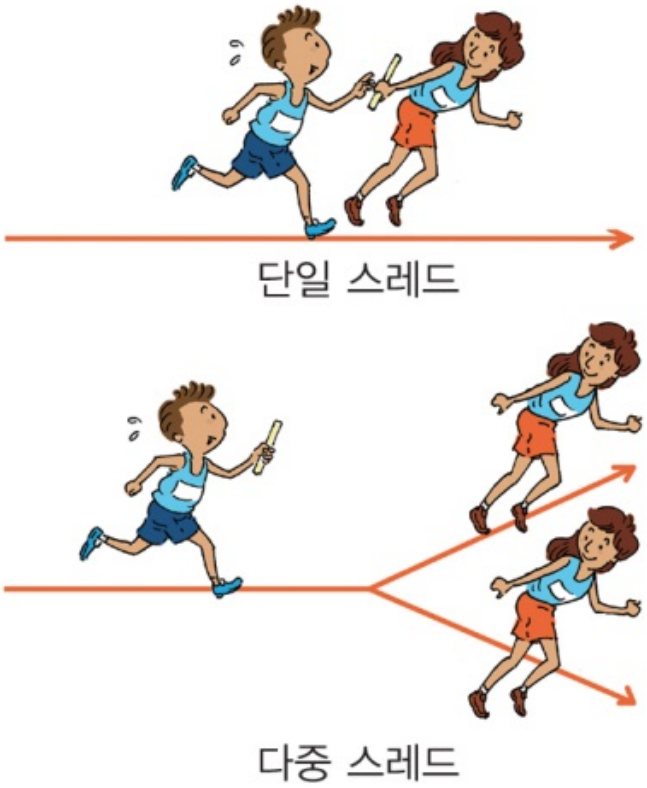
CHAP 13. 프로세스와 스레드

다중 스레딩

- 애플리케이션의 실행하는 하나의 흐름

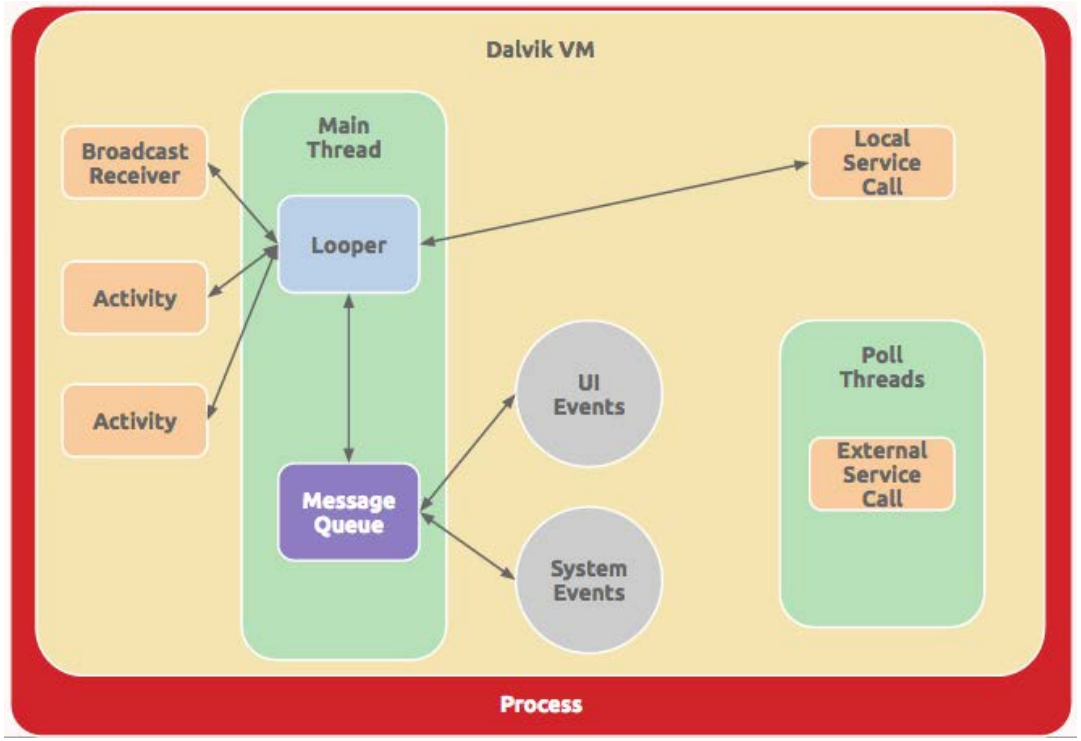


안드로이드에서의 프로세스와 스레드

- 애플리케이션이 시작되면 안드로이드 시스템은 새로운 리눅스 프로세스를 생성.
- 기본적으로 애플리케이션 안의 모든 컴포넌트들은 동일 프로세스 내의 동일한 스레드로 실행.
- 이 기본적인 스레드를 **메인 스레드(main thread 또는 UI thread)**라고 부른다.

메인 스레드

- 메인 스레드는 사용자 인터페이스 위젯에게 이벤트를 전달하거나 화면을 그리는 작업을 담당
- **UI 스레드(user interface thread)**라고도 불린다.



작업 스레드

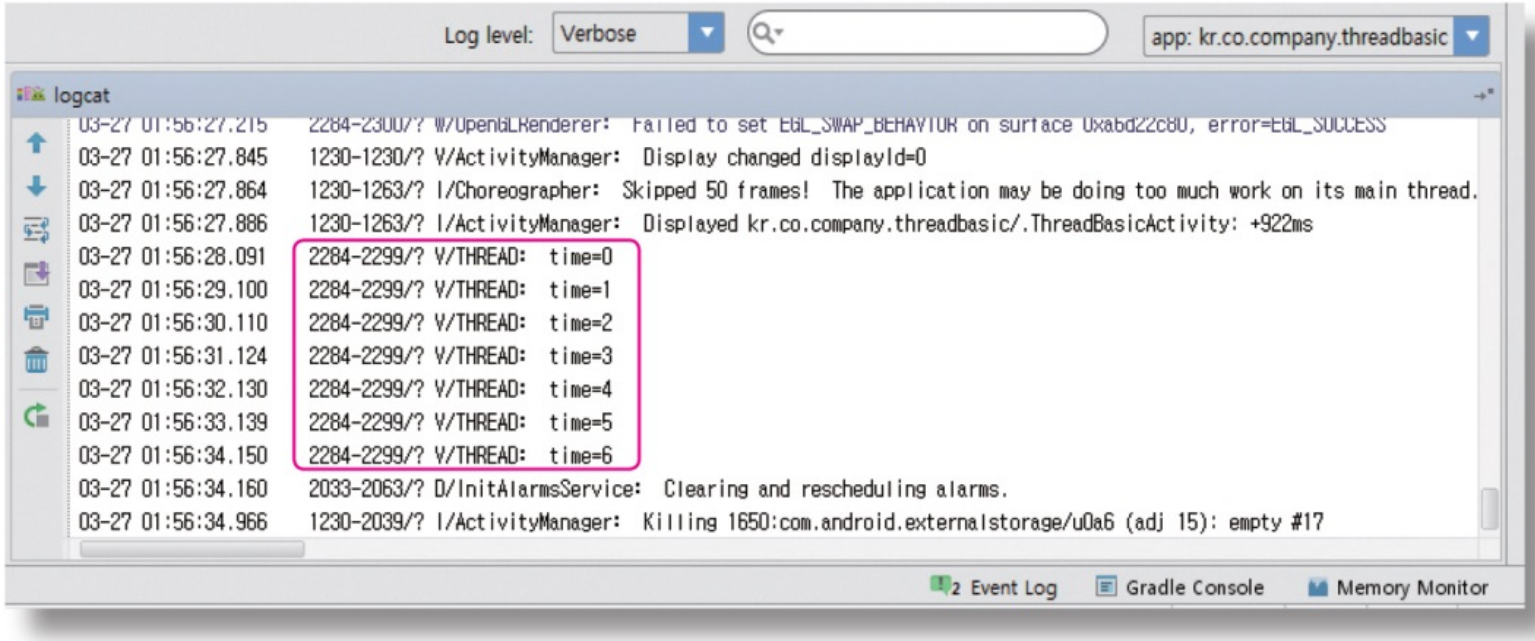
- UI스레드 외의 별도로 생성되는 모든 스레드
- **배경 스레드**(" *background* " thread)라고도 함.
- 스레드 생성
  - Thread 클래스를 상속받아 새로운 스레드를 정의 후에 생성
  - Runnable 인터페이스를 구현한 후에 Thread 객체에 전달

Thread
상속방법

```
public class MainActivity extends AppCompatActivity {

    WorkerThread w;
    boolean running = true;

    class WorkerThread extends Thread {
        public void run() {
            int i = 0;
            for (i = 0; i < 20 && running; i++) {
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                }
                Log.v("THREAD", "time=" + i);
            }
        }
    }
```

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

    @Override
    public void onStart() {
        super.onStart();
        w = new WorkerThread();
        w.start();
        running = true;
    }

    @Override
    public void onStop() {
        super.onStop();
        running = false;
    }
}
```

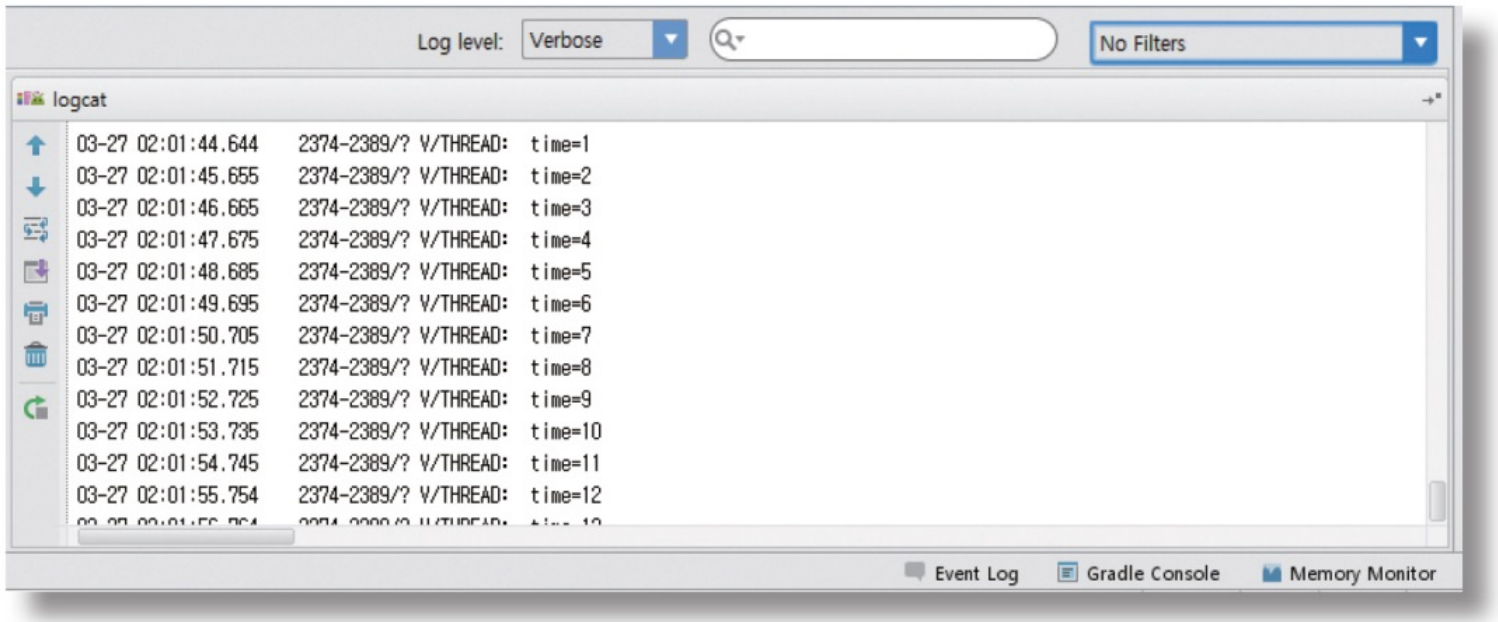
실행 결과


Log level: Verbose
app: kr.co.company.threadbasic
logcat
03-27 01:56:27.215 2284-2300/? W/OpenGLRenderer: Failed to set EGL_SWAP_BEHAVIOR on surface 0xabd22c80, error=EGL_SUCCESS
03-27 01:56:27.845 1230-1230/? V/ActivityManager: Display changed displayId=0
03-27 01:56:27.864 1230-1263/? I/Choreographer: Skipped 50 frames! The application may be doing too much work on its main thread.
03-27 01:56:27.886 1230-1263/? I/ActivityManager: Displayed kr.co.company.threadbasic/.ThreadBasicActivity: +922ms
03-27 01:56:28.091 2284-2299/? V/THREAD: time=0
03-27 01:56:29.100 2284-2299/? V/THREAD: time=1
03-27 01:56:30.110 2284-2299/? V/THREAD: time=2
03-27 01:56:31.124 2284-2299/? V/THREAD: time=3
03-27 01:56:32.130 2284-2299/? V/THREAD: time=4
03-27 01:56:33.139 2284-2299/? V/THREAD: time=5
03-27 01:56:34.150 2284-2299/? V/THREAD: time=6
03-27 01:56:34.160 2033-2063/? D/InitAlarmsService: Clearing and rescheduling alarms.
03-27 01:56:34.966 1230-2039/? I/ActivityManager: Killing 1650:com.android.externalstorage/u0a6 (adj 15): empty #17
2 Event Log
Gradle Console
Memory Monitor

Runnable 인터페이스를 구현 방법

```
@Override
public void onStart() {
    super.onStart();
    w = new Thread( new Runnable() {
        public void run() {
            int i = 0;
            for (i = 0; i < 20 && running; i++) {
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                }
                Log.v("THREAD", "time=" + i);
            }
        }
    });
    w.start();
    running = true;
}
```

실행결과


Log level:
Verbose
No Filters
logcat
03-27 02:01:44.644 2374-2389/? V/THREAD: time=1
03-27 02:01:45.655 2374-2389/? V/THREAD: time=2
03-27 02:01:46.665 2374-2389/? V/THREAD: time=3
03-27 02:01:47.675 2374-2389/? V/THREAD: time=4
03-27 02:01:48.685 2374-2389/? V/THREAD: time=5
03-27 02:01:49.695 2374-2389/? V/THREAD: time=6
03-27 02:01:50.705 2374-2389/? V/THREAD: time=7
03-27 02:01:51.715 2374-2389/? V/THREAD: time=8
03-27 02:01:52.725 2374-2389/? V/THREAD: time=9
03-27 02:01:53.735 2374-2389/? V/THREAD: time=10
03-27 02:01:54.745 2374-2389/? V/THREAD: time=11
03-27 02:01:55.754 2374-2389/? V/THREAD: time=12
Event Log
Gradle Console
Memory Monitor

안드로이드의 단일 스레드 모델 원칙

- 안드로이드 시스템은 애플리케이션 구성 요소의 각 인스턴스에 대해 별도의 스레드를 생성하지 않고, 기본적으로 단일의 UI 스레드에서 실행.
- 원칙
  1. UI 스레드는 블록시키면 안된다.

     네트워크 액세스, 데이터베이스 쿼리, 동영상 인코딩 등.

     ANR(application not responding) 상황 발생 (5초 이상 반응하지 않는 경우)
  2. UI 스레드 외부에서 안드로이드 UI 툴킷을 조작하면 안된다.

     UI툴킷은 thread-safe하지 않으므로 다른 스레드에서 UI를 조작하면 UI 동작에 오류가 발생할 수 있음.

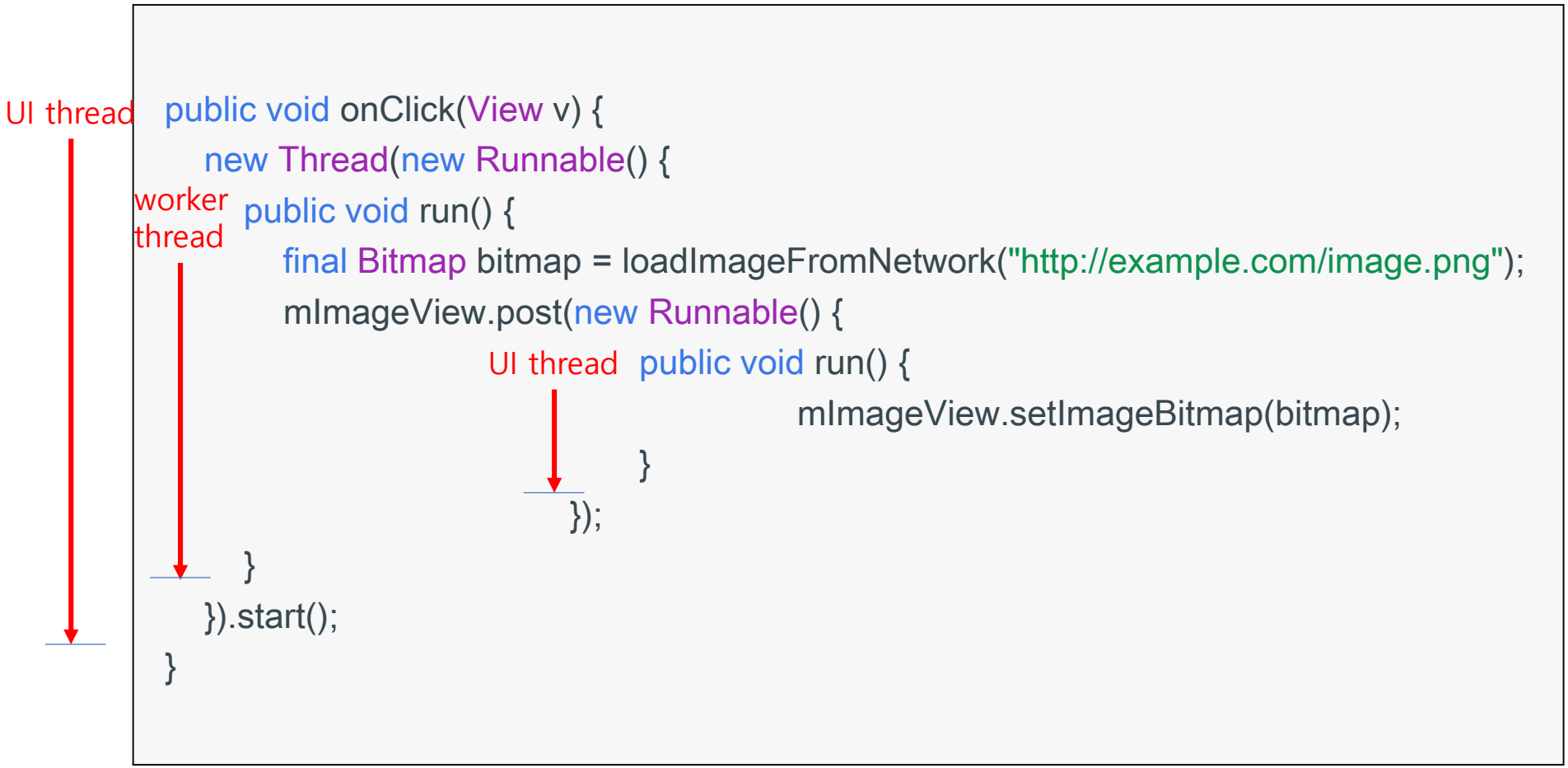
주의할 점

- 스레드에서 직접적으로 사용자 인터페이스 위젯을 변경하면 안 된다.

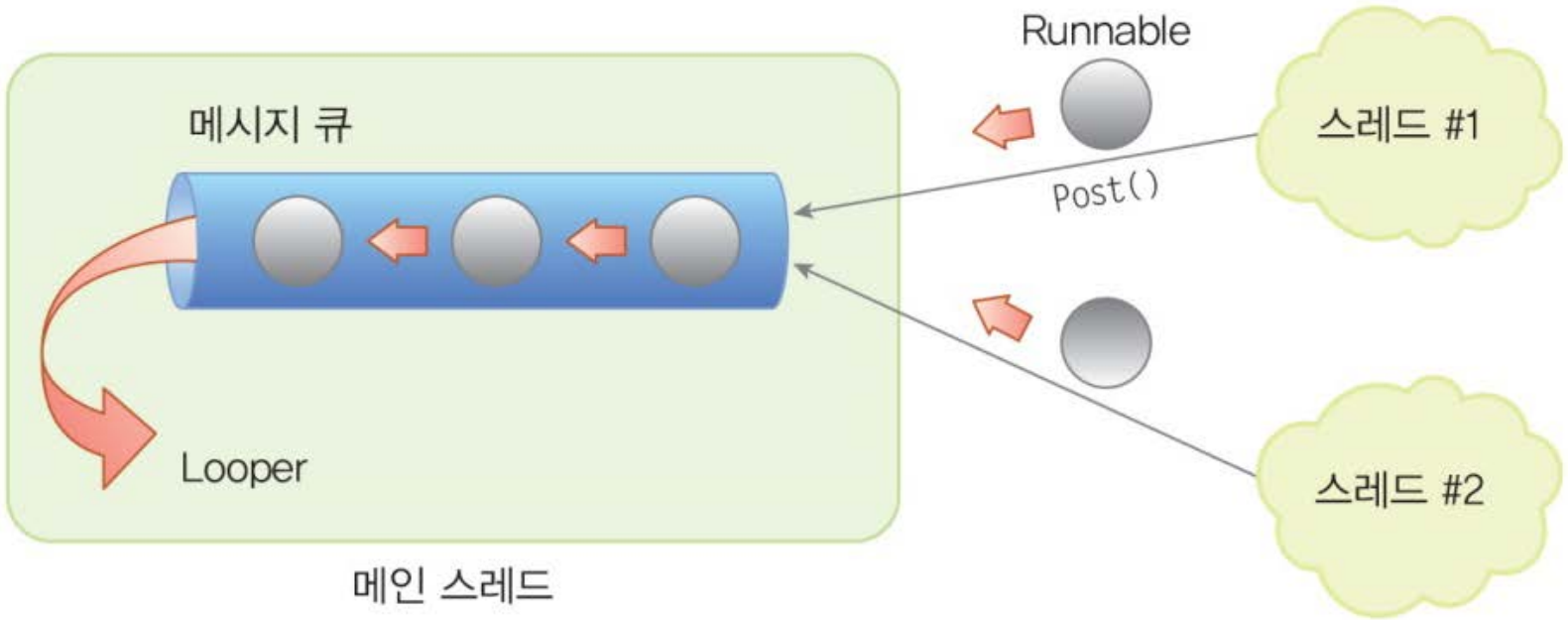
해결할 수 있는 3가지 방법

- **View.post(Runnable)**
- **View.postDelayed(Runnable, long)**
- **Activity.runOnUiThread(Runnable)**

View.post(Runnable) 사용

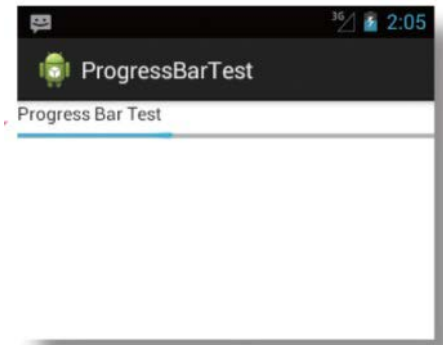


post() 메소드



사용자 인터페이스 작성

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Progress Bar Test" />

    <ProgressBar
        android:id="@+id/progress_bar"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" >
    </ProgressBar>

</LinearLayout>
```



여기서 프로그레시브 바의 속성이 style="?android:attr/progressBarStyleHorizontal"과 같이 설정된 것에 주의하라. 이것은 안드로이드에서 미리 준비된 수평 막대 형태의 스타일로 설정하는 문장이다.

액티비티 작성

ProgressBarTestActivity.java

```
package kr.co.company.progressbartest;
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.

public class ProgressBarTestActivity extends ActionBarActivity {
    private static final int PROGRESS = 0x1;
    private ProgressBar mProgress;
    private int mProgressStatus = 0;
    int i = 0;

    protected void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);
        mProgress = (ProgressBar) findViewById(R.id.progress_bar);

        //
        new Thread(new Runnable() {
            public void run() {
```

작업 스레드가 무명 클래스로 정의되었다.

```
                while (mProgressStatus < 100) {
                    try {
                        Thread.sleep(1000);
                    } catch (InterruptedException e) {
                    }
                    mProgressStatus = i++;

                    //
                    mProgress.post(new Runnable() {
                        public void run() {
                            mProgress.setProgress(mProgressStatus);
                        }
                    });
                }
            }
        }).start();
    }
}
```

네를 업데이트하는 러너블 객체를 전송한다.

Activity.runOnUiThread

- View.post()와 유사
- runOnUiThread 코드

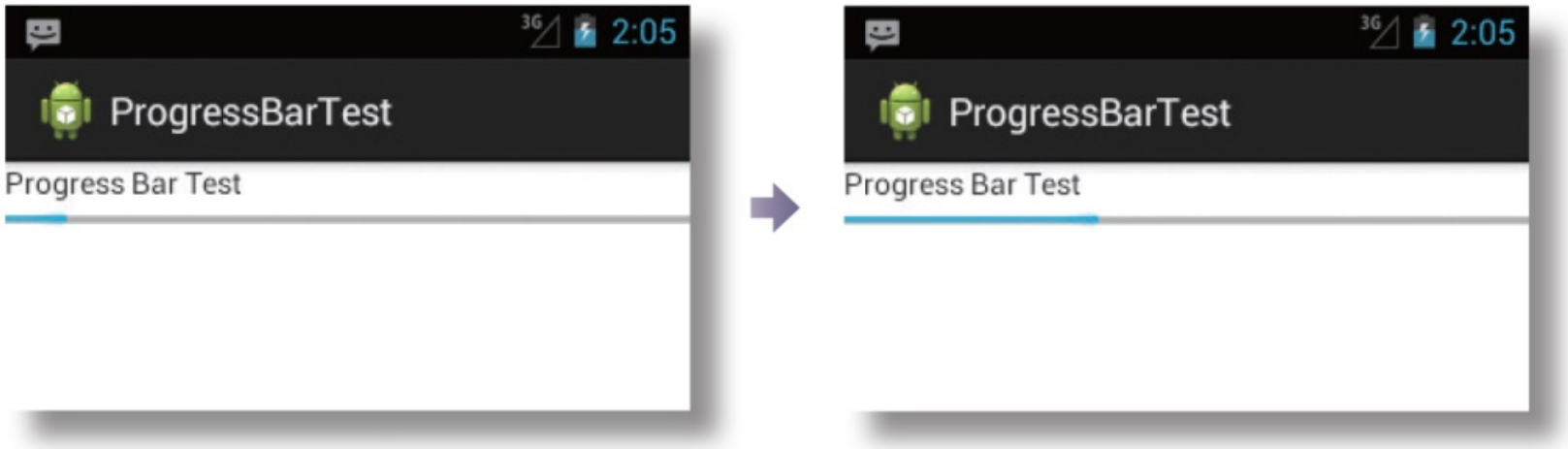
Activity:

```
public final void runOnUiThread(Runnable action) {
    if (Thread.currentThread() != mUiThread)
        mHandler.post(action);
    else
        action.run();
}
```

- 사용법

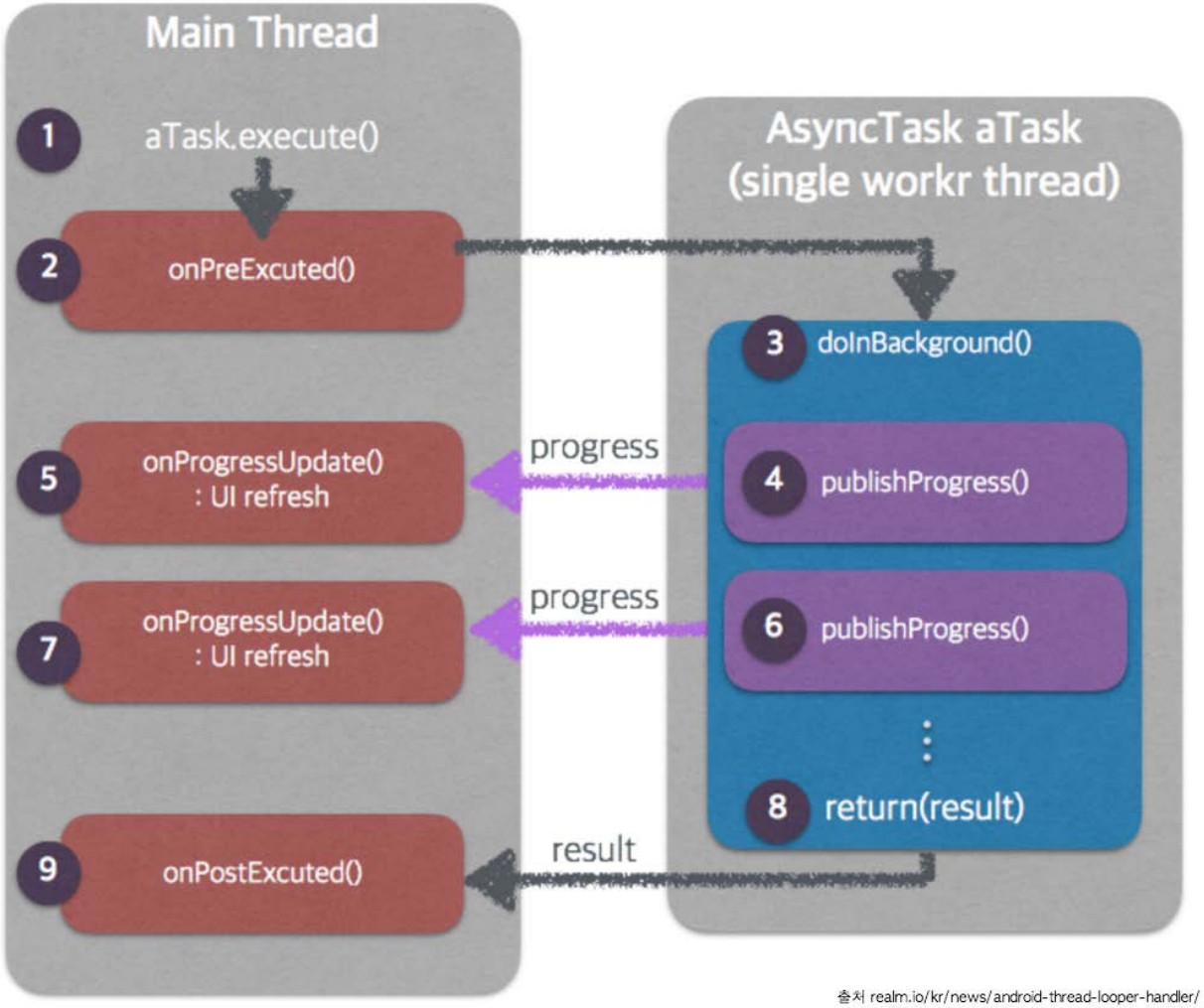
```
new Thread(new Runnable() {
    public void run() {
        for (int i = 0; i <= 100; i++) {
            runOnUiThread(new Runnable() {
                public void run() {
                    textView.setText("This is set by runOnUiThread");
                }
            });
        }
    }
}).start();
```

실행 결과


2:05
ProgressBarTest
Progress Bar Test
2:05
ProgressBarTest
Progress Bar Test

AsyncTask 클래스 사용 방법

- AsyncTask 클래스는 1.5 버전부터 추가된 클래스로서 작업 스레드와 관련된 복잡한 부분을 쉽게 처리해주는 클래스



출처 realm.io/kr/news/android-thread-looper-handler/

AsyncTask 클래스 파라메터

- 타입 파라메터

  AsyncTask<Params, Progress, Result>

  - void execute(**Params**... params)
  - **Result** doInBackground(**Params**... params)
  - void publishProgress(**Progress**... values)
  - void onProgressUpdate(**Progress**... values)
  - void onPostExecute(**Result** res)

AsyncTask 예

UI thread

worker thread

UI thread

```
public void onClick(View v) {
    new DownloadImageTask().execute("http://example.com/image.png");
}

private class DownloadImageTask extends AsyncTask<String, Void, Bitmap> {

    protected Bitmap doInBackground(String... urls) {
        return loadImageFromNetwork(urls[0]);
    }


    protected void onPostExecuted(Bitmap result) {
        mImageView.setImageBitmap(result);
    }
}
```

예제

```
class CounterTask extends AsyncTask<Integer, Integer, Integer> {
    protected void onPreExecute() {   }

    // run by the worker thread
    protected Integer doInBackground(Integer... value) {
        mProgressStatus = value[0];
        while (mProgressStatus < 100) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
            }
            mProgressStatus++;
            publishProgress(mProgressStatus);
        }
        return mProgressStatus;
    }

    // run by the main thread
    protected void onProgressUpdate(Integer... value) {
        mProgress.setProgress(value[0]);
    }

    // run by the main thread
    protected void onPostExecute(Integer result) {
        mProgress.setProgress(result);
    }
}
```

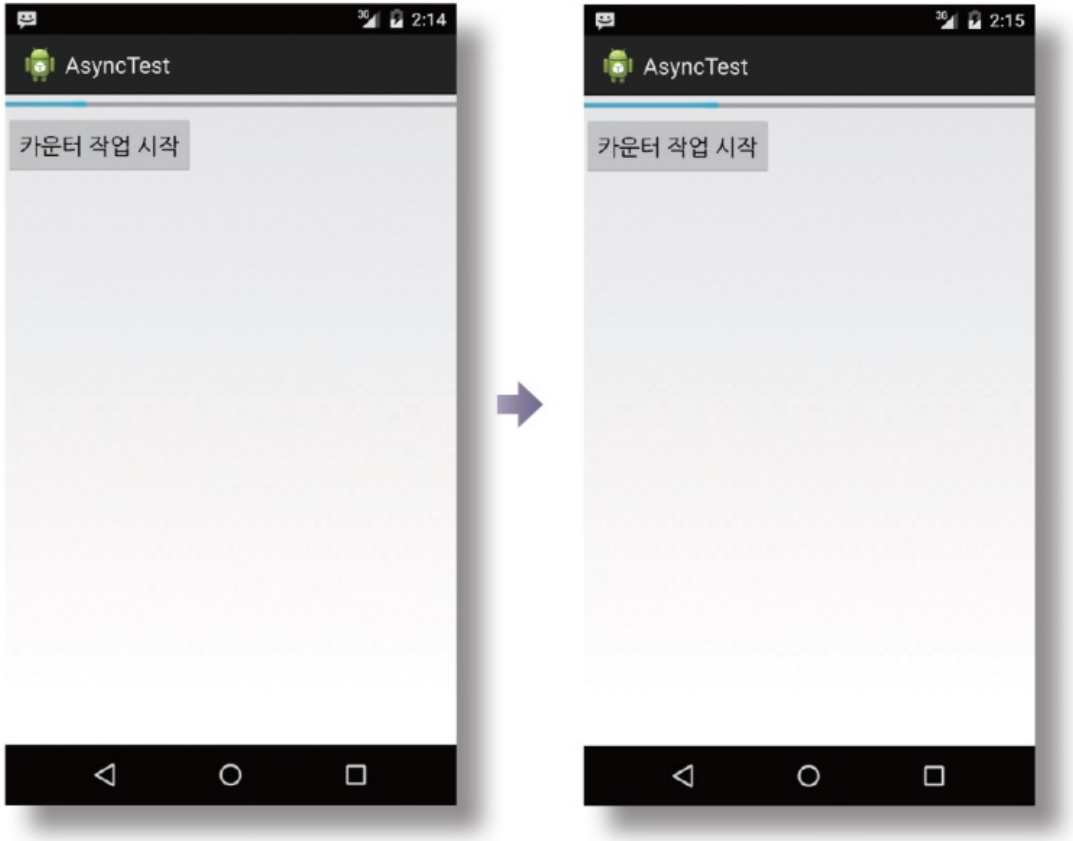
UI thread

worker thread

UI thread

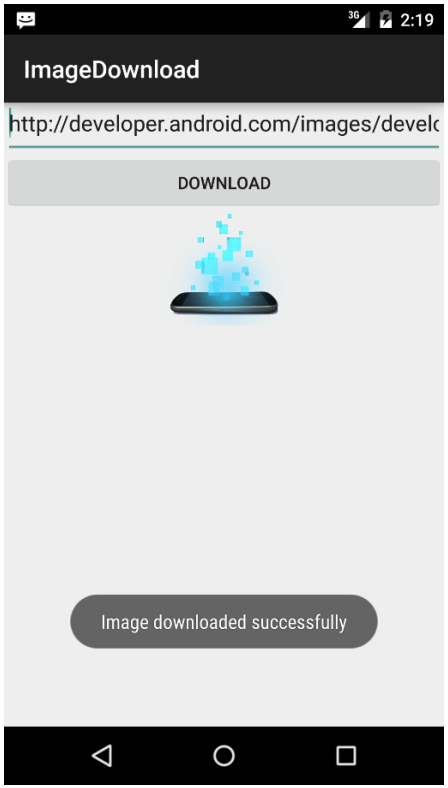
UI thread

실행 결과



예제: 스레드를 이용한 이미지 다운로드

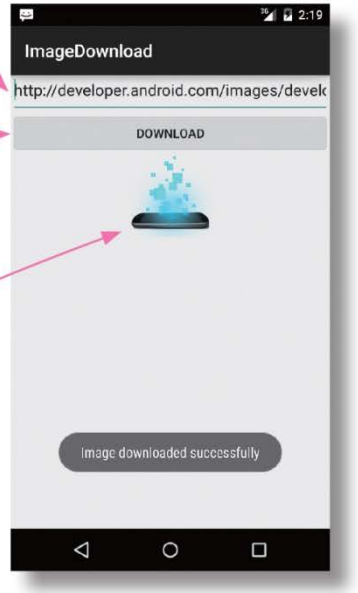
- 스레드를 이용하여서 URL로부터 이미지를 다운로드하고 이것을 이미지뷰에 표시



사용자 인터페이스 작성

main.xml

```
<RelativeLayout 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" >

    <EditText
        android:id="@+id/et_url"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:text="http://developer.android.com/images/develop/app_components.png"
        android:inputType="textUri"
        tools:context=".MainActivity" />

    <Button
        android:id="@+id/btn_download"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/et_url"
        android:text="download"
        tools:context=".MainActivity" />

    <ImageView
        android:id="@+id/iv_image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/btn_download"
        android:layout_centerHorizontal="true"
        android:contentDescription="description"
        tools:context=".MainActivity" />

</RelativeLayout>
```



코드 작성

ImageDownloadActivity.java

```
package kr.co.company.imagedownload;
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.

public class ImageDownloadActivity extends ActionBarActivity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btnDownload = (Button) findViewById(R.id.btn_download);

        btnDownload.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
```

버튼이 클릭되면 다운로드 스레드를 생성하고 실행한다.

```
        EditText etUrl = (EditText) findViewById(R.id.et_url);
        DownloadTask downloadTask = new DownloadTask();
        downloadTask.execute(etUrl.getText().toString());
    }
});
}
```

인터넷 URL 주소를 받아서 연결하고 이미지 파일을 받아서 해독하여 비트맵으로 만든다.

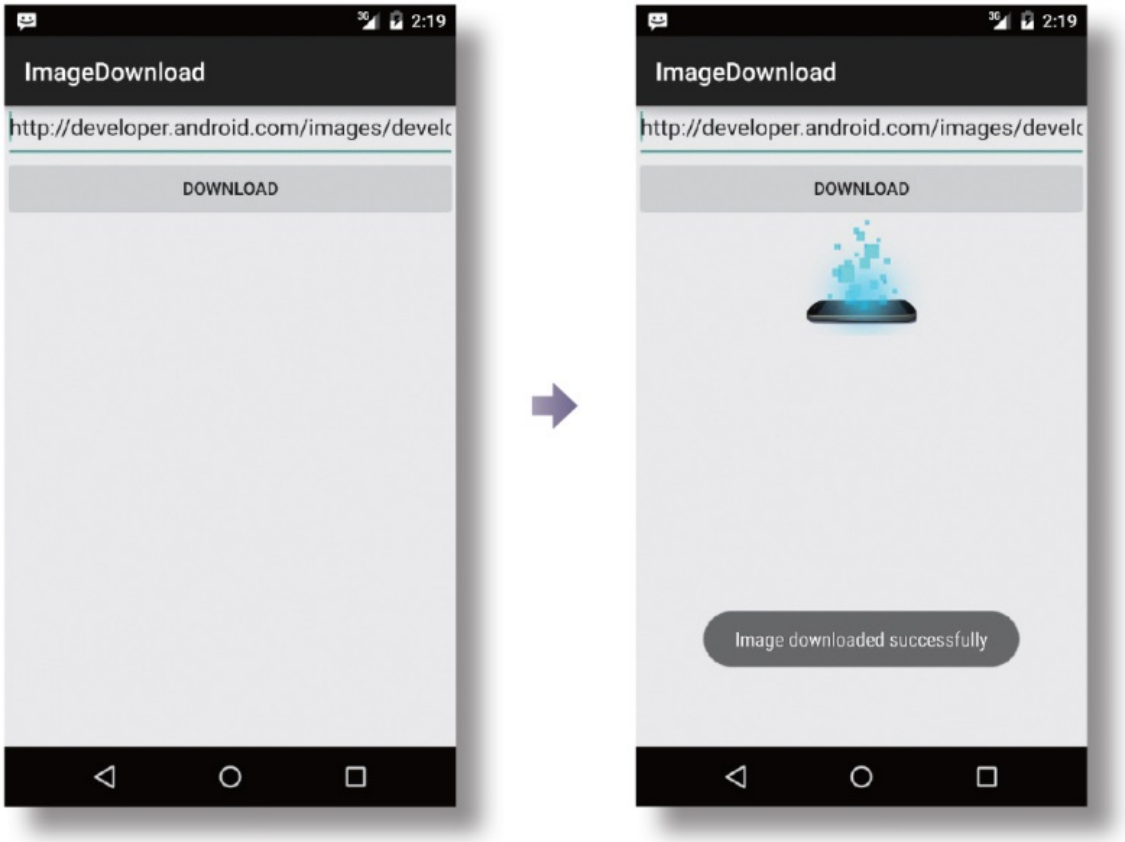
```
private Bitmap downloadUrl(String strUrl) throws IOException {
    Bitmap bitmap = null;
    InputStream iStream = null;
    try {
        URL url = new URL(strUrl);
        HttpURLConnection urlConnection = (HttpURLConnection) url
                .openConnection();
        urlConnection.connect();
        iStream = urlConnection.getInputStream();
        bitmap = BitmapFactory.decodeStream(iStream);
    } catch (Exception e) {
        Log.d("Exception while downloading url", e.toString());
    } finally {
        iStream.close();
    }
    return bitmap;
}
```

AsyncTask 클래스를 이용하여서 다운로드를 진행한다.

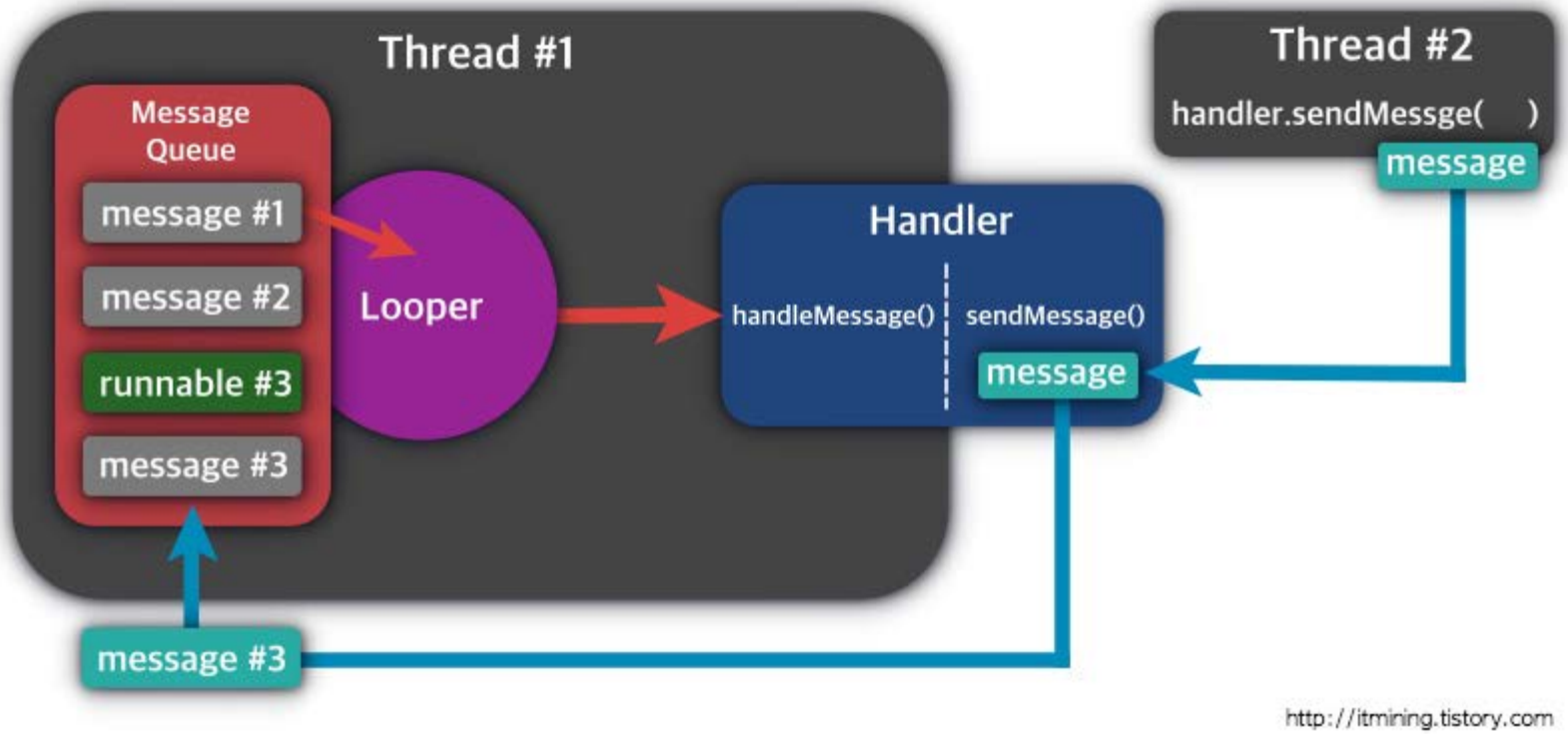
```
private class DownloadTask extends AsyncTask<String, Integer, Bitmap> {
        Bitmap bitmap = null;

        @Override
        protected Bitmap doInBackground(String... url) {
            try {
                bitmap = downloadUrl(url[0]);
            } catch (Exception e) {
                Log.d("Background Task", e.toString());
            }
            return bitmap;
        }

        @Override
        protected void onPostExecute(Bitmap result) {
            ImageView iView = (ImageView) findViewById(R.id.iv_image);
            iView.setImageBitmap(result);
            Toast.makeText(getBaseContext(), "Image downloaded successfully",
                    Toast.LENGTH_SHORT).show();
        }
    }
}
```

스레드로 실행되면 배경 작업은 여기에 기술한다.

실행 결과

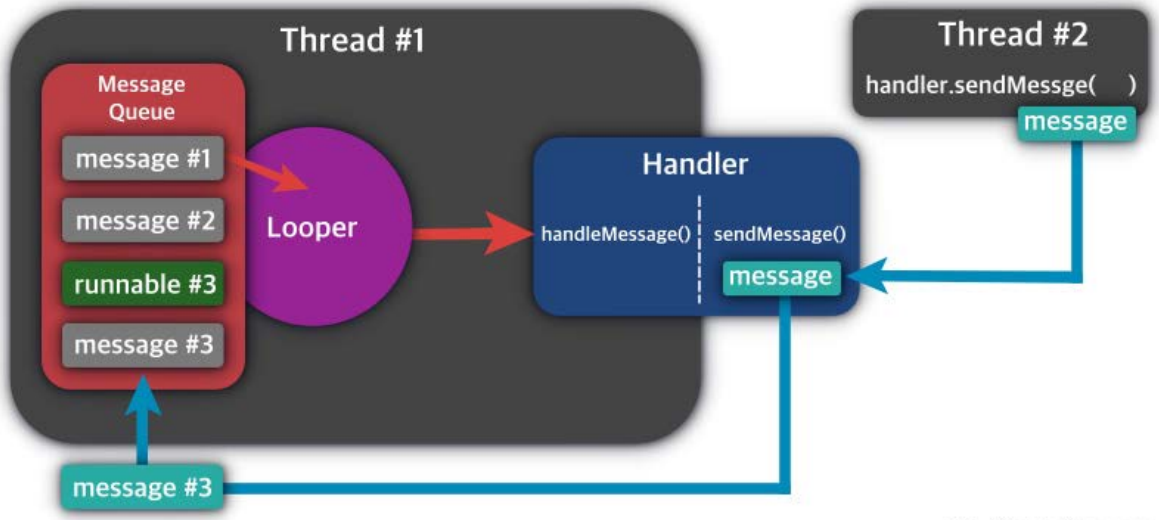

2:19
ImageDownload
http://developer.android.com/images/develc
DOWNLOAD
2:19
ImageDownload
http://developer.android.com/images/develc
DOWNLOAD
Image downloaded successfully

Handler 클래스를 사용하는 방법



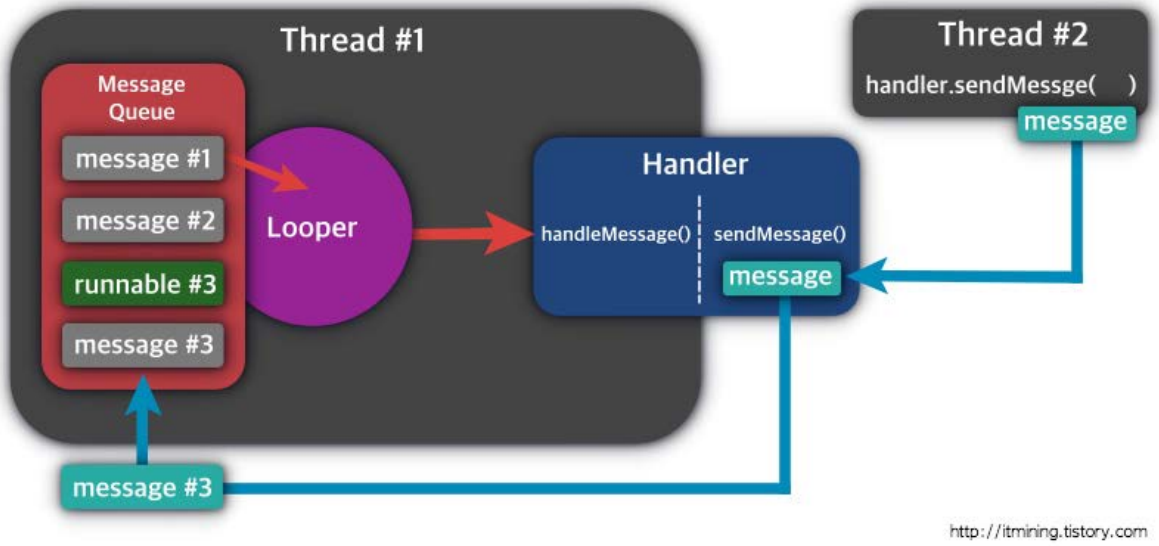
Handler 클래스를 사용하는 방법

```
Thread t = new Thread(new Runnable(){
        public void run() {
                Looper.prepare();
                handler = new Handler();
                Looper.loop();
        } });
t.start();
```

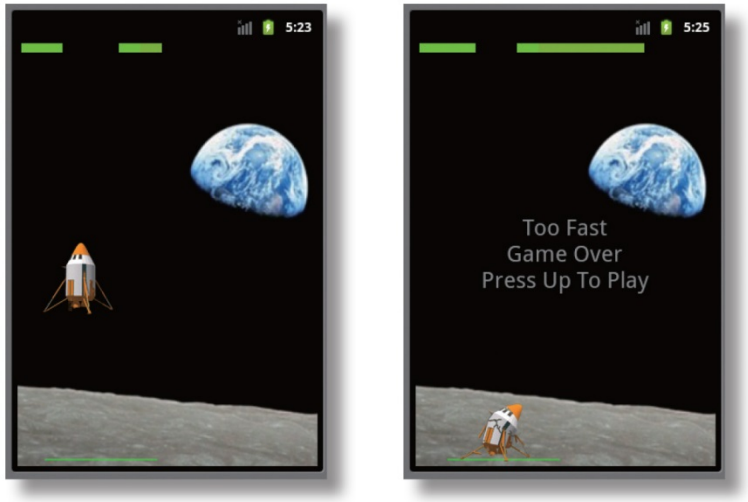


Handler 클래스를 사용하는 방법

```
HandlerThread t = new HandlerThread("My Handler Thread");
t.start();
handler = new Handler(t.getLooper());
```



예제: LunarLander 게임



서피스 뷰

```
class LunarView extends SurfaceView implements SurfaceHolder.Callback {
    public Handler mHandler;
    ...
    public void surfaceCreated(SurfaceHolder holder) {
        // 스레드를 시작한다.
        thread.setRunning(true);
        thread.start();
    }

    public void surfaceDestroyed(SurfaceHolder holder) {
        boolean retry = true;
        thread.setRunning(false);
        while (retry) {
            try {
                thread.join();
                retry = false;
            } catch (InterruptedException e) {
            }
        }
    }
}
```

서피스가 생성되면 스레드를 시작한다.

서피스가 소멸되면 스레드를 중지시킨다.

LunarView.java

```
class LunarView extends SurfaceView implements SurfaceHolder.Callback {
    ...
    class LunarThread extends Thread {
    ...
        @Override
        public void run() {
            while (mRun) {
                Canvas c = null;
                try {
                    c = mSurfaceHolder.lockCanvas(null);
                    synchronized (mSurfaceHolder) {
                        doDraw(c);
                    }
                } finally {
                    if (c != null) {
                        mSurfaceHolder.unlockCanvasAndPost(c);
                    }
                }
            }
        private void doDraw(Canvas canvas) {
            canvas.drawBitmap(mBackgroundImage, 0, 0, null);
            mLanderImage.setBounds(x++, y++, x + 100, y + 100);
            if( x > mCanvasWidth ) x = 0;
            if( y > mCanvasHeight ) y = 0;
            mLanderImage.draw(canvas);
        }
    ...
    }
}
```

스레드로 구현된다.

동기화 부분으로 서피스를 독점하면서 그림을 그린다.

달 착륙선은 무조건 왼쪽 상단에서 오른쪽 하단으로 이동한다.

사용자 인터페이스

lunar_layout.xml


```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.example.android.lunarlander.LunarView
      android:id="@+id/lunar"
      android:layout_width="match_parent"
      android:layout_height="match_parent"/>

</FrameLayout>
```

사용자가 정의한 LunarView를 화면에 표시한다.

실행 결과