proceed straight to setting its content.
The View must be returned.

Override getView:

We control what happens when the UI asks for a row to be painted.

Several components in a row

Finally, a TRLAdapter object is connected to the ListView component

```
public class MyActivity extends Activity {
    private String[] content = {"Detta","är","innehållet","i","en","ListView", "rad7","rad8"};

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

    private void initializeComponents() {
        ListView lvTrl = (ListView)findViewById(R.id.lvTrl);
        lvTrl.setAdapter(new TRLAdapter(this,content));
    }
```

ViewHolder Pattern

E. g. a line in TRLAdapter consists of a linear layout containing a TextView and TRLTextView.

Even reusing views, we are still calling findViewByID twice in every call to getView.

This is still very resource consuming.

Therefore, we are tagging every row so that we can store a reference to it and skip calling repeatedly findViewByID.

We are using a ViewHolder:

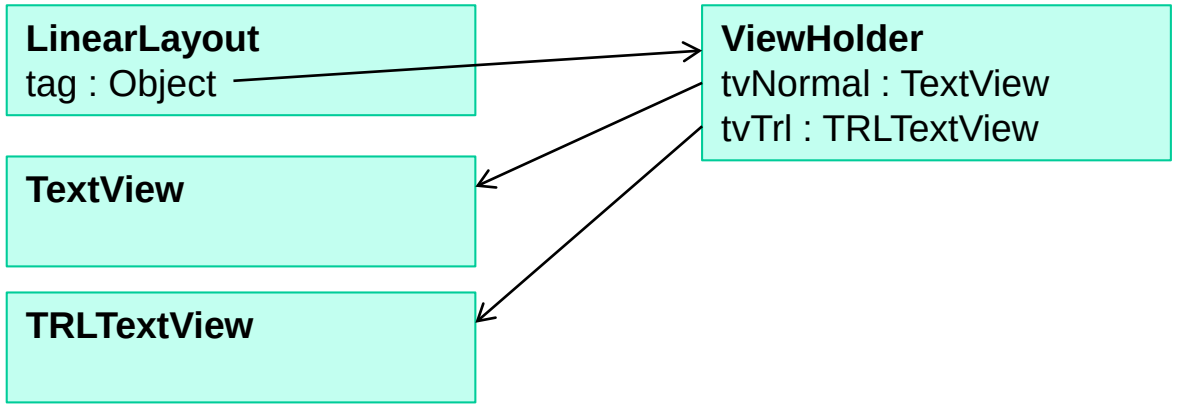
```
private class ViewHolder {
    TextView tvNormal;
    TRLTextView tvTrl;
}
```

ViewHolder Pattern

Now in getView we:

- If the row view is null:
  - Create a Viewholder object.
  - Link this object to the View's content.
  - Tag the View with the ViewHolder.
- Otherwise:
  - Retrieve the ViewHolder that we're looking for using its tag.

For each row, we build these systems:



ViewHolder Pattern

```
public class TRLAdapter extends ArrayAdapter<String> {
    private LayoutInflater inflater;

    public TRLAdapter(Context context, String[] objects) {
        super(context, R.layout.listview_item, objects);
        inflater = (LayoutInflater)context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder holder;
        if(convertView==null) {
            convertView = (LinearLayout)inflater.inflate(R.layout.listview_it
            holder = new ViewHolder();
            holder.tvNormal = (TextView)convertView.findViewById(R.id.tvNormal);
            holder.tvTrl = (TRLTextView)convertView.findViewById(R.id.ttvTrl);
            convertView.setTag(holder);
        } else {
            holder = (ViewHolder)convertView.getTag();
        }
        holder.tvNormal.setText(this.getItem(position));
        holder.tvTrl.setText(this.getItem(position));
        return convertView;
    }

    class ViewHolder {
        TextView tvNormal;
        TRLTextView tvTrl;
    }
}
```

If the View (row) is null, inflate it, bind it to a ViewHolder, and use this to tag the View.

Otherwise, retrieve the ViewHolder tagged to the View.

Fill it in.

Place Fragments in static layout

Fragments can be placed in a layout with fragments tag. How does the Activity's layout file look like?

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical“ >

    <fragment
        android:id="@+id/viewer_fragment"
        android:name="se.mah.tsroax.staticfragment.ViewerFragment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <fragment
        android:id="@+id/input_fragment"
        android:name="se.mah.tsroax.staticfragment.InputFragment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

FragmentManager

Activities can manage their Fragments with a FragmentManager. E.g. it is possible to make references to different fragments from the Activity using findFragmentByID.

```
FragmentManager fm = getFragmentManager();
ViewerFragment viewer =
  (ViewerFragment)fm.findFragmentById(R.id.viewer_fragment);
InputFragment input =
  (InputFragment)fm.findFragmentById(R.id.input_fragment);
```

With Fragment Manager object you can also start a Fragment Transaction: you can add / remove / replace fragments in a container.

Dynamic Placement of Fragments in a layout

If you only enter one (or more containers) in the layout (e.g. Linear layout), they may be used to hold fragments at runtime. In the layout below, two containers can be used:

- "upper_container"
- "lower_container"

```
<LinearLayout xmlns:android="http://..."
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <FrameLayout
        android:id="@+id/upper_container"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
    </FrameLayout>

    <FrameLayout
        android:id="@+id/lower_container"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
    </FrameLayout>

</LinearLayout>
```

Dynamic Placement of Fragments in a layout

Fragments can be dynamically placed in different layouts thanks to the Transactions provided by the FragmentManager.

A transaction begins with the invocation of the method beginTransaction and ends with a call to commit.

```
FragmentManager fragManager = getFragmentManager();
FragmantTransaction fragTransaction = fragManager.beginTransaction()

// add actions to the transaction

fragTransaction.commit();
```

Dynamic Placement of Fragments in a layout

```
<FrameLayout
    android:id="@+id/upper_container"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
</FrameLayout>
```

Place a Fragment object in the container with the method add.

```
FragmentTransaction transcation = fragManager.beginTransaction();
transaction.add( R.id.upper_container, fragment );
transaction.commit();
```

1. Request a transaction from the FragmentManager.
2. Add the target layout and Fragment object to the transaction using add.
3. Call commit. The action will never happen if you don't call this method.

Remove fragments dynamically

```
<FrameLayout
    android:id="@+id/upper_container"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
</FrameLayout>
```

Remove fragments from the container with the method remove

```
FragmentTransaction transcation = fragManager.beginTransaction();
transaction.remove( fragment );
transaction.commit();
```

remove method is called with one argument: a reference to the Fragment object to be removed.

Replace Fragments dynamically

```
<FrameLayout
    android:id="@+id/upper_container"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
</FrameLayout>
```

Switch Frements by calling ***replace***

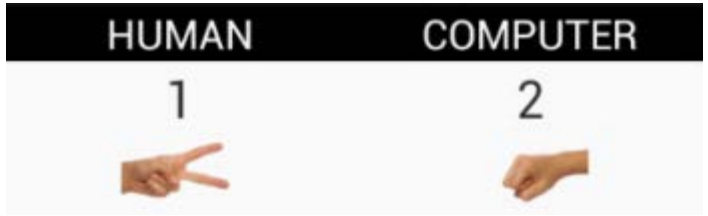
```
FragmentTransaction transcation = fragManager.beginTransaction();
transaction.replace( R.id.upper_container, fragment );
transaction.commit();
```

Use replace as you would use add, passing the replacing fragment as an argument.

Workshop

A Rock, Paper, Scissors game! Offer the user two different Uis using two different Input and Viewer fragments

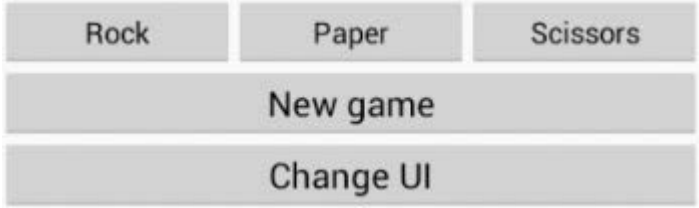
ImageViewerFragment



TextViewerFragment

ImageInputFragment

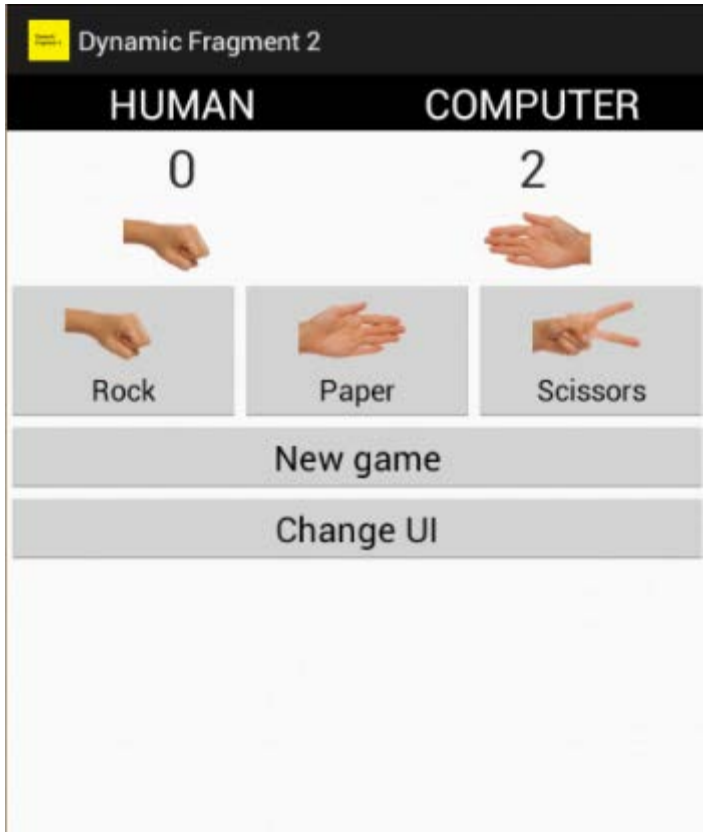
TextInputFragment



DynamicRPS

Workshop

User can select UI with images or without images, click on the Change UI - button



Interfaces

Three interfaces are used in the program. Viewers shall be simply replaced with one another, as well as Inputs and Controllers.

Nevertheless, **Controllers are not replaced in this workshop.**

Viewer Interface

```
public interface Viewer {
    public void setController(Controller controller);
    public void setHumanPoints(int points);
    public void setComputerPoints(int points);
    public void setHumanChoice(int choice);
    public void setComputerChoice(int choice);
    public void humanWinner();
    public void computerWinner();
}
```

Input Interface

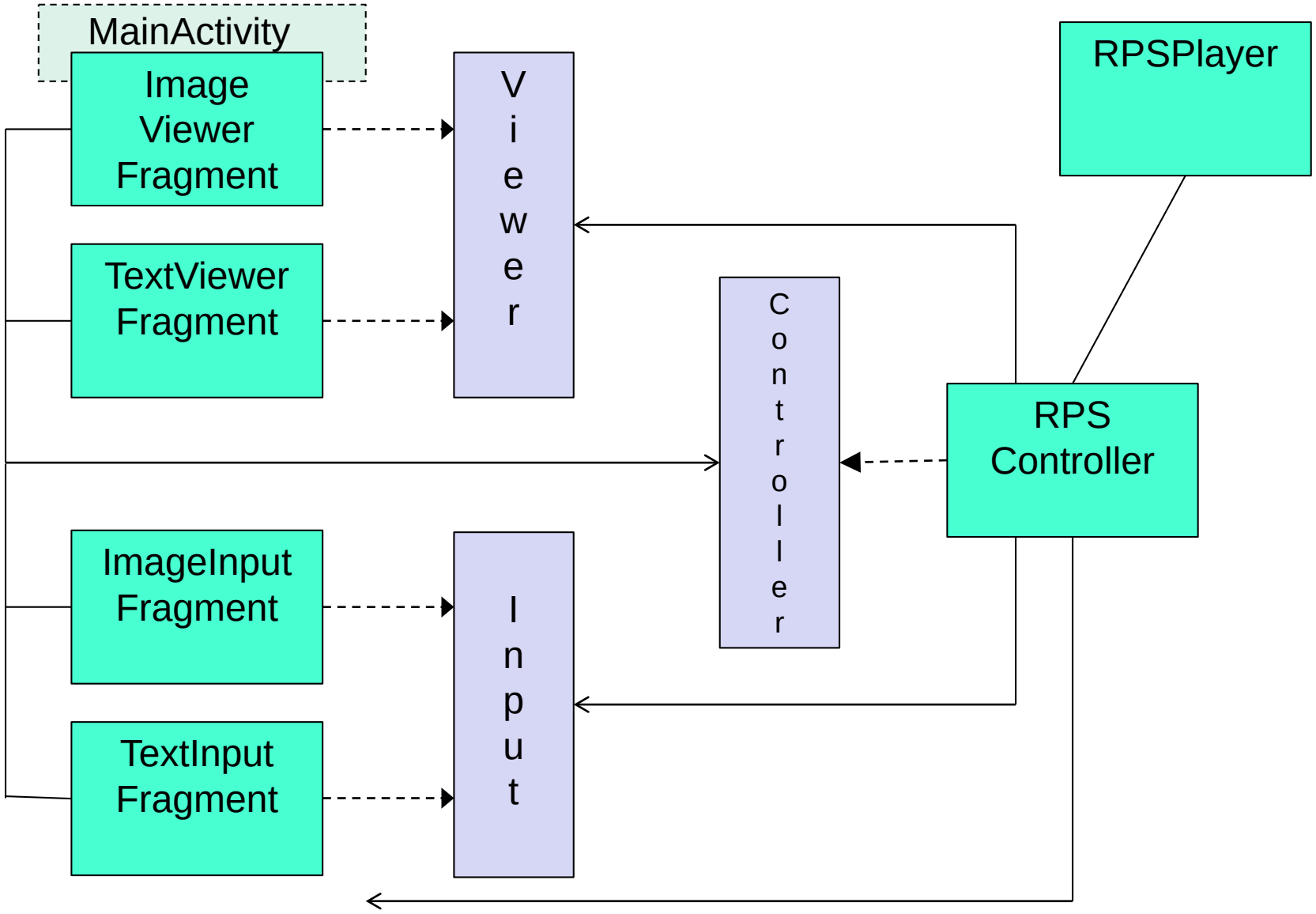
```
public interface Input {
    public void setController(Controller controller);
    public void enableChoiceButtons(boolean enabled);
}
```

Interfaces

Interface Controller

```
public interface Controller {
    public static final int ROCK = 0, PAPER = 1, SCISSORS = 2,
        EMPTY = 3, WINNER = 4, LOSER = 5;
    public void newGame();
    public void newChoice(int choice);
    public void updateViewer();
    public void changeUI();
}
```

Class Diagram



MainActivity

```
public class MainActivity extends Activity {
    protected void onCreate(Bundle savedInstanceState) {
        :
    }

    public boolean onCreateOptionsMenu(Menu menu) {
        :
    }

    private void initController() {
        // instantierar objekt
    }

    public void setViewer(Viewer viewer) {
        FragmentManager fragManager = getFragmentManager();
        FragmentTransaction transaction = fragManager.beginTransaction();
        transaction.replace(R.id.upper_container, (Fragment)viewer);
        transaction.commit();
    }

    public void setInput(Input input) {
        FragmentManager fragManager = getFragmentManager();
        FragmentTransaction transaction = fragManager.beginTransaction();
        transaction.replace(R.id.lower_container, (Fragment)input);
        transaction.commit();
    }
}
```

RPSController

```
public class RPSController implements Controller {
    private MainActivity activity;
    private RPSPlayer computerPlayer;
    private Input input;
    private Viewer viewer;
    private int humanPoints, computerPoints;
    private int humanChoice, computerChoice;

    public RPSController(MainActivity activity, RPSPlayer computerPlayer) {
        this.activity = activity;
        this.computerPlayer = computerPlayer;
        viewer = new ImageViewerFragment();
        input = new ImageInputFragment();
        viewer.setController(this);
        input.setController(this);
        activity.setViewer(viewer);
        activity.setInput(input);
    }
    :
    private void rules(int humanChoice, int computerChoice) {…}
    private void winner() {…}
    private void initGame() {
        humanPoints = computerPoints = 0;
        humanChoice = computerChoice = Controller.EMPTY;
        input.enableChoiceButtons(true);
        updateViewer();
    }
}
```

RPSController

```
public class RPSController implements Controller {
    :
    public void newGame() {
        initGame();
    }

    public void newChoice(int humanChoice) {
        :
    }

    public void updateViewer() {
        viewer.setHumanPoints(humanPoints);
        viewer.setComputerPoints(computerPoints);
        viewer.setHumanChoice(humanChoice);
        viewer.setComputerChoice(computerChoice);
    }

    public void changeUI() {
        if(viewer instanceof TextViewerFragment)
             viewer = new ImageViewerFragment();
        else
            viewer = new TextViewerFragment();
        viewer.setController(this);
        activity.setViewer(viewer);
        // motsvarande för input
    }
}
```

Viewer implementation

```
public class AAAViewerFragment extends Fragment implements Viewer {
    private Controller controller;

    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.textviewer, container, false);
        :
        return view;
    }

    public void onResume() {
        super.onResume();
        controller.updateViewer(); // Uppdatering av nytt fragment
    }

    // Viewer-implementering
    public void setController(Controller controller) {…}
    public void setHumanPoints(int points) {…}
    public void setComputerPoints(int points) {…}
    public void setHumanChoice(int choice) {…}
    public void setComputerChoice(int choice) {…}
    public void humanWinner() {…}
    public void computerWinner() {…}
}
```

Input implementation

```
public class ...InputFragment extends Fragment implements Input {
    private Controller controller;

    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.imageinput, container, false);
        :
        return view;
    }

    private void registerListeners() {
        :
        btnChangeUI.setOnClickListener(new OnClickListener() {
            public void onClick(View view) {
                controller.changeUI();
            }
        });
    }
    // Input-implementering
    public void setController(Controller controller) {…}
    public void enableChoiceButtons(boolean enabled) {…}

    private class ChoiceButtonListener implements OnClickListener {
        :
    }
    :
}
```