TASK GUIDE (B4.07)

A. Objectives.
Student will start to write the code for MediaActivity which contains Video Player, YouTube Player, and ViewFlipper. Student also will learn how to use gesture.

B. Requirements.
Hardware:
- 2 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution
- Intel processor with support for Intel VT-x, Intel EM64T (Intel 64), and Execute Disable (XD) Bit functionality

Software:
- Microsoft Windows 7/8/10 (32-bit or 64-bit)
- JDK 8
- Android Studio IDE (Minimum 3.2) with AndroidX library.

C. Resources.
Documents:
- Guide

Supplement files:

Test code:
- TestB4MultimediaResources071.java

D. Task Description.
Student start to write the code for MediaActivity.

E. Specification.

1. Open "MediaActivity.java" in java folder.

2. Declare all variables that represents all widgets and global variable in activity_main.xml.

```java
public class MainActivity extends AppCompatActivity {

    // Put Here

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

Using this template:

```
private <Type_Data> <variable_name>;
```

All the widgets are:

Type Data	Variable_name	Group
TextView	mediaTitle	Widgets
VideoView	vid	Widgets
Button	prevBtn, nextBtn	Widgets
ImageButton	finishBtn, invertBtn	Widgets
ViewFlipper	flipper	Widgets
GestureDetector	detector	Android class
YouTubePlayer	youPlayer	Android class
String[]	listTitles, listYoutube, listVideo	Array
int[]	listColor	Array
int	currIndex = 0	-
Animation	titleAnimation, leftInAnimation, leftOutAnimation, rightAnimation, rightOutAnimation	Animation
static final String	API_KEY = "AIzaSyBo5KRuoOVO4xkYIVMp6t2c5I22tkKLF4I"	-

3. In the onCreate method, define all <mark>Widgets</mark> variables, which has been declared in point 2, to the related widget id using this template:

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

    <variable_name> = (<Widget_type>)findViewById(R.id.<widget_id>);
    .
    .
    .
}
```

Then assign all **Array** variable with related array resource in String resource or Color resource using this templates:

```
//for String array
    <variable_name> = getResources().getStringArray(R.array.<resource_name>);

//for int array
    <variable_name> = getResources().getIntArray(R.array.<resource_name>);
```

Also assign all **Animation** variable with this code:

```
titleAnimation = AnimationUtils.loadAnimation(this, R.anim.text_fade_in);
leftInAnimation = AnimationUtils.loadAnimation(this, R.anim.slide_in_left);
leftOutAnimation = AnimationUtils.loadAnimation(this, R.anim.slide_out_left);
rightInAnimation = AnimationUtils.loadAnimation(this, R.anim.slide_in_right);
rightOutAnimation = AnimationUtils.loadAnimation(this, R.anim.slide_out_right);
```

4. Now all array variables have data/contents and each data has index. Then we have to create a method to set the style of MediaActivity layout depend on the animal that has been clicked on previous Activity.

   Create a method 'setTitleAndBtn' to set the stryling of layout that has a int parameter (as array index) using this code:

```
private void setTitleAndBtn(int idx) {
        mediaTitle.setBackgroundColor(listColor[idx]);
        mediaTitle.setText(listTitles[idx]);
        mediaTitle.startAnimation(titleAnimation);

        int prevIdx = (idx==0)?listColor.length-1:idx-1;
        int nextIdx = (idx>=listColor.length-1)?0:idx+1;

        prevBtn.setText("<< "+listTitles[prevIdx]);
        prevBtn.setBackgroundColor(listColor[prevIdx]);
        nextBtn.setText(listTitles[nextIdx]+" >>");
        nextBtn.setBackgroundColor(listColor[nextIdx]);
}
```

   This method will set the title text and color related to clicked animal and start the animation on title. Also the prevBtn and nextBtn will be set to prevous and next animal.

The clicked animal name will be sent in extra by MainActivity. In the **onCreate** method we must catch this extra by Intent and get the index of the animal title. With this index we can call 'setTItleAndBtn' method, like below:

```
Intent intent = getIntent();
String title = intent.getStringExtra("TITLE_ANIMAL");
String youtube = intent.getStringExtra("YOUTUBE_LINK");
String video = intent.getStringExtra("VIDEO_ANIMAL");

if (title == null) {
      currIndex=0;
} else {
      currIndex = Arrays.asList(listTitles).indexOf(title);
}
setTitleAndBtn(currIndex);
```

5. The ViewFlipper is a widget that can be filled by some layouts and each layout can be shown as a slide. The ViewFlipper is still empty currently and we have to fill the ViewFlipper with 5 slide which is inflated from 'layout_media.xml'. Each layout contains a VideoView that must be assign by video in raw resource.

```
private void addMedia(ViewFlipper fl) {
   for (int i=0; i<listTitles.length; i++) {
         LayoutInflater inflater = (LayoutInflater)
            this.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
         View view = inflater.inflate(R.layout.layout_media, null);
         fl.addView(view);

         vid = (VideoView) view.findViewById(R.id.videoView);
         vid.setBackgroundColor(listColor[i]);
         vid.setZOrderOnTop(true);
         MediaController m = new MediaController(this);
         m.setAnchorView(vid);
         vid.setMediaController(m);

         Uri u = getMedia(listVideo[i]);
         vid.setVideoURI(u);
   }
}
```

Code for method 'getMedia':

```
private Uri getMedia(String mediaName) {
    return Uri.parse("android.resource://" + getPackageName() +
               "/raw/" + mediaName);
}
```

In the **onCreate** method we can call method 'addMedia' then set the displayed slide to current Index animal:

```
addMedia(flipper);
flipper.setDisplayedChild(currIndex);
```

6. Then we have to set the YouTube fragment to show the video of currentIdx animal. Create a method to initialize the YouTube player like below:

```
private void initializeYoutubePlayer() {
    YouTubePlayerFragment youFragment = (YouTubePlayerFragment)
        getFragmentManager().findFragmentById(R.id.youtubeFrame);
    if (youFragment == null) return;

    youFragment.initialize(API_KEY, new YouTubePlayer.OnInitializedListener() {

    @Override
    public void onInitializationSuccess(YouTubePlayer.Provider provider,
        YouTubePlayer player, boolean wasRestored) {
        if (!wasRestored) {
            youPlayer = player;
            youPlayer.cueVideo(listYoutube[currIndex]);
        }
    }

    @Override
    public void onInitializationFailure(YouTubePlayer.Provider provider,
        YouTubeInitializationResult error) {
        String errorMessage =
            String.forma("There was an error initializing the YoutubePlayer (%1$s)",
                error.toString());
        Toast.makeText(getApplicationContext(), errorMessage,
            Toast.LENGTH_LONG).show();
    } });
}
```

Call this method in 'onCreate' method.

7. Then we will make methods to response when 'prevBtn' and 'nextBtn' are clicked. The buttons is used to change the currentIdx of animal and it will change the displayed video in ViewFlipper and YouTube player, including the layout style.

First we have to create a method 'showPrevFlipper' like below:

```
private void showPrevFlipper(ViewFlipper fl) {
    VideoView v = (VideoView) fl.getCurrentView().findViewById(R.id.videoView);
    v.pause();
    fl.setInAnimation(AnimationUtils.loadAnimation(getApplicationContext(),
        R.anim.slide_in_right));
    fl.setOutAnimation(AnimationUtils.loadAnimation(getApplicationContext(),
        R.anim.slide_out_right));

    fl.showPrevious();
    currIndex = fl.getDisplayedChild();
    setTitleAndBtn(currIndex);

    if (youPlayer != null) {
        youPlayer.cueVideo(listYoutube[currIndex]);
    }
}
```

Create an onClickListener in onCreate method for 'prevBtn'. Call this method 'showPrevFlipper(flipper)' in the onClick.

Then we have to create a method 'showNextFlipper' like below:

```
private void showNextFlipper(ViewFlipper fl) {
    VideoView v = (VideoView) fl.getCurrentView().findViewById(R.id.videoView);
    v.pause();
    fl.setInAnimation(AnimationUtils.loadAnimation(getApplicationContext(),
        R.anim.slide_in_left));
    fl.setOutAnimation(AnimationUtils.loadAnimation(getApplicationContext(),
        R.anim.slide_out_left));

    fl.showNext();
    currIndex = fl.getDisplayedChild();
    setTitleAndBtn(currIndex);

    if (youPlayer != null) {
        youPlayer.cueVideo(listYoutube[currIndex]);
    }
}
```

Create an onClickListener in onCreate method for 'nextBtn'. Call this method 'showNextFlipper(flipper)' in the onClick.

8. We will apply touching gesture on ViewFlipper. We have to create a class 'CustomGestureDetector' like below:

```
class CustomGestureDetector extends GestureDetector.SimpleOnGestureListener {
    //put the methods here…
}
```

The first scenario is swiping the ViewFlipper:

- when user swipes right on ViewFlipper, the content will be changed to previous animal that is meant calling 'showPrevFlipper',
- when user swipes left on ViewFlipper, the content will be changed to next animal that is meant calling 'showNextFlipper'.

To apply this scenario, we have to add an override method 'onFling' in class 'CustomGestureDetector' to detect swiping from user then call related method (prev or next), like below:

```
@Override
public boolean onFling(MotionEvent e1, MotionEvent e2,
    float velocityX, float velocityY) {
    // Swipe left (next)
    if (e1.getX() > e2.getX()) {
        showNextFlipper(flipper);
    }

    // Swipe right (previous)
    if (e1.getX() < e2.getX()) {
        showPrevFlipper(flipper);
    }
    return super.onFling(e1, e2, velocityX, velocityY);
}
```

The second scenario is single touch/tap the ViewFlipper:

- when user taps the ViewFlipper on the left side, the content will be changed to previous animal that is meant calling 'showPrevFlipper',
- when user taps the ViewFlipper on the right side, the content will be changed to next animal that is meant calling 'showNextFlipper'.

To apply this scenario, we have to add an override method 'onDown' in class 'CustomGestureDetector' to detect single touch/tap from user then call related method (prev or next), like below:

```
@Override
public boolean onDown(MotionEvent e) {
    if (e.getX() < flipper.getLeft() + 100) {
        showPrevFlipper(flipper);
    } else if (e.getX() > flipper.getRight() - 100) {
        showNextFlipper(flipper);
    }
    return true;
}
```

You can learn the parameter in this code.
Finally, we have to create an 'onTouchListener' from ViewFlipper in the onCreate method. Apply this code:

```
CustomGestureDetector customGestureDetector = new CustomGestureDetector();
detector = new GestureDetector(this, customGestureDetector);

flipper.setOnTouchListener(new View.OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
    detector.onTouchEvent(event);
        return true;
    }
});
```

9. Create onClickListener for 'finishBtn' in onCreate method to open the MainActivity with transition animation. Put this code in the 'onClick' method:

```
startActivity(new Intent(getApplicationContext(),MainActivity.class));
overridePendingTransition(R.anim.slide_in_right, R.anim.slide_out_right);
```

10. Finally, create onClickListener for 'invertBtn' in onCreate method to open the InvertActivity with transition animation. Put this code in the 'onClick' method:

```
Intent invert = new Intent(getApplicationContext(),InvertActivity.class);
startActivity(invert);
overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
```

First, we have to make a method 'openMediaActivity' with a string parameter (animal name), to open MediaActivity as intent with inserting 'TITLE_ANIMAL' as extra and using transition animation 'slide_in_left' and 'slide_out_left' when change the active activity with 'overridePendingTransition'. The code like below:

```
private void openMediaActivity(String title) {
    Intent media = new Intent(getApplicationContext(),MediaActivity.class);
    media.putExtra("TITLE_ANIMAL", title);
    startActivity(media);
    overridePendingTransition(R.anim.slide_in_left, R.anim.slide_out_left);
}
```

Then, we have to make a method 'openInvertActivity' to open InvertActivity as intent using transition animation 'fade_in' and 'fade_out' when change the active activity with 'overridePendingTransition'. The code like below:

```
private void openInvertActivity() {
    Intent intent = new Intent(getApplicationContext(),InvertActivity.class);
    startActivity(intent);
    overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
}
```

In onCreate method, we have to create an item click listener for the RecyclerView. When invertebrates is clicked, it will call 'openInvertActivity' method, otherwise will call 'openMediaActivity' with related item as input parameter. Set the item click listener for mAdapter like below:

```
mAdapter.setOnItemClickListener(new DataAdapter.OnItemClickListener() {
    @Override
    public void onItemClick(View view, int position) {
        TextView itemTitle = (TextView) view.findViewById(R.id.animalTitle);
        if (itemTitle.getText().toString().equals("Invertebrates")) {
            openInvertActivity();
        } else {
            openMediaActivity(itemTitle.getText().toString());
        }
    }
});
```

The code was complete, next go to testing stage.

F. Testing.

1. Copy "TestB4MultimediaResources071.java" file to "org.aplas.animaltour (test)" folder.

2. Right click on the "TestB4MultimediaResources071.java" file then choose Run. It may take long time to execute.

3. Get the result of your task. If passed you will get green check like below. If the test failed, you would get orange check get the messages and you must check your work again.
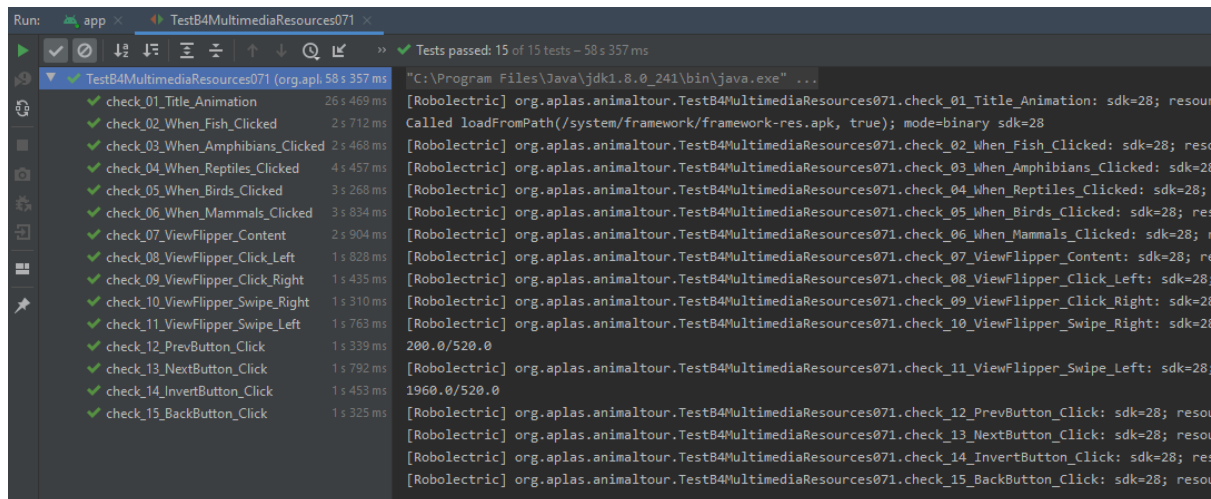
You have to try until get all green checkes and continue to the next task.

You can get the screen display like below when running this Activity. The style of layout depends on your layout design.