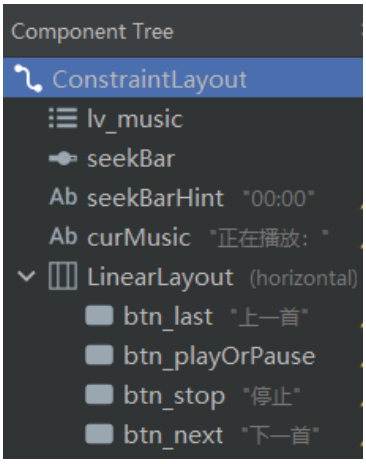
第二次作业：

音乐播放器

1.运行效果展示

2.布局结构



布局由音乐列表(ListView[图中id:lv_music])和播放控件(其他组件)两部分组成

3. 思路分析与部分代码展示

1) 实现全屏效果并修改 ActionBar 样式

```
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
        WindowManager.LayoutParams.FLAG_FULLSCREEN);
getSupportActionBar().setTitle("Music");
```

ActionBar 颜色在 themes 文件设置 colorPrimary 项的值 #3F51B4

2) 显示音乐列表

- 通过 getAssets().list("music")获得音乐名的数组
- 构造 ArrayAdapter 适配器装载音乐信息并与 ListView 绑定

```
public void initListView() {
    listView = (ListView) findViewById(R.id.lv_music);
    music_list = getMusic_list();
    listView.setChoiceMode(AbsListView.CHOICE_MODE_SINGLE);
    adapter = new ArrayAdapter<>( context: MainActivity.this,
            android.R.layout.simple_list_item_1,
            music_list);
    listView.setAdapter(adapter);
    listView.setOnItemClickListener(listViewListener);
}
```

3) 初始化 ExoPlayer 加载 playlist

- 使用 Uri.parse("asset:///path/file")获得 asset/music 下音乐资源的 uri 并建立 mediaItem 媒体资源
- 通过 addMediaItem()将所有音乐资源加载到 player 的播放列表中

```
public void initExoPlayer() {
    player = new ExoPlayer.Builder( context: MainActivity.this).build();
    for (String music : music_list) {
        MediaItem mediaItem = MediaItem.fromUri(Uri.parse("asset:///music/" + music));
        player.addMediaItem(mediaItem);
    }
    player.setRepeatMode(ExoPlayer.REPEAT_MODE_ALL);
    player.addListener(playerListener);
}
```

4) 创建定时任务类

- 定时刷新 SeekBar 进度条

```
class ProgressUpdate extends TimerTask {

    @Override
    public void run() {
        runOnUiThread(() -> {
            long position = player.getContentPosition();
            seekBar.setProgress((int) position);
            seekBarHint.setText(format(position));
        });
    }
}

public String format(long position) {
    SimpleDateFormat sdf = new SimpleDateFormat( pattern: "mm:ss");
    String timeStr = sdf.format(position);
    return timeStr;
}
```

5) 注册 player 的监听事件

◆ 重写 onMediaItemTransition()方法，在音乐资源发生改变或单曲循环模式下当前资源的循环会回调执行

- 更新成员变量 curIndex 为当前回调执行的音乐资源
- 读取当前资源内容长度设置 SeekBar 最大值
- 通过 curIndex 更新“当前播放:”内容
- 更新音乐列表 UI(当前播放音乐背景高亮显示)
- 创建定时任务即时同步 SeekBar 进度为音乐播放进度

```
@Override
public void onMediaItemTransition(@Nullable MediaItem mediaItem, int reason) {
    Player.Listener.super.onMediaItemTransition(mediaItem, reason);
    if (timer != null) {
        timer.cancel();
    }
    if (player.getPlaybackState() != ExoPlayer.STATE_READY) {
        player.prepare();
        prepared = true;
        player.play();
    }
    Log.d( tag: "myflag0",  msg: "State:" + player.getPlaybackState());
    long realDurationMillis = player.getDuration();
    if (realDurationMillis > 0) seekBar.setMax((int) realDurationMillis);
    curIndex = player.getCurrentMediaItemIndex();
    curMusic.setText("正在播放：" + music_list.get(curIndex));
    clearHighlight();
    setCurrentHighlight();
    timer = new Timer();
    timer.schedule(new ProgressUpdate(),  delay: 300,  period: 500);
    Log.d( tag: "myflag0",  msg: "" + getResources().getColor(android.R.color.transparent));
    Log.d( tag: "myflag0",  msg: "" + getResources().getColor(R.color.purple_200));
    Log.d( tag: "myflag",  msg: "first" + realDurationMillis);
}
```

◆ 重写 onPlaybackStateChanged()，在 player 状态改变时回调执行
• 读取当前资源内容长度设置 SeekBar 最大值

```
@Override
public void onPlaybackStateChanged(int playbackState) { //播放列表自然切换时状态不变，不会回调
    Player.Listener.super.onPlaybackStateChanged(playbackState);
    if (!prepared) return;
    long realDurationMillis = player.getDuration();
    if (realDurationMillis > 0) seekBar.setMax((int) realDurationMillis);
    Log.d( tag: "myflag",  msg: "second" + realDurationMillis);
}
```

✧ SeekBar 的最大值在两个回调都做了设置，原因见实验总结

6）注册 ListView 单击监听器实现音乐选择

◆ 通过 seekTo(int mediaItemIndex, long positionMs)实现播放列表音乐跳转

```
AdapterView.OnItemClickListener listViewListener = new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        if (!prepared) {
            player.prepare();
            prepared = true;
        }
        curIndex = position;
        player.seekTo(position, C.POSITION_UNSET);
    }
};
```

7）注册 SeekBar 监听器实现用户可调整音乐进度

```
SeekBar.OnSeekBarChangeListener seekBarListener = new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        if (prepared && fromUser) {
            player.seekTo(progress);
        }
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
        seekBarHint.setText(format(seekBar.getProgress()));
    }
};
```

8）注册按钮点击事件监听器

```
@Override
public void onClick(View v) {
    if (!prepared) return;
    switch (v.getId()) {
        case R.id.btn_playOrPause:
            if (player.isPlaying()) {
                player.pause();
                timer.cancel();
                timer = new Timer();
            } else {
                player.play();
                timer = new Timer();
                timer.schedule(new ProgressUpdate(), delay: 300, period: 500);
            }
            break;
        case R.id.btn_next:
            player.seekToNextMediaItem();
            break;
        case R.id.btn_last:
            player.seekToPreviousMediaItem();
            break;
        case R.id.btn_stop:
            player.stop();
            if (timer != null) timer.cancel();
            seekBar.setProgress(0);
            curMusic.setText("正在播放：");
            seekBarHint.setText("00:00");
            seekBar.setMax(0);
            clearHighlight();
            prepared = false;
            break;
    }
```

4.实验总结

在实验中两次获取音乐长度设置 SeekBar 最大值，在两个回调都做设定的原因：首先不能单独在 onMediaItemTransition()设置是因为该回调在切歌时只调一次，并且可能在未完成缓存期间调用,那么就会得到-9223372036854775807 的无效歌曲长度，而 onPlaybackStateChanged()该回调在缓存前调用，状态由缓存转变播放也会触发回调，所以无论如何最后一次都会得到正确值并覆盖无效值。其次不能单独在 onPlaybackStateChanged()设置是因为，有种特殊情况是当播放列表播放完当前歌曲自然切到下一首歌曲时,播放状态保持 ready 不会改变，所以该回调并不会执行，那么这种情况下 seekbar 最大长度不能得到更新，但回调 1 是只要资源改变都会调用，同时也正因为处于这种保证状态 ready 的情况下，所以回调 1 在自然切换情景下获得的值也必然是有效的