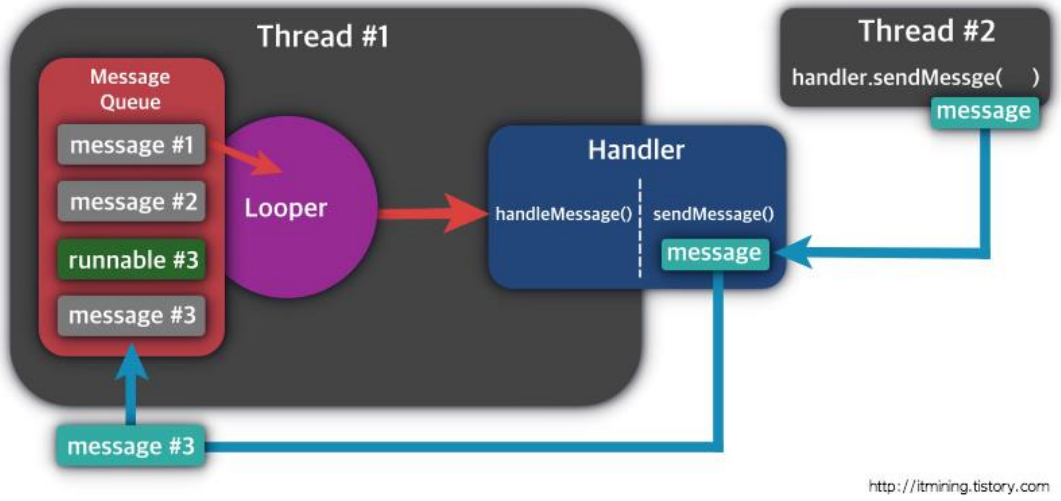
핸들러를 이용한 프로그래스바 구현하기

기초 원리 학습

- 핸들러는 쓰레드와 같이 주기적인 처리를 할 수 있는 기능을 제공
- 쓰레드는 사용할 경우 화면 접근이 어려운 반면 핸들러는 자유롭게 접근 가능
- 핸들러
  - 기본 생성자를 통해 Handler를 생성하면, 생성되는 Handler는 해당 Handler를 호출한 스레드의 MessageQueue와 Looper에 자동 연결
  - 이런 식으로 메인 스레드에서 Handler를 생성하면 해당 Handler는 호출한 스레드의 MessageQueue와 Looper에 자동 연결 되므로 다른 스레드에서 Handler를 통해 메시지를 전달하면 메인 스레드에서 UI 작업을 할 수 있도록 하는 형식



기초 원리 학습

OneKick

- 안드로이드에서는 메인스레드와 서브스레드 간에 Handler를 통해 메시지를 전달하여 메시지 큐에 저장하는 방식의 통신을 사용.
  - 메시지 큐는 FIFO(First In First OUT) 방식이기 때문에 먼저 전달 받은 메시지를 먼저처리

기초 원리 학습

OneKick

- Handler의 메시지 전달 메소드
  - boolean sendEmptyMessage(int what)
    - what이라고 하는 숫자하나 정보로만 전달
  - boolean sendMessage(Message msg)
    - 전달해야할 데이터가 많다면 Message라는 객체를 전달하는 것이 좋음

기초 원리 학습

OneKick

- 핸들러는 일정 지연시간을 줄 수 있는 메소드를 이용하면 자기 자신을 지연시간을 주면서 무한히 호출할 수 있기 때문에 쓰레드 대신 사용할 수도 있음

- 딜레이가 있는 메소드
  - boolean sendEmptyMessageDelayed(int what, long delayMillis)
  - boolean sendMessageDelayed(Message msg, long delayMillis)
  - 밀리세컨드 단위의 지연시간을 인자로 전달하면 그 시간이 지난 이후에 메시지가 전달

기초 원리 학습

OneKick

- 무한히 호출되는 핸들러를 어느순간 멈추고 싶다면 특정 메시지를 삭제 함

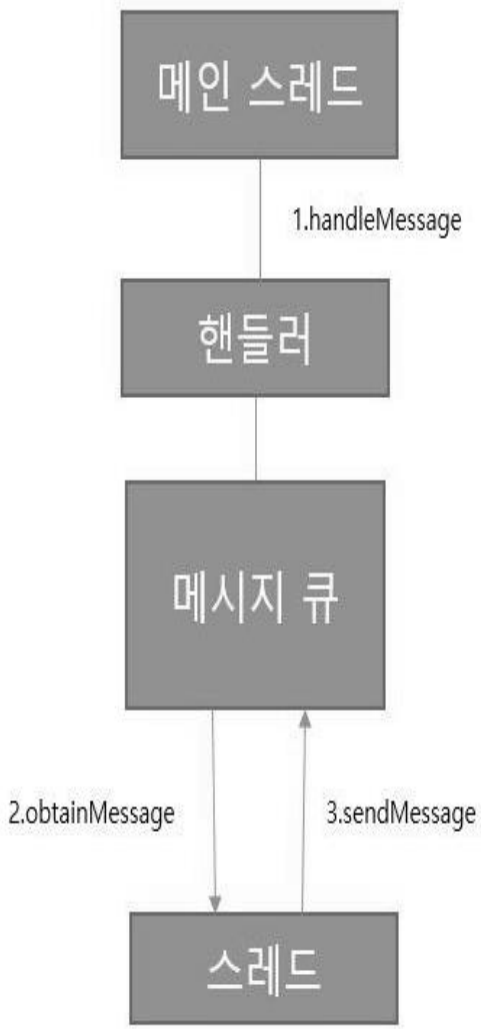
- 핸들러에 있는 특정 메시지를 삭제하는 메소드
  - void removeMessages(int what)
    - 메소드에서 what으로 지정한 메시지를 모두 삭제
  - void removeMessages(int what, Object object)
    - object를 인자로 받고 있는데 특정 객체로 부터 수신한 메시지만 삭제

  - removeMessages 함수는 처리되지 않은 모든 메시지를 제거한다.

기초 원리 학습

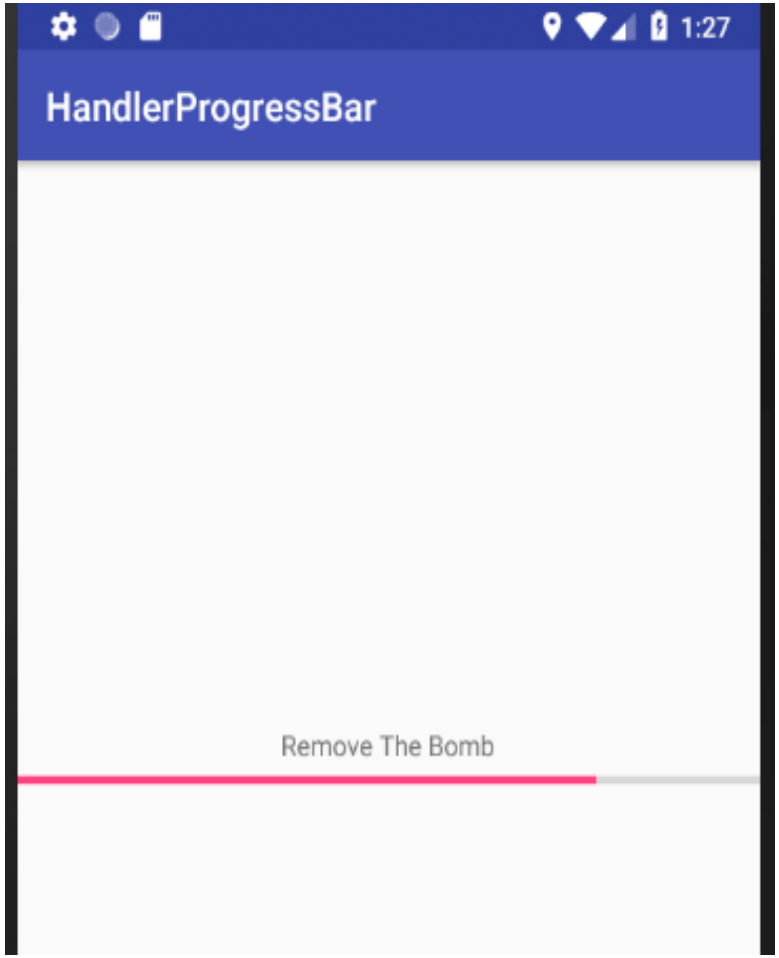
OneKick

- 핸들러를 사용하는 과정
  - 스레드가 UI를 제어하기 위해서는 핸들러에게 정보를 줘야 함.
  - 스레드가 먼저 핸들러 정보를 얻어야하는데 그것이 바로 2.obtainMessage함수를 호출하면 Message객체를 얻음
  - Message객체를 얻었으니 여기에 데이터를 넣고 3.sendMessage함수를 사용하여 메세지 큐에 데이터를 보냄
  - 메시지 큐에 들어간 데이터 순서대로 1.handleMessage 함수를 통하여 데이터를 제어 함.



핸들러 실행 구조

- 100초 동안 1씩 프로그래스 감소 시키다가 멈추기



기초 실습

OneKick

- 핸들러를 이용하여 시간이 감소되는 프로그래스바를 작성

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="안드로이드 APK 파일의 스키마 경로는 생략합니다."
    android:layout_width="match_parent" // 부모를 안벗어나게 최대화
    android:layout_height="match_parent" // 부모를 안벗어나게 최대화
    android:orientation="vertical"
    android:gravity="center"
    >
    <TextView
        android:layout_width="wrap_content" // 컨텐츠에 맞게 최소화
        android:layout_height="wrap_content" // 컨텐츠에 맞게 최소화
        android:text="Remove The Bomb"
        />
    <ProgressBar
        android:id="@+id/progressBar"
        style="@style/Widget.AppCompat.ProgressBar.Horizontal"
        android:layout_width="match_parent" // 부모를 안벗어나게 최대화
        android:layout_height="wrap_content" // 컨텐츠에 맞게 최소화
        android:max="100"
        android:progress="100" />
</LinearLayout>
```

응용 실습

OneKick

- 핸들러를 이용해서 시간이 점차 줄어드는 프로그래스 기능을 구현

```
package com.iot.handlerprogressbar;

import android.os.Message;
import android.support.v7.app.AppCompatActivity; // 액티비티 부모 클래스
import android.os.Bundle; // 액티비티 생성 번들
import android.widget.ProgressBar;
import android.os.Handler;

public class MainActivity extends AppCompatActivity { // 메인 화면
    public static final long DELAY_MS = 100;
    public static final int WHAT_PROGRESS = 1;
    private ProgressBar progressBar;
    private Handler handler = new Handler() {
        @Override // 부모 메소드 재정의
        public void handleMessage(Message msg) {
            super.handleMessage(msg);
            if(progressBar.getProgress()>0) {
                progressBar.setProgress(progressBar.getProgress()-1);
            }
            handler.sendEmptyMessageDelayed(WHAT_PROGRESS, DELAY_MS);
        }
    };
```

응용 실습

OneKick

```
    @Override // 부모 메소드 재정의
    protected void onCreate(Bundle savedInstanceState) { // 화면생성 이벤트
        super.onCreate(savedInstanceState); // 부모 생성자 호출
        setContentView(R.layout.activity_main); // 메인 화면 표시
        progressBar = (ProgressBar)findViewById(R.id.progressBar);
        handler.sendEmptyMessageDelayed(WHAT_PROGRESS, DELAY_MS);
    }
}
```

심화 학습

OneKick

- 만약 프로그래스가 0에 다다르면 폭발화면을 표시하도록 수정해 보자.
  - 그리고 폭발 전에 화면을 터치하면 프로그래스가 멈추도록 구현해 보도록 한다.

```
private ProgressBar progressBar;

private Handler handler = new Handler() {

        @Override // 부모 메소드 재정의
        public void handleMessage(Message msg) {
                super.handleMessage(msg);

                if(progressBar.getProgress()>0) {
                        progressBar.setProgress(progressBar.getProgress()-1);
                }
                handler.sendEmptyMessageDelayed(WHAT_PROGRESS, DELAY_MS);
        }
};
```