

8. 실습 보고서
- Stop Watch -

보 고 자 성 명	20181151 김 지 원
보 고 자 학 과	산 업 경 영 공 학 과
담 당 교 수 성 명	정 진 만 교 수 님
과 목 명	모 바 일 프 로 그 래밍 01분 반
제 출 날 짜	2020.05.09

- 목차 -

1. 서론	3p
1) 개요	
2) 요청사항 분석	
3) 추가 사항	
4) 초기 디자인	
2. 본론	
1) activity_main.xml	4p
2) res > raw > snd.mp3	4p
3) MainActivity.java	
① 소스 코드 원본	5p
② 소스 코드 분석	7p
4) 결과 화면	11p
3. 결론	
1) 나만의 스톱워치 앱 장점	13p
2) 나만의 스톱워치 앱 단점	13p
3) 배운 점	14p

1. 서론

1) 개요

스톱워치는 대부분의 모바일 기기에 기본적으로 포함하고 있는 어플 중 하나이다. 그 외에 필요한 기능을 추가하여 나만의 스톱워치 앱을 제작하여 사용할 수 있다. 스톱워치를 실행시키면 start 한 후부터 증가한 시간이 실시간으로 수정되어 출력된다. 실행되는 중간에 LAP 버튼을 통해 당시의 시간을 기록하는 기능과 원하는 만큼의 시간이 경과된 후 알람이 울리는 기능을 추가한 스톱워치 앱을 제작한다.

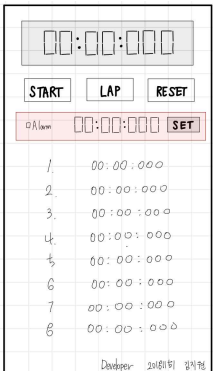
2) 요청사항 분석

- ① 분, 초, 밀리초 단위까지 표시한다.
- ② 스톱워치를 시작하는 버튼과 중간에 경과된 시간을 기록하는 버튼, 기록과 경과 시간을 모두 초기화하는 버튼을 포함한다.
- ③ 스크롤 기능을 추가하여 기록을 개수의 제한 없이 지속적으로 추가할 수 있다.

3) 추가 사항

원하는 만큼의 시간이 경과되면 알람이 울리는 벨 소리가 울리는 기능을 추가한다. checkbox를 이용하여 alarm 기능의 사용 여부를 확인하고 사용할 경우, 초기 비활성화 상태인 시간을 설정하는 editText와 set Button을 활성화시켜 시간을 지정한다.

4) 초기 디자인

	<ul style="list-style-type: none">- 상단에 스톱워치의 경과된 시간을 나타내는 Text를 출력한다.- 스톱워치를 제어할 수 있는 “START”, “LAP”, “RESET” Button을 배치한다.- 알람 사용 여부를 판단하는 checkBox와 알람 세부사항을 설정하는 editText 와 Button을 추가한다.- 하단에 스톱워치의 기록을 추가할 수 있는 TextView를 배치한다.- 가장 하단에 개발자의 정보를 기입한다.
---	--

2. 본문

1) layout > activity_main.xml

- 모든 항목을 Scroll View에 포함시켜 스크롤을 사용한다. (마우스 휠 지원)
- vertical Linear Layout에 Linear Layout 3개와 textView 4개를 삽입한다.
- 첫 번째 Linear Layout에 어플의 제목과 개발자 정보를 입력한다.
- 첫 번째 Text View에는 스톱워치의 화면을 출력한다.
- 두 번째 Linear Layout에는 스톱워치를 제어하는 버튼 3개를 삽입한다.
- 세 번째 Linear Layout에 알람 사용 여부를 확인하는 CheckBox와 알람 시간을 설정하는 EditText, 시간을 저장하는 Button을 추가한다.
- 스톱워치의 시간을 기록하는 TextView와 음원의 저작권을 게시하는 TextView를 삽입한다.

2) res > raw > snd1.mp3

- resource에 "raw" directory를 새로 생성하여 음원을 복사하여 삽입한다.

3) MainActivity.java

① 소스 코드 원본

```
package com.example.week8_stopwatch_assign;

import androidx.appcompat.app.AppCompatActivity;

import android.media.AudioManager;
import android.media.SoundPool;
import android.os.Bundle;
import android.os.Handler;
import android.os.SystemClock;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    TextView tv_DP;
    TextView tv_TM;
    Button btn_ST;
    Button btn_LP;
    Button btn_RS;
    CheckBox cb_AR;
    EditText et_AR_min;
    EditText et_AR_sec;
    EditText et_AR_ms;
    Button btn_AR;
    SoundPool sp_AR;
    TextView tv_LP;

    boolean tv_DP_state = false;
    long mstartTime;
    Handler mHandler = new Handler();
    int record = 0;
    boolean SW_state = false;
    String alarm="0000000";
    String alarm_check="0000000";
    int snd_id;
    boolean alarm set = false;

    class MyRunnable implements Runnable{
        @Override
        public void run() {
            long t = SystemClock.elapsedRealtime();
            long lab = t - mstartTime;

            long ms, sec, min;
            String ms_s, sec_s, min_s;

            ms = lab % 1000;
            ms_s = "" + ms;
            if(ms<10)
                ms_s = "00" + ms;
            else if(ms<100)
                ms_s = "0" + ms;

            sec = lab / 1000 % 60;
            sec_s = "" + sec;
            if(sec<10)
                sec_s = "0" + sec;

            min = lab / 60000;
            min_s = "" + min;
            if(min<10)
                min_s = "0" + min;

            tv_TM.setText(min_s + " : " + sec_s+" : "+ms_s);
            alarm_check = min_s+sec_s+ms_s;

            if(SW_state)
            {
                if((Integer.parseInt(alarm) < Integer.parseInt(alarm_check)) && (alarm_set == true))
                {
```

```

sp_AR.play(snd_id,1,1,0,0,1);
SW_state =false;
alarm_set = false;
}
mhandler.post(this);
}
else
{
tv_TM.setText("00 : 00 : 000");
}
}
}

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

tv_DP = findViewById(R.id.textView_DP);
tv_DP.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
tv_DP_state = !tv_DP_state;
if(tv_DP_state)
tv_DP.setText("20181151 김지원");
else
tv_DP.setText("Developer");
}
});

tv_TM = findViewById(R.id.textView_TM);
tv_TM.setText("00 : 00 : 000");

btn_ST = findViewById(R.id.button_ST);
btn_ST.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
SW_state = true;
mstartTime = SystemClock.elapsedRealtime();
MyRunnable runnable = new MyRunnable();
mhandler.post(runnable);
}
});

btn_LP = findViewById(R.id.button_LP);
btn_LP.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
record = record + 1;
String currentTime = tv_TM.getText().toString();
tv_LP.append(record + ". \t" + currentTime + "\n");
}
});

btn_RS = findViewById(R.id.button_RS);
btn_RS.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
SW_state = false;
tv_LP.setText("");
record = 0;
alarm_set = false;
}
});

cb_AR = findViewById(R.id.checkBox_AR);
cb_AR.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
@Override
public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
if(cb_AR.isChecked()==false)
{
et_AR_min.setEnabled(false);
et_AR_sec.setEnabled(false);
et_AR_ms.setEnabled(false);
btn_AR.setEnabled(false);
}
else
{

```

```

et_AR_min.setEnabled(true);
et_AR_sec.setEnabled(true);
et_AR_ms.setEnabled(true);
btn_AR.setEnabled(true);
}
}
});

et_AR_min = findViewById(R.id.editText_AR_min);
et_AR_min.setEnabled(false);

et_AR_sec = findViewById(R.id.editText_AR_sec);
et_AR_sec.setEnabled(false);

et_AR_ms = findViewById(R.id.editText_AR_ms);
et_AR_ms.setEnabled(false);

btn_AR = findViewById(R.id.button_AR);
btn_AR.setEnabled(false);
btn_AR.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String AR_min = et_AR_min.getText().toString();
        String AR_sec = et_AR_sec.getText().toString();
        String AR_ms = et_AR_ms.getText().toString();

        if(AR_min.matches("") || AR_sec.matches("") || AR_ms.matches(""))
            Toast.makeText(getApplicationContext(), "시간을 입력하세요.", Toast.LENGTH_SHORT).show();
        else
        {
            if(Integer.parseInt(AR_sec) >= 60)
            {
                Toast.makeText(getApplicationContext(), "초는 59초까지만 입력 가능합니다.", Toast.LENGTH_SHORT).show();
            }
            else
            {
                alarm = AR_min + AR_sec + AR_ms;
                Toast.makeText(getApplicationContext(), "알람이 설정되었습니다.", Toast.LENGTH_SHORT).show();
                alarm_set = true;
            }
        }
    }
});

sp_AR = new SoundPool(1, AudioManager.STREAM_MUSIC, 0);
snd_id = sp_AR.load(this, R.raw.snd, 1);

tv_LP = findViewById(R.id.textView_LP);
tv_LP.setText("");

}
}

```

② 소스 코드 분석

```

package com.example.week8_stopwatch_assign;

import androidx.appcompat.app.AppCompatActivity;

import android.media.AudioManager;
import android.media.SoundPool;
import android.os.Bundle;
import android.os.Handler;
import android.os.SystemClock;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

```

1. Import

```

public class MainActivity extends AppCompatActivity {

    TextView tv_DP;
    TextView tv_TM;
    Button btn_ST;
    Button btn_LP;
    Button btn_RS;
    CheckBox cb_AR;
    EditText et_AR_min;
    EditText et_AR_sec;
    EditText et_AR_ms;
    Button btn_AR;
    SoundPool sp_AR;
    TextView tv_LP;

    boolean tv_DP_state = false;
    long mstartTime;
    Handler mHandler = new Handler();
    int record = 0;
    boolean SW_state = false;
    String alarm="000000";
    String alarm_check="000000";
    int snd_id;
    boolean alarm_set = false;
}

```

2. Main Activity class

- 각 위젯을 선언한다.
- 아래의 코드에서 필요한 매개변수들을 선언해준다.

```

class MyRunnable implements Runnable{
    @Override
    public void run() {
        long t = SystemClock.elapsedRealtime();
        long lab = t - mstartTime;

        long ms, sec, min;
        String ms_s, sec_s, min_s;

        ms = lab % 1000;
        ms_s = "" + ms;
        if(ms<10)
            ms_s = "00" + ms;
        else if(ms<100)
            ms_s = "0" + ms;

        sec = lab / 1000 % 60;
        sec_s = "" + sec;
        if(sec<10)
            sec_s = "0" + sec;

        min = lab / 60000;
        min_s = "" + min;
        if(min<10)
            min_s = "0" + min;

        tv_TM.setText(min_s + " : " + sec_s + " : " + ms_s);
        alarm_check = min_s+sec_s+ms_s;

        if(SW_state)
        {
            if((Integer.parseInt(alarm) < Integer.parseInt(alarm_check)) && (alarm_set == true))
            {
                sp_AR.play(snd_id, leftVolume: 1, rightVolume: 1, priority: 0, loop: 0, rate: 1);
                SW_state =false;
                alarm_set = false;
            }
            mHandler.post( this);
        }
        else
        {
            tv_TM.setText("00 : 00 : 000");
        }
    }
}

```

3. Runnable class

- 반응성 향상과 Lengthy operation 처리를 같이 해결하기 위해 Runnable을 사용하였다.
- Start Button을 클릭하면 호출이 되는 다음 클래스는 클릭 당시의 시간과 현재의 시간의 차이를 이용하여 스톱 위치의 시간을 출력한다. 시간은 SystemClock에서 받아온다.
- 수식을 이용하여 경과한 시간을 밀리초, 초, 분으로 나누어 출력한다.

- 스톱워치가 작동하는 상태인 SW_state가 true일 때 알람의 기능을 사용하고, 알람 설정을 해놓은 시간 보다 스톱워치의 시간이 더 커지는 경우에 음원을 출력한다.
- mHandler.post를 이용하여 runnable을 반복적으로 수행한다.
- 만약 Reset을 눌러 SW_state가 false가 되는 상황에서는 스톱워치의 시간이 “00 : 00 : 000”으로 변경된다.

```

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

    tv_DP = findViewById(R.id.textView_DP);
    tv_DP.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            tv_DP_state = !tv_DP_state;
            if(tv_DP_state)
                tv_DP.setText("20181151 김지원");
            else
                tv_DP.setText("Developer");
        }
    });
}

```

4. onCreate() method & DP (개발자 정보)

- 개발자 정보를 출력하는 text View에 “Developer”를 출력하고, 해당 text View를 클릭하면 학번과 이름을 출력하게 구현하였다.
- tv_DP_state를 true와 false를 번갈아 설정되게 하여 text View에는 “Developer”와 “20181151”이 반복해서 변경되며 출력된다.

```

tv_TM = findViewById(R.id.textView_TM);
tv_TM.setText("00 : 00 : 000");

```

5. TM (스톱워치)

- 어플이 실행되면 스톱워치 진행 화면에 “00 : 00 : 000”이 출력되게 설정한다.

```

btn_ST = findViewById(R.id.button_ST);
btn_ST.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        SW_state = true;
        mstartTime = SystemClock.elapsedRealtime();
        MyRunnable runnable = new MyRunnable();
        mHandler.post(runnable);
    }
});

```

6. ST (스톱워치 시작)

- OnClickListener를 이용하여 start Button이 클릭되었을 때 실행되는 코드를 정의한다.
- 스톱워치의 상태를 설정하는 SW_state를 true로 전환하여 스톱워치가 사용되고 있음 표현한다.
- SystemClock을 이용하여 mstartTime에 시간 정보를 저장한다.
- Runnable의 객체를 생성하고 시작한다.

```

btn_LP = findViewById(R.id.button_LP);
btn_LP.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        record = record + 1;
        String currentTime = tv_TM.getText().toString();
        tv_LP.append(record + ". \t" + currentTime + "\n");
    }
});

```

7. LP (스톱워치 시간 기록)

- OnClickListener를 이용하여 Lap Button이 클릭되었을 때 실행되는 코드를 정의한다.
- 기록되는 시간 정보에 순번을 추가하기 위해 Button이 클릭될 때마다 record를 1 증가시킨다.
- tv_TM에 출력되는 시간을 순번과 함께 tv_LP에 추가되는 방식으로 저장한다.

```

    btn_RS = findViewById(R.id.button_RS);
    btn_RS.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            SW_state = false;
            tv_LP.setText("");
            record = 0;
            alarm_set = false;
        }
    });
}
}

```

8. RS (스톱워치 시간 초기화)

- OnClickListener를 이용하여 Reset Button이 클릭되었을 때 실행되는 코드를 정의한다.
- 스톱워치가 더 이상 실행되지 않는다는 의미로 SW_state를 false로 전환한다.
- Lap을 통해 기록되었던 정보들과 순번을 정하는 record도 초기화한다.
- 알람 기능의 사용 여부도 false로 전환한다.

```

    cb_AR = findViewById(R.id.checkBox_AR);
    cb_AR.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            if(cb_AR.isChecked()==false)
            {
                et_AR_min.setEnabled(false);
                et_AR_sec.setEnabled(false);
                et_AR_ms.setEnabled(false);
                btn_AR.setEnabled(false);
            }
            else
            {
                et_AR_min.setEnabled(true);
                et_AR_sec.setEnabled(true);
                et_AR_ms.setEnabled(true);
                btn_AR.setEnabled(true);
            }
        }
    });
}
}

```

9. cb_AR (알람기능 checkBox)

- OnCheckedChangeListener를 이용하여 checkBox의 선택 여부가 바뀌면 실행되는 코드를 정의한다.
- checkBox가 선택되어있지 않다면 알람 시간을 설정하는 edit Text와 Button가 비활성화된다.
- checkBox가 선택되어 있다면 알람 시간을 설정하는 edit Text와 Button가 활성화된다.

```

    et_AR_min = findViewById(R.id.editText_AR_min);
    et_AR_min.setEnabled(false);

    et_AR_sec = findViewById(R.id.editText_AR_sec);
    et_AR_sec.setEnabled(false);

    et_AR_ms = findViewById(R.id.editText_AR_ms);
    et_AR_ms.setEnabled(false);
}
}

```

10. et_AR (알람기능 edit Text)

- 알람 시간을 설정하는 edit Text가 초기 화면에서는 비활성화된 상태로 출력되게 한다.

```

    btn_AR = findViewById(R.id.button_AR);
    btn_AR.setEnabled(false);
    btn_AR.setOnClickListener((v) -> {
        String AR_min = et_AR_min.getText().toString();
        String AR_sec = et_AR_sec.getText().toString();
        String AR_ms = et_AR_ms.getText().toString();

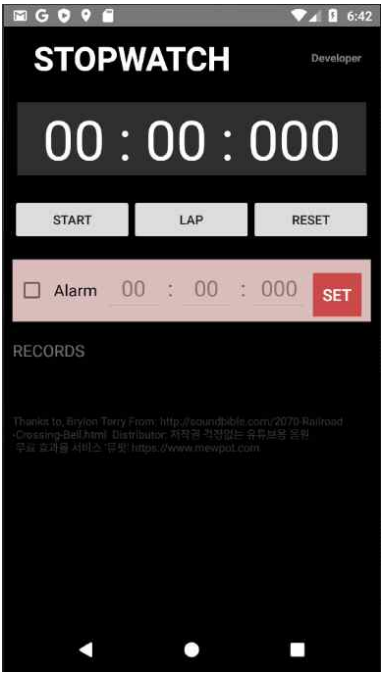
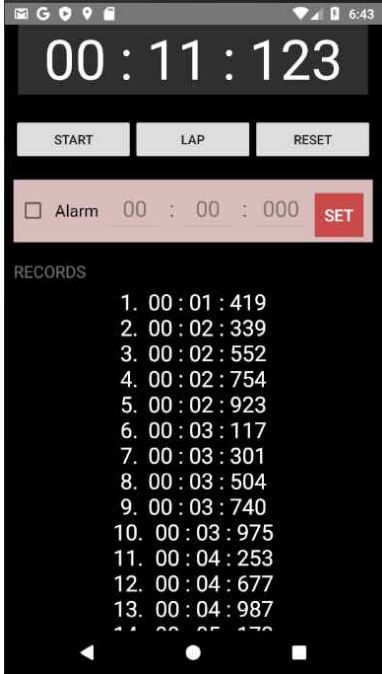
        if(AR_min.matches( regex: "" ) || AR_sec.matches( regex: "" ) || AR_ms.matches( regex: "" ))
            Toast.makeText(getApplicationContext(), text: "시간을 입력하세요.", Toast.LENGTH_SHORT).show();
        else
        {
            if(Integer.parseInt(AR_sec)>= 60)
            {
                Toast.makeText(getApplicationContext(), text: "초는 59초까지만 입력 가능합니다.", Toast.LENGTH_SHORT).show();
            }
            else
            {
                alarm = AR_min + AR_sec + AR_ms;
                Toast.makeText(getApplicationContext(), text: "알람이 설정되었습니다.", Toast.LENGTH_SHORT).show();
                alarm_set = true;
            }
        }
    });
}
}

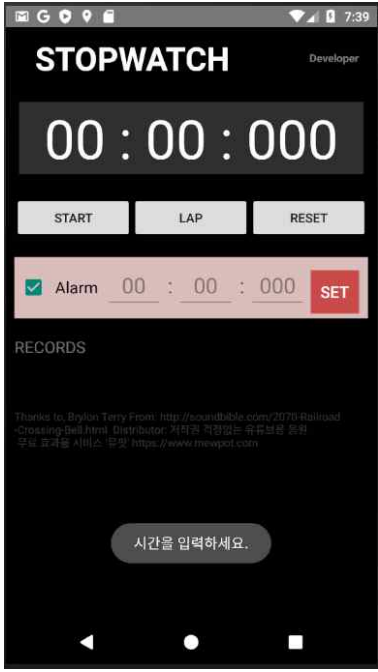
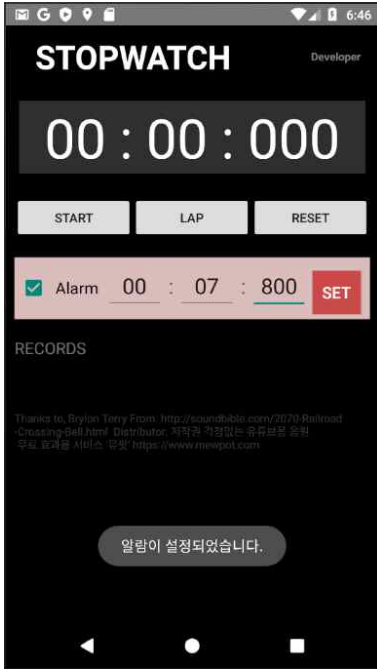
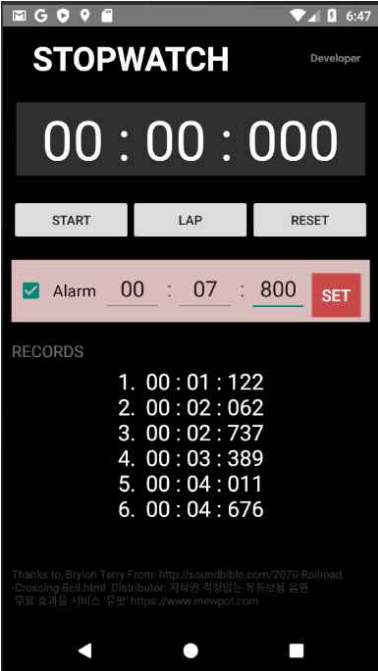
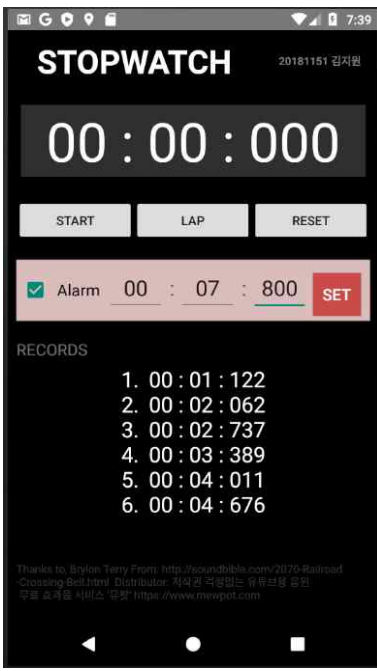
```

11. btn_AR (알람기능 edit Text)

<ul style="list-style-type: none"> - 알람 설정 Button도 초기 화면에서는 비활성화된 상태로 출력된다. - 알람 기능을 사용하지만 시간 설정 없이 set Button을 클릭하면 “시간을 입력하세요”라는 Toast가 출력된다. - 초 단위가 60초 이상이면 분 단위로 넘어가기 때문에 최대로 나올 수 있는 초는 59초이다. 따라서 60초 이상의 숫자가 나오면 “초는 59초까지만 입력 가능합니다”라는 Toast가 출력된다. - 앞에 나온 조건을 모두 통과한다면 알람이 설정된다. 	
<pre>sp_AR = new SoundPool(maxStreams: 1, AudioManager.STREAM_MUSIC, srcQuality: 0); snd_id = sp_AR.load(context: this, R.raw.snd, priority: 1);</pre>	
12. SoundPool (음원 resource 사용) <ul style="list-style-type: none"> - SoundPool을 사용하여 raw directory에 추가해준 음원을 load한다. - Runnable class 안에서 제약 조건에 부합하면 음원이 출력된다. 	
<pre>tv_LP = findViewById(R.id.textView_LP); tv_LP.setText(""); } }</pre>	
13. LP (중간 시간 기록) <ul style="list-style-type: none"> - 초기 화면에서 기록이 비어있는 상태로 출력된다. 	

4) 결과 화면

	
<p style="text-align: center;">앱 초기 화면</p>	
	
<p style="text-align: center;">스톱워치와 시간 기록 화면 (+ 스크롤)</p>	

	
<p>시간 미설정 시 화면</p>	<p>알람 설정 화면</p>
	
<p>지정 시간 이후 화면</p>	<p>학번, 이름 출력 (Easter Egg)</p>

3. 결론

1) 나만의 스톱워치 앱 장점

① 알람 기능을 추가하였다.

: 어플이 아닌 일반 스톱워치들도 보통 여러 가지 기능이 융합되어 있다. 그중 카운트다운의 기능을 구현하고 싶었으나 어플의 주제가 스톱워치이기 때문에 카운트다운과 유사한 다른 부가 옵션으로 무엇을 넣을지 고민했다. 결과적으로, 알람 설정 기능을 고안하였다. 우선 알람 설정과 관련된 세부사항 위젯들은 초기 화면에서 비활성화시킨다. 알람 기능을 사용하기 위해 CheckBox를 선택해 주면 원하는 시간을 유저가 직접 설정할 수 있다. 설정한 기간 이후에 알람이 울린다.

② 유저의 입장을 고려해 세부사항을 설정하였다.

: 직접 어플을 사용한다 생각하고 세부사항 설정에 에러를 줄였다. EditText를 사용하면 초기 화면에서 해당 EditText에 Focus가 맞춰지게 설정되어 있어서 자판이 함께 출력되게 된다. 하지만 자판이 필요한 부분은 필수적인 설정 항목이 아니기 때문에 이러한 자판의 출력이 유저의 입장에서 불편할 수 있다. 이를 해결하기 위해 초기 화면에서 Focus를 Linear Layout에 설정하여 자판이 출력되지 않게 설정하였다. 또한, Easter EGG를 사용하여 학번, 이름을 출력하였다. 어플 사용 시, 너무 많은 공간을 제작자의 정보를 출력하는 것에 사용한다면 화면 활용의 효율성이 떨어질 것이다. 하지만 개발자의 정보는 꼭 들어가야 하기 때문에 easter Egg를 사용하여 학번과 이름이 출력되게 하였다. 어플의 오른쪽 상단에 “Developer”라고 적힌 TextView를 클릭하면 학번과 이름이 출력되게 하였다. 학번과 이름이 적힌 Text를 한 번 더 클릭하면 원래의 상태인 “Developer”가 출력된다. 마지막으로 editText에 숫자를 기입할 때 숫자 외에 다른 문자를 입력하는 상황을 고려하여 0부터 9까지의 숫자만 입력되도록 digit을 설정하였다. editText 설정 시 숫자 자판이 출력되고, “.”이나 “,”을 입력하면 기입이 되지 않는다. edit Text의 설정에서 유저의 숫자 입력의 가이드를 제공하기 위해 어떤 식으로 기입하는지 hint도 나타낸다.

2) 나만의 스톱워치 앱 단점

① 알람에서 벨 소리의 제어 기능이 포함되어 있지 않다.

: 보통 스톱워치에서 지정한 시간이 되면 벨 소리가 울리고 사용자가 멈추면 소리의 출력이 멈추는 것이 대부분이다. 하지만 이 어플은 지정된 시간이 되면 아주 짧은 벨 소리 한 번만 울리게 지정하였다. 소리의 지속시간이 짧기 때문에, 굳이 소리의 출력을 멈추는 기능을 추가하지 않았다. 다음과 같

이 설정한 이유는 적당한 길이의 무료이면서 배포 가능한 음원을 찾지 못하였기 때문이다. 저작권의 문제가 되지 않는 음원이 있다면 음원 선택부터 벨 소리 크기 반복, 속도 등을 사용자가 직접 지정할 수 있게 하는 어플을 제작해보는 것도 흥미로울 것이다.

3) 배운 점

어플을 제작하면서 예상과는 다른 결과가 나왔지만, 그 결과가 어떻게 나온 것인지 이해가 가지 않은 경우가 발생하였다. `snd`가 `runnable`에서 지정한 조건일 경우에 출력되어야 하는데 전혀 상관없는 `Reset Button`을 클릭할 경우에 출력이 되었다. 디버깅을 해보아도 다음과 같은 결과를 이해할 수 없었다. 고민하고, 여러 가지 시도를 해본 결과 `Runnable`이 반복되는 과정을 잘못 이해하고 있었다고 깨달았다. 이러한 오류를 해결하는 것에 장시간이 걸렸지만, `Runnable`의 정확한 작동 방법을 알게 되었다.

지금까지 학습했던 내용 중 이번 스레드에 관한 학습 내용이 가장 어려움이 많았다. 초반에 어플을 설계할 때 구현하고 싶은 기능이 더 많이 있었지만, 구현 능력에 한계가 느껴져 포기하고 적당한 수준의 기능을 추가하여 어플을 제작하였다. 일주일 사이에 이론을 숙지하고 실습을 매끄럽게 진행하는 것은 어렵지만 지금까지 과제를 수행한 것을 보았을 때 앞으로도 계속 과제 수행을 해 나간다면 마지막 프로젝트를 제작할 때는 원하는 기능을 모두 구현할 수 있는 능력을 갖출 수 있을 것이다.