2023 안드로이드 Android Application

대전대 온디바이스 AI 과정
하성호

Android Thread

목표

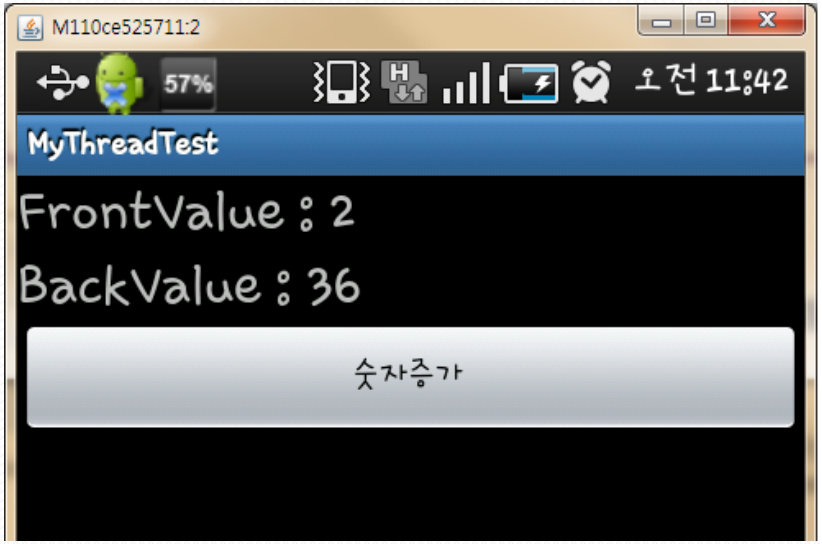
- 안드로이드 멀티스레드
- 스레드를 생성하는 방법
- 핸들러를 통한 스레드 간의 통신방법
- 작업스케줄링과 ANR을 회피하는 방법

스레드(Thread)

- Thread
  - 멀티 스레드 : 한번에 여러개의 작업을 동시에 실행
  - 리눅스는 멀티 스레드 지원 운영체제
  - Java언어 또한 가상머신 차원의 멀티 스레드 지원
  - 스레드는 고유의 스택을 가지며 메인 스레드와는 별도의 CPU 시간을 할당 받아 실행
  - 스레드의 작업은 run()에서 수행 run메서드가 스레드 진입점이면서 메인 스레드

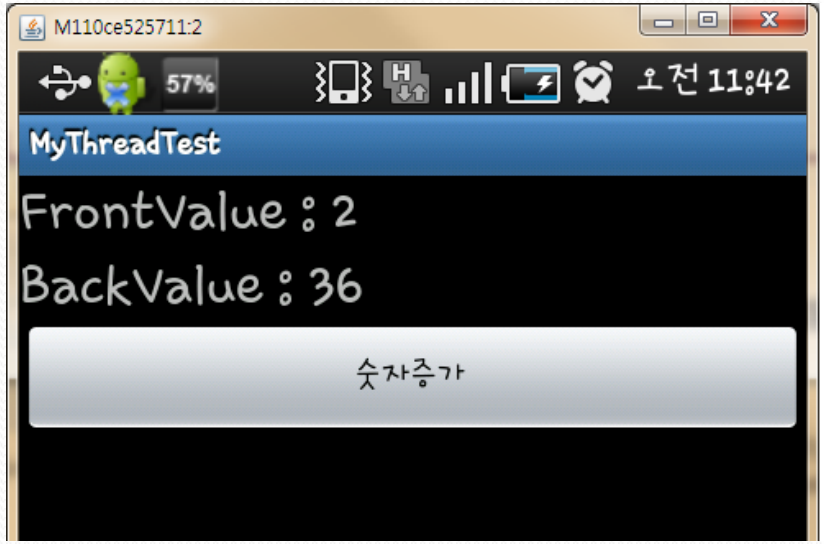
[실습] 스레드 테스트

Project Name
: MyThreadTest
Application Name
: MyThreadTest
Package Name
: com.hn.mythreadtest
Activity Name
: MyThreadTest



main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
>
    <TextView
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:text="TextView"
        android:textSize="25px"
        android:id="@+id/frontvalue"
    ></TextView>
    <TextView
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:text="TextView"
        android:textSize="25px"
        android:id="@+id/backvalue"
    ></TextView>
    <Button
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:text="숫자증가 "
        android:id="@+id/button"
    ></Button>
</LinearLayout>
```



MyThreadTest.java

```
public class MyThreadTest extends Activity {

    int frontValue = 0;
    int backValue = 0;
    TextView frontText;
    TextView backText;
```

onCreate()

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

        frontText = (TextView)this.findViewById(R.id.frontvalue);
        backText = (TextView)this.findViewById(R.id.backvalue);
        Button button = (Button)this.findViewById(R.id.button);

        button.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                frontValue++;
                frontText.setText("FrontValue : " + frontValue);
                backText.setText("BackValue : " + backValue);
            }
        });
        BackThread thread = new BackThread();
        thread.setDaemon(true);
        thread.start();
    }//end of onCreate
```

BackThread class

```
class BackThread extends Thread{
        public void run(){
            while(true){
                backValue++;
                try{
                    Thread.sleep(1000);
                }catch(InterruptedException e) {}
            }
        }
    }
}
```

Runnable 형태

```
class BackThread implements Runnable{
        public void run(){
            while(true){
                backValue++;
    // backText.setText("BackVaue : "+backValue); //에러
                try{
                    Thread.sleep(1000);
                }catch(InterruptedException e) {}
            }
        }
    }
}
```

핸들러(Handler)

- 스레드간의 통신을 할 수 있는 장치
- 스레드간의 메시지나 Runnable 객체를 통해 메시지를 주고 받는 장치
- 핸들러는 항상 하나의 스레드와 관련을 맺고 자신을 생성하는 스레드에 부착되며 그 스레드의 메시지 큐를 통해 다른 스레드와 통신함
- 메시지가 도착하면 핸들러 메서드가 호출됨
  - public void handlerMessage(Message msg)
  - 인수로 Message 객체를 전달받는데 msg는 스레드간의 통신 내용을 저장하는 객체
  - 추가 정보도 받기 위해 여러 필드를 가짐 (e.g> arg1, arg2)

핸들러

필드	설명
int what	메시지 ID
int arg1	추가정보 1
int arg2	추가정보 2
Object obj	임의의 객체로 추가정보 제공
Messenger replyTo	메시지에 대한 응답을 받은 객체를 지정

- boolean Handler.sendEmptyMessage(int what)
- boolean Handler.sendMessage(Message msg)
- boolean sendMessageAtFrontOfQueue(Message msg)

핸들러

- sendEmptyMessage는 메시지 ID에 해당하는 what 값만 전달
- 메시지를 만들어 보낼 땐 sendMessage 사용

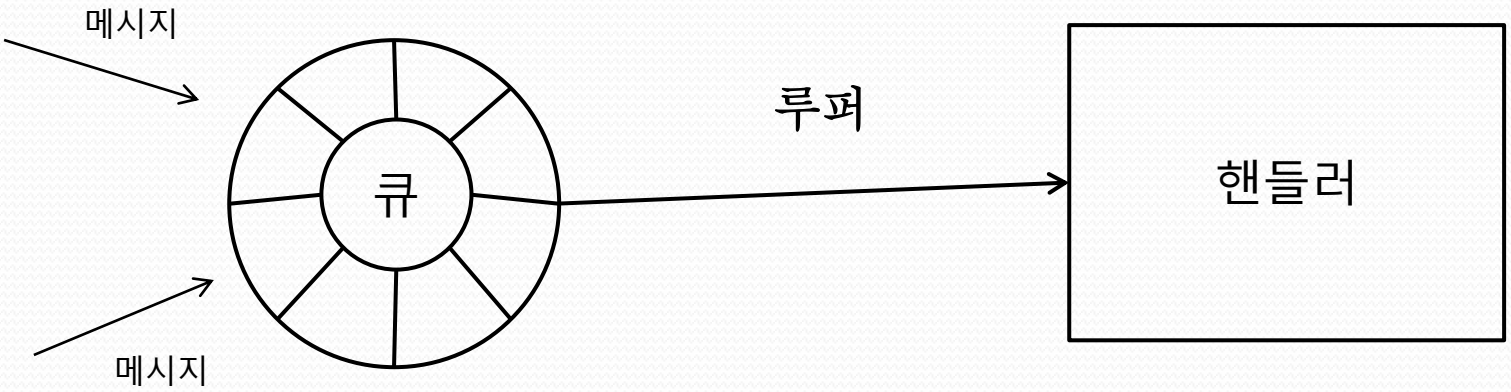
핸들러 처리

```
class BackThread extends Thread{
        public void run(){
            while(true){
                backValue++;
                myHandler.sendEmptyMessage(0);
                try{
                    Thread.sleep(1000);
                }catch(InterruptedException e) {}
            }
        }
    }
    Handler myHandler = new Handler(){
        @Override
        public void handleMessage(Message msg) {
            if(msg.what == 0){
                backText.setText("BackValue : " + backValue);
            }
        }

    };
```

Looper

- 메시지큐(Message) : 메시지가 저장되는 장소
- 메시지나 Runnable 객체는 일단 큐에 저장되고 들어온 순서대로 순차적으로 처리됨
- 루퍼(Looper) : 메시지 큐에서 메시지를 꺼내어 핸들러로 전달
- 루퍼는 무한히 실행되는 메시지 루프를 통해 큐에 메시지가 들어오는지 감시하며 들어온 메시지를 처리할 핸들러를 찾아 handleMessage 메서드를 호출



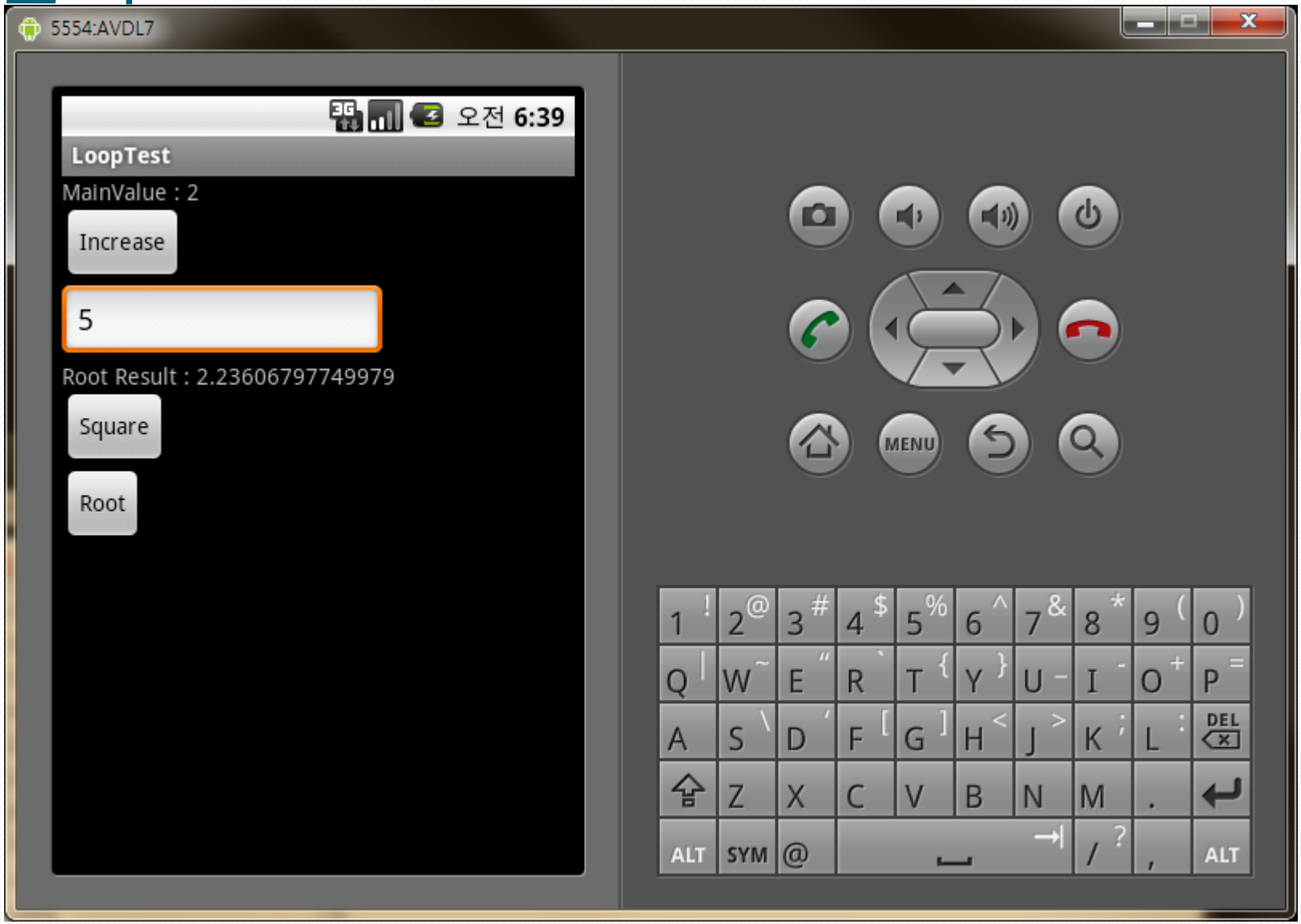
Looper

- UI를 관리하는 메인 스레드는 기본적으로 루퍼를 가짐
- 계산을 주로 수행하는 작업 스레드는 기본적으로 루퍼를 가지지 않으며 run 메서드의 코드만 실행하고 종료됨
- 작업 스레드는 디폴트 루퍼가 없으므로 루퍼를 직접 생성하고 실행시켜야 메시지를 받을 수 있는데 이때 루퍼의 다음 메서드를 호출함
  - static void prepare()
  - static void loop()
  - void quit()
- Prepare는 현재 스레드를 위한 루퍼를 준비함
- loop 메서드는 큐에서 메시지를 꺼내 핸들러로 전달하는 loop를 실행하고, 루프는 quit 메서드가 호출되어 루프를 종료할 때까지 무한 반복됨

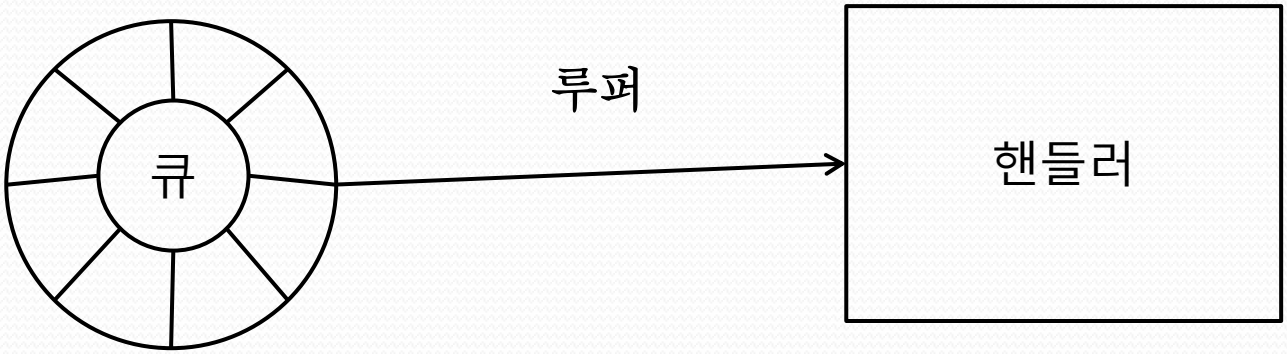
Looper

- Loop는 스레드별로 하나씩 생성되며, 관련된 스레드와 루퍼를 구하는 메서드는 다음과 같음
  - Thread getThread()
  - static Looper getMainLooper()
  - static Looper myLooper()
- getThread 메서드는 루퍼와 연결된 스레드를 구함
- getMainLooper는 응용 프로그램 주 스레드의 루퍼를 구함
- myLooper는 현재 스레드의 루퍼를 구하며, 모든 스레드가 루퍼를 가지는 것은 아니므로 null이 리턴될 수 있음

결과


5554:AVDL7
오전 6:39
LoopTest
MainValue : 2
Increase
5
Root Result : 2.23606797749979
Square
Root
MENU

Looper



Main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/mainvalue"
        android:text="MainValue : "
    ></TextView>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Increase"
        android:id="@+id/increase"
    ></Button>
```

Main.xml(계속)

```
<EditText
        android:layout_height="wrap_content"
        android:id="@+id/number"
        android:layout_width="200px"
    ></EditText>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/backvalue"
        android:text="BackValue :"
    ></TextView>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Square"
        android:id="@+id/square"
    ></Button>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Root"
        android:id="@+id/root"
    ></Button>

</LinearLayout>
```

LoopTest.java(선언부)

```
package com.hn.myloopertest;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.os.Message;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
```

LoopTest.java(계속)

```
public class MyLooperTest extends Activity {
    int mMainValue=0;
    TextView mMainText;
    TextView mBackText;
    EditText mNumEdit;
    CalcThread mThread;
```

onCreate()

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

        mMainText = (TextView)this.findViewById(R.id.mainvalue);
        mBackText = (TextView)this.findViewById(R.id.backvalue);
        mNumEdit = (EditText)this.findViewById(R.id.number);
```

버튼1

```
Button btn = (Button)this.findViewById(R.id.increase);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                mMainValue++;
                mMainText.setText("MainValue : " + mMainValue);
            }
        });
```

버튼2

```
btn = (Button)this.findViewById(R.id.square);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                Message msg = Message.obtain();
                msg.what = 0;
                msg.arg1 =
                Integer.parseInt(mNumEdit.getText().toString());
                mThread.mBackHandler.sendMessage(msg);
            }
        });
```

버튼3

```
btn = (Button)this.findViewById(R.id.root);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                Message msg = Message.obtain();
                msg.what = 1;
                msg.arg1 =
                 Integer.parseInt(mNumEdit.getText().toString());
                mThread.mBackHandler.sendMessage(msg);
            }
        });
 mThread = new CalcThread(mHandler);
        mThread.setDaemon(true);
        mThread.start();
}//end of onCreate
```

Handler

```
Handler mHandler = new Handler(){
        @Override
        public void handleMessage(Message msg) {
            // TODO Auto-generated method stub
            switch(msg.what){
            case 0:
                mBackText.setText("Square Result : " + msg.arg1);
                break;
            case 1:
                mBackText.setText("Root Result : " +
                        ((Double)msg.obj).doubleValue());
                break;
            }
        }
    };
```

CalcThread.java

```
class CalcThread extends Thread{
    Handler mMainHandler;
    public CalcThread(Handler handler) {
        //super();
        mMainHandler = handler;
    }
    @Override
    public void run() {
        // TODO Auto-generated method stub
        Looper.prepare();
        Looper.loop();
    }
```

Handler

```
public Handler mBackHandler = new Handler(){
        @Override
        public void handleMessage(Message msg) {
          Message retmsg = Message.obtain();
            switch(msg.what){
            case 0:
                try{
                    Thread.sleep(200); }catch(InterruptedException e){;}
                retmsg.what = 0;
                retmsg.arg1 = msg.arg1 * msg.arg1;
                break;
            case 1:
                try{ Thread.sleep(200);
                }catch(InterruptedException e){;}
                retmsg.what = 1;
                retmsg.obj = new Double(Math.sqrt((double)msg.arg1));
                break;
            }
            mMainHandler.sendMessage(retmsg);
        }
    };
```

작업스케줄링

Project Name
: WorkThread1
Application Name
: WorkThread1
Package Name
: com.hn.workthread1
Activity Name
: WorkThread1

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="업로드를 시작하려면 다음 버튼을 누르세요." />
    <Button
        android:id="@+id/upload"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Upload" />
</LinearLayout>
```

WorkThread

```
public class WorkThread1 extends Activity
                implements View.OnClickListener {
    Button mUpload;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        mUpload = (Button)findViewById(R.id.upload);
        mUpload.setOnClickListener(this);
    }
```

WorkThread

```
@Override
    public void onClick(View v) {
        switch (v.getId()) {
        case R.id.upload:
            new AlertDialog.Builder(WorkThread1.this)
            .setTitle("질문")
            .setMessage("업로드 하시겠습니까?")
            .setPositiveButton("예", new DialogInterface.OnClickListener()
  {
                public void onClick(DialogInterface dialog,
                 int whichButton) {
                    doUpload();
                }
            })
            .setNegativeButton("아니오", null)
            .show();
            break;
        }
    }
```

WorkThread

```
void doUpload() {
        for (int i = 0; i < 30; i++) {
          try { Thread.sleep(100); }
          catch (InterruptedException e) {;}
        }
     Toast.makeText(this, "업로드를 완료했습니다.", 0).show();
    }
}
```

수정

```
public void onClick(View v) {
        switch (v.getId()) {
        case R.id.upload:
            new AlertDialog.Builder(WorkThread1.this)
            .setTitle("질문")
            .setMessage("업로드 하시겠습니까?")
            .setPositiveButton("예", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int whichButton) {
                    //doUpload();
                    mHandler.sendEmptyMessageDelayed(0, 10);
                }
            })
            .setNegativeButton("아니오", null)
            .show();
            break;
        }
    }
```

핸들러 추가

```
Handler mHandler = new Handler(){
        @Override
        public void handleMessage(Message msg) {
            if(msg.what == 0){
                doUpload();
            }
        }
    };
```

Runnable 객체

```
mHandler.postDelayed(new Runnable()
    {
        public void run(){doUpload();}
    }, 10);
```

- `Handler mHandler = new Handler();`

postDelayed

```
mUpload.postDelayed(new Runnable()
    {
            public void
  run(){doUpload();}
    }, 10);
```