实验三 事件机制

一、实验目的

1、了解 Android 事件处理机制；
2、掌握基于监听的事件处理机制和基于回调的事件处理机制；
3、使用 Configuration 类获取系统的配置信息；
4、熟练掌握 Handler 的消息传递机制。
5、掌握 ButterKnife 开源框架
6、设计一个登录界面

二、实验环境

1.JDK-15.0.2 以上
2.Android Studio 4.1 以上
3.Android 10.0（API level 29）
4.墨刀或 Balsamiq Mockups 3 原型设计工具

三、实验内容

1、基于不同事件处理机制，实现如下效果，在第二个TextView中显示当前处理点击事件的方法

2、编写不同按钮监听实现方式代码。

3、点击最后一个按钮，通过Configuration类来获取系统配置信息，补充程序，使程序正常运行。

ConfigurationTest.java 代码如下：

```
package cn.edu.hqu.cst.zwl.ex3demo;


import

androidx.appcompat.app.AppCompatActivity;
```

```
import android.content.res.Configuration;
import android.os.Bundle;
import android.widget.TextView;

public class SystemInfoActivity extends
AppCompatActivity {
    TextView
tvShowOri,tvShowNav,tvShowMnc,tvShowTouch;
    @Override
    protected void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);

setContentView(R.layout.activity_system_info);

tvShowOri=findViewById(R.id.tv_system_info_ac
tivity_ori);

tvShowNav=findViewById(R.id.tv_system_info_ac
tivity_nav);

tvShowTouch=findViewById(R.id.tv_system_info_
```

```
activity_touch);

tvShowMnc=findViewById(R.id.tv_system_info_activity_mnc);
        Configuration cfg=getResources().getConfiguration();;
        String screen = cfg.orientation == Configuration.ORIENTATION_LANDSCAPE ?
                "横向屏幕": "竖向屏幕";
        String mncCode = cfg.mnc + "";
        String naviName = cfg.orientation == Configuration.NAVIGATION_NONAV ? "没有方向控制": cfg.orientation == Configuration.NAVIGATION_WHEEL ? "滚轮控制方向": cfg.orientation == Configuration.NAVIGATION_DPAD ? "方向键控制方向": "轨迹球控制方向";
        tvShowNav.setText(naviName);
        String touchName = cfg.touchscreen == Configuration.TOUCHSCREEN_NOTOUCH ? "无触摸屏": cfg.touchscreen == Configuration.TOUCHSCREEN_STYLUS ? "触摸笔式触
```

```
摸屏": "接受手指的触摸屏";
        tvShowOri.setText(screen);
        tvShowMnc.setText(mncCode);
        tvShowTouch.setText(touchName);
    }
}
```

4、点击最后一个按钮，实现进度条模拟过程，掌握Handler的使用，完成程序。

ProgressDialogTest.java 代码如下：

```
import android.app.Activity;
import android.app.Dialog;
import android.app.ProgressDialog;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
public class ProgressDialogTest extends Activity {
    // 该程序模拟填充长度为 100 的数组
```

```
private int[] data = new int[100];
int hasData = 0;
// 定义进度对话框的标识
final int PROGRESS_DIALOG = 0x112;
// 记录进度对话框的完成百分比
int progressStatus = 0;
ProgressDialog pd;
// 定义一个负责更新的进度的 Handler
Handler handler;
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Button execBn = (Button) findViewById(R.id.exec);
    execBn.setOnClickListener(new OnClickListener() {
        public void onClick(View source) {
            showDialog(PROGRESS_DIALOG);
        }
    });
    //Handler 消息处理，请补全代码，是多行。

}
@Override
public Dialog onCreateDialog(int id, Bundle status) {
    System.out.println("------create------");
    switch (id) {
    case PROGRESS_DIALOG:
        // 创建进度对话框
        pd = new ProgressDialog(this);
        pd.setMax(100);
        // 设置对话框的标题
        pd.setTitle("任务完成百分比");
        // 设置对话框 显示的内容
        pd.setMessage("耗时任务的完成百分比");
        // 设置对话框不能用“取消”按钮关闭
        pd.setCancelable(false);
        // 设置对话框的进度条风格
        pd.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
        // 设置对话框的进度条是否显示进度
        pd.setIndeterminate(false);
        break;
    }
    return pd;
}
// 该方法将在 onCreateDialog 方法调用之后被回调
```

```
@Override
public void onPrepareDialog(int id, Dialog dialog) {
    System.out.println("------prepare------");
    super.onPrepareDialog(id, dialog);
    switch (id) {
    case PROGRESS_DIALOG:
        // 对话框进度清零
        pd.incrementProgressBy(-pd.getProgress());
        new Thread() {
            public void run() {
                while (progressStatus < 100) {
                    // 获取耗时操作的完成百分比
                    progressStatus = doWork();
                    // 发送消息到 Handler，请补全代码

                }
                // 如果任务已经完成
                if (progressStatus >= 100) {
                    // 关闭对话框
                    pd.dismiss();
                }
            }
        }.start();
        break;
    }
}
// 模拟一个耗时的操作。
public int doWork() {
    // 为数组元素赋值
    data[hasData++] = (int) (Math.random() * 100);
    try {
        Thread.sleep(100);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    return hasData;
}
```

四、拓展与提升

1.学习 databinding 和 viewbinding 库，了解其在 MVVP 中的作用

五、问题思考

1.Android 如何避免 UI“死机”影响用户体验？
2.在那些场景下会出现耗时操作，如何避免影响？