캡스톤 개인보고서

이름 : 차무송

학번 : 32154577

학과 : 소프트웨어학괴

교수님 : 송 인식 교수님

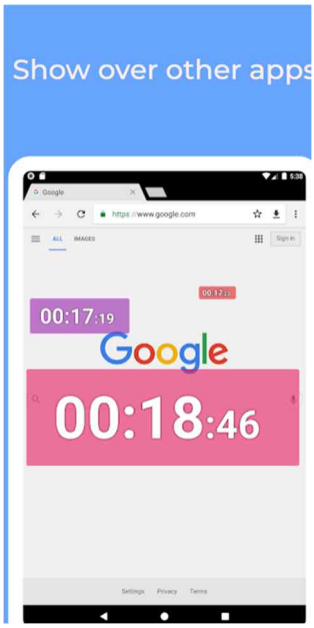
저번 주에 이어서 타이머의 기능을 구현하기로 하였습니다. 타이머를 구현에 앞서 저는 크게 두 가지 기능을 구현할 예정에 있었습니다.

- 타이머 화면에서 작동하는 타이머.
- 백그라운드 즉, 저희 다른 기능 혹은 다른 어플리케이션을 이용하면서 사용할 수 있는 타이머

첫 번째의 타이머 화면에서 작동하는 타이머의 구현을 그리 어렵지 않을 것이라 생각하였습니다. 하지만 두 번째의 백그라운드 타이머 기능을 구현하기에 앞서 어떠한 기술을 사용하여 구현해야 할 지 감이 잡히질 않아 유사한 어플리케이션을 먼저 조사하였습니다.

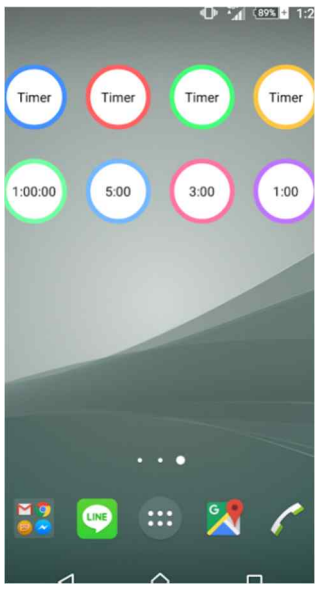
1. Floating Stopwatch

앱을 실행하여 사용자의 설정에 따라 다른 어플리케이션과 동시에 실행이 가능한 스탑워치가 생성됩니다

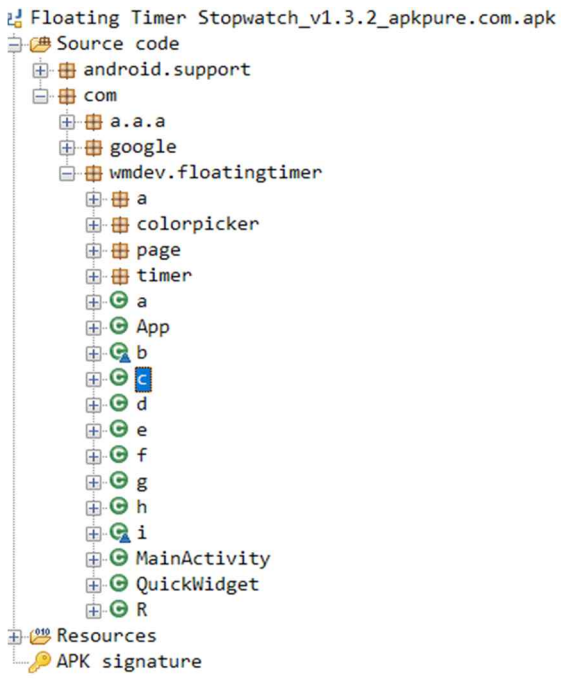


2. Floating Timer

위와 마찬가지로 사용자 설정에 맞게 다른 앱 위에서 실행이 가능한 Timer가 생성됩니다

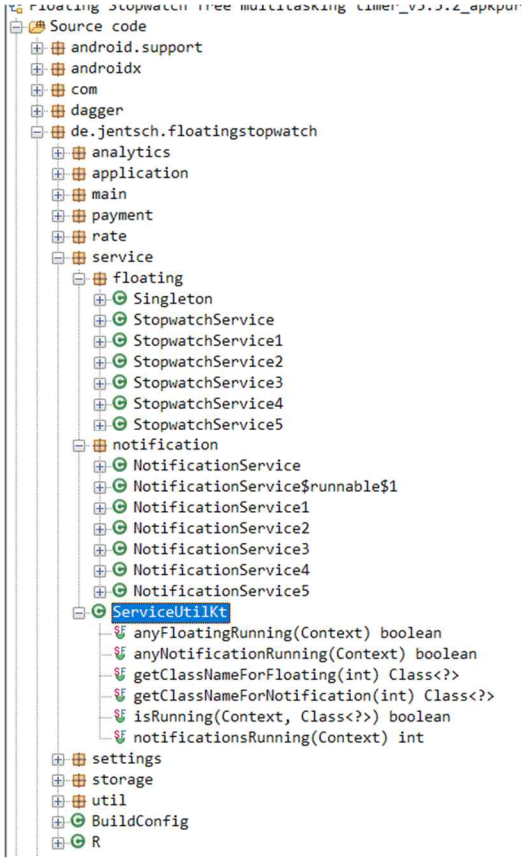


해당 기술에 대한 배경지식이 전혀 없어 Jadx 를 통하여 디컴파일 후 어떠한 기술로 작동하는지 확인해보려고 하였습니다. 먼저 첫번째 Floating stopwatch 에 대하여 디컴파일을 해보았습니다.



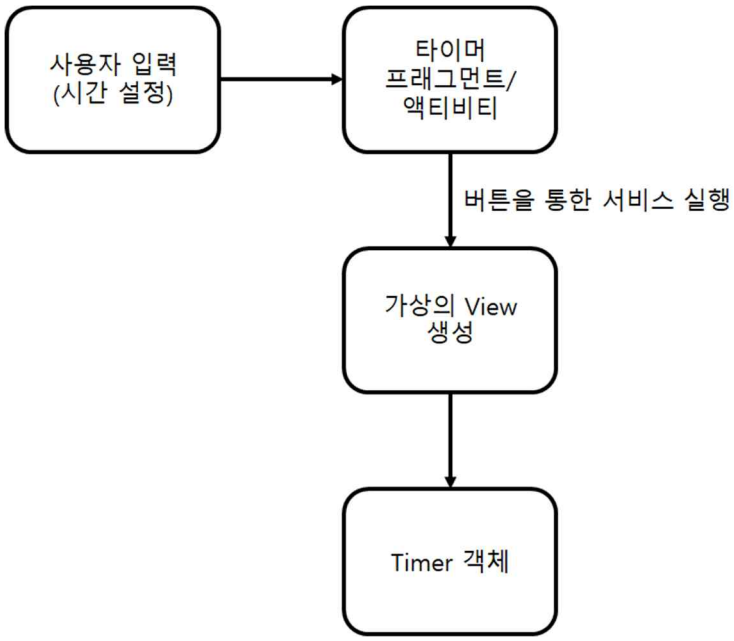

```java
package com.wmdev.floatingtimer;

import com.wmdev.floatingtimer.a.b;
import com.wmdev.floatingtimer.a.f;
import com.wmdev.floatingtimer.a.g;
import com.wmdev.floatingtimer.a.h;

/* access modifiers changed from: package-private */
public class c implements b.c {
    final /* synthetic */ MainActivity a;

    c(MainActivity mainActivity) {
        this.a = mainActivity;
    }

    @Override // com.wmdev.floatingtimer.a.b.c
    public void a(f fVar, g gVar) {
        if (!fVar.d()) {
            h a2 = gVar.a("pro");
            this.a.o = a2 != null;
            g.a(this.a.o);
            if (this.a.o) {
                this.a.setTitle(R.string.title_pro);
                this.a.n();
                return;
            }
            this.a.m();
        }
    }
}
```

위와 같이 디컴파일 시 class 명, 변수명으로 암호화가 되어있어 어플리케이션을 파악하기가 쉽지 않았습니다. 따라서 두번 째 어플리케이션을 디컴파일하여 원리를 파악하였습니다.

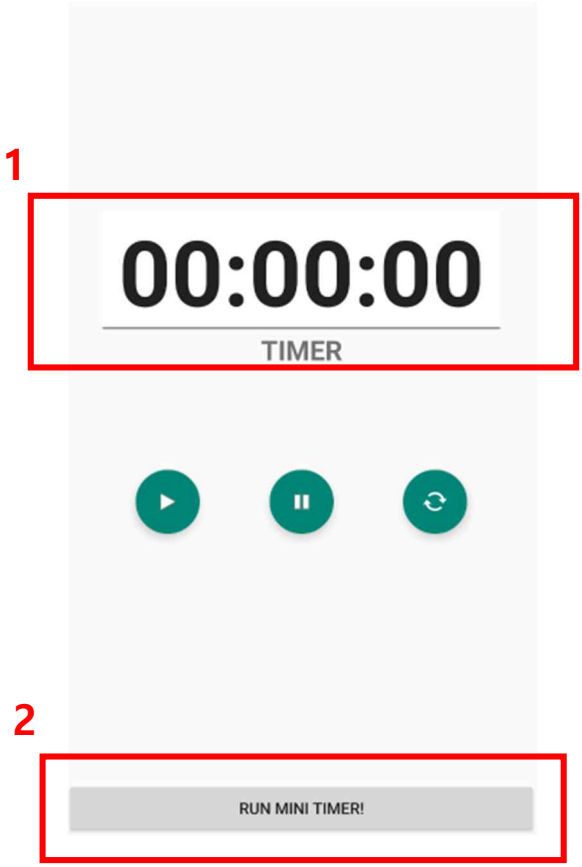


위와 같이 service 라는 폴더에 각 타이머의 객체를 정의해놨습니다. 제가 이걸 파악할 수 있는 이유는 에뮬레이터를 통하여 사용해본 결과 타이머는 최대 5 개까지 생성이 가능하였습니다. 이를 봐서 객체가 5 개 존재할 것이라 생각하였고 개수가 얼추 맞고 해당 기능을 수행하는 것처럼 보이는 class 를 찾았습니다. 위에서 파악한 원리를 찾아 검색해본 결과 타이머를 다른 앱들과 동시에 수행하기 위해서는 일반적인 방법이 아닌 service 부분에 코딩을 하여 사용한다는 것을 알아냈습니다.

구현 설명에 앞서 간단하게 그림으로 데이터 흐름을 표현해보도록 하겠습니다.



기본적으로 사용자에 보여주는 임의의 객체를 보여주기 위해서는 항상 view 위에 존재합니다.
다른 어플리케이션 위에서 타이머 객체를 생성하기 위해서는 다른 앱에 지장을 주지 않는 가상의 View 를 생성해줄 필요가 있으며 그 위에 작은 원형의 타이머 객체를 생성할 예정입니다.



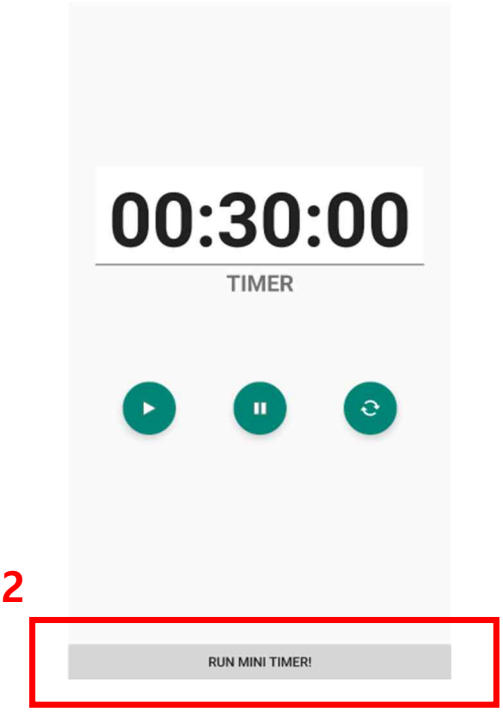
[타이머 프래그먼트/액티비티]

1 번의 빨간 부분을 클릭하게 되면 사용자가 원하는 시간을 입력 받을 수 있습니다.
안드로이드에서 지원해주는 입력창은 상당히 불편해 보여서 View 로 팝업창을 새로 만들었습니다.

[팝업창의 xml]

원하는 시간을 입력하게 되지만 빈칸으로 확인을 누르면 0 이 들어가고 90 초를 입력하면 알아서 1 분 30 초로 변하게 끔 설계되어 있습니다.

지금은 30 초를 입력했다고 가정하겠습니다.



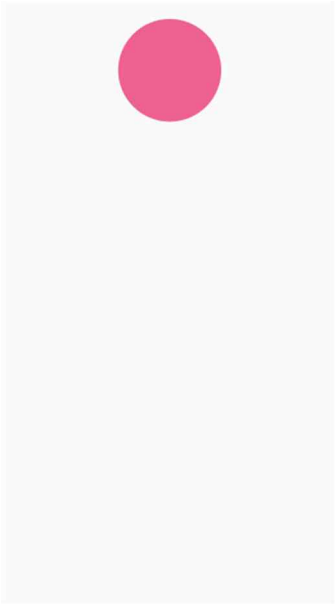
[30 초가 입력된 타이머 모습]

위와 같이 시간을 설정 후 버튼을 클릭하면 서비스가 실행됩니다. 서비스로 선언된 객체와 액티비티가 소통을 하기 위하여 **.bindService** 메소드로 쌍방향 소통을 선언해주고 **.startService** 메소드로 서비스 객체를 실행시켜줍니다. 해당 기능을 수행하는 코드는 다음과 같습니다.

```
@Override
public void onClick(View v) {
    if(!Settings.canDrawOverlays((requireContext())))
    {
        getPermission();
    }else{
        Log.d("e", "onClick: widgetbutton");
        Intent intent = new Intent(requireContext(), WidgetService.class);
        intent.putExtra("name" , time);
        //타이머를 생성해주는 서비스 호출 onstart 용
        requireContext().startService(intent);
        //통신을 위한 서비스 호출 onbind 용
        requireContext().bindService(intent, conn, Context.BIND_ABOVE_CLIENT);
        buttonAddWidget.setText("Already running");
        buttonAddWidget.setEnabled(false);
    }
}
```

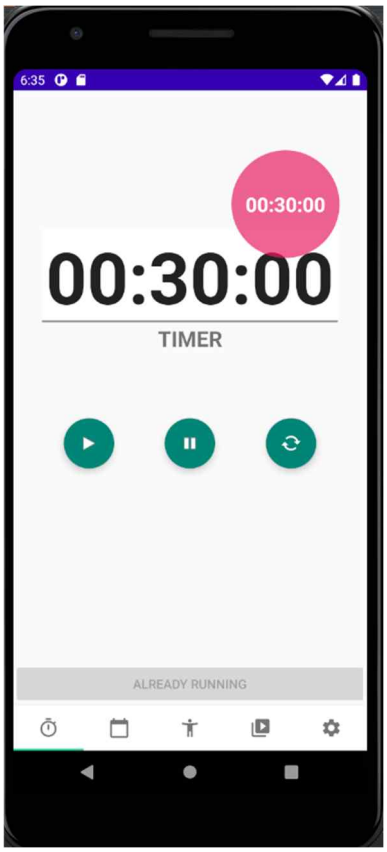
위의 intent 변수에 인자로 넘겨주는 WidgetService.class 가 실제 서비스 객체 클래스입니다. 다음은 WidgetService.class 에 대해 설명드리겠습니다.

Class 에 대해 설명하기 앞서 타이거 객체를 작동시키기 위한 View 단을 먼저 설명하겠습니다. View 형식은 다음과 같습니다.



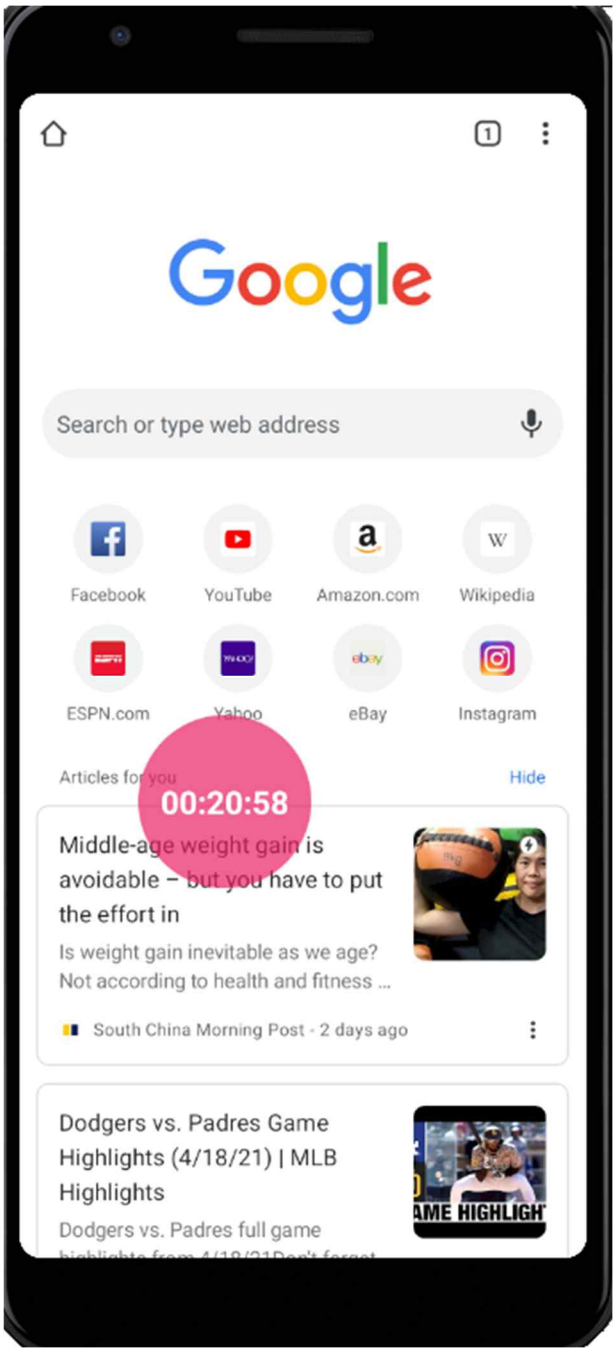
[widget view]

뒤의 배경은 실제 실행하면 가운데의 원 크기만큼 작아지며 타이머의 시간을 빨간 원안에 표시됩니다. 다음 사진과 같이 실행됩니다.

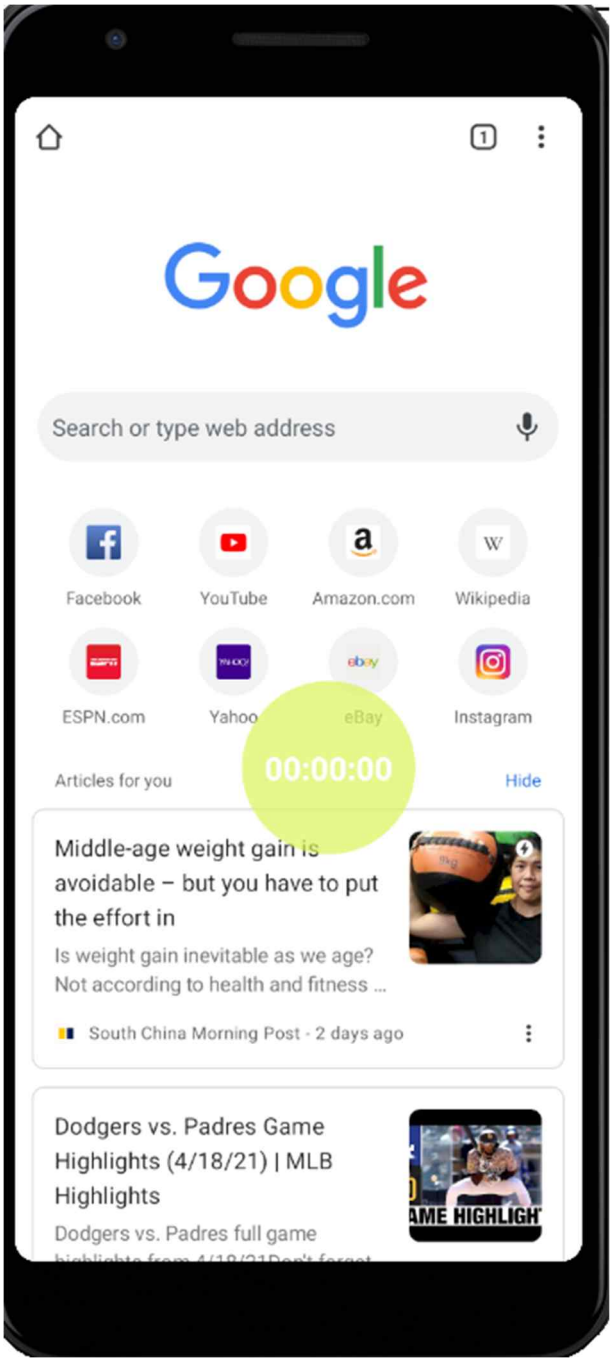


기본적인 인터페이스는 다음과 같습니다.

- 최초 한번 클릭 시 타이머 시작.
- 그 뒤 한번 클릭 시 중지
- 더블클릭 시 리셋
- 꾹 누르면 임의의 위치로 이동 가능.



[구글 위에서 실행되는 장면]



[타이머가 끝난 후]

코드는 타이머를 수행하는 부분과 인터페이스에 관하여 설명하겠습니다.

```
private void fn_countdown() {
    String timeInterval = tvWidget.getText().toString();
    endTime = time_to_seconds(timeInterval);


    countDownTimer = new CountDownTimer(endTime * 1000, 1) {
        @Override
        public void onTick(long millisUntilFinished) {
            int milli_seconds = (int) (millisUntilFinished % 100);
            int seconds = (int) (millisUntilFinished / 1000) % 60;
            int min = (int) ((millisUntilFinished / (1000 * 60)) % 60);

            String newtime = min + ":" + seconds + ":" + milli_seconds;
            System.out.println(millisUntilFinished);
            if (newtime.equals("0:0:0")) {
                tvWidget.setText("00:00:00");
            } else if ((String.valueOf(min).length() == 1) &&
(String.valueOf(seconds).length() == 1) && (String.valueOf(milli_seconds).length()
== 1)) {
                tvWidget.setText("0" + min + ":0" + seconds + ":0" +
milli_seconds);
            } else if ((String.valueOf(min).length() == 1) &&
(String.valueOf(seconds).length() == 1)) {
                tvWidget.setText("0" + min + ":0" + seconds + ":" + milli_seconds);
            } else if ((String.valueOf(min).length() == 1) &&
(String.valueOf(milli_seconds).length() == 1)) {
                tvWidget.setText("0" + min + ":" + seconds + ":0" + milli_seconds);
            } else if ((String.valueOf(seconds).length() == 1) &&
(String.valueOf(milli_seconds).length() == 1)) {
                tvWidget.setText(min + ":0" + seconds + ":0" + milli_seconds);
            } else if (String.valueOf(min).length() == 1) {
                tvWidget.setText("0" + min + ":" + seconds + ":" + milli_seconds);
            } else if (String.valueOf(seconds).length() == 1) {
                tvWidget.setText(min + ":0" + seconds + ":" + milli_seconds);
            } else if (String.valueOf(milli_seconds).length() == 1) {
                tvWidget.setText(min + ":" + seconds + ":0" + milli_seconds);
            } else {
                tvWidget.setText(min + ":" + seconds + ":" + milli_seconds);
            }

        }

        @Override
        public void onFinish() {
            tvWidget.setText("00:00:00");
            tvWidget.setBackgroundResource(R.drawable.end_circle);
        }
    };
    countDownTimer.start();
```

countDownTimer 를 import 해주고 분, 초, 밀리 초 단위로 계산해준 후 View 에 표시하기 위해 각각 String 으로 변경해주어서 맵핑시켜줍니다.

인터페이스는 크게 onTouch 함수 안에 정해집니다. Switch - case 문으로 정의하며 세 가지의 기준이 있습니다.

- ACTION_DOWN : 어떤 부분을 누를 당시를 정의하는 부분입니다.
- ACTION_UP : 어떤 부분을 누르고 손가락을 땔 때 정의하는 부분입니다.
- ACTION_MOVE : 객체를 움직일 때를 정의하는 부분입니다.

제가 정의한 인터페이스에는 더블 클릭도 정의했지만 onTouch 에서는 더블 클릭은 정의할 수 없습니다. 따라서 임의로 정의하였습니다. 객체서 손을 멀리할 때 시간과 다시 객체를 누를 때 시간이 일정한 시간보다 작으면 더블클릭이라 간주하고 작동하게 하였습니다.

```
@Override
    public boolean onTouch(View v, MotionEvent motionEvent) {

        switch (motionEvent.getAction())
        {
            case MotionEvent.ACTION_DOWN:

                startClickTime = Calendar.getInstance().getTimeInMillis();
                initialX = layoutParams.x;
                initialY = layoutParams.y;

                long double_tab = Calendar.getInstance().getTimeInMillis() -
dropClickTime;

                if(double_tab < MIN_CLICK_DURATION){
                    countDownTimer.cancel();
                    tvWidget.setText(original_time);
                    flag = 1;
                    tvWidget.setBackgroundResource(R.drawable.back_circle);
                }

                initialTouchX = motionEvent.getRawX();
                initialTouchY = motionEvent.getRawY();


                return true;
            case MotionEvent.ACTION_UP:
                imageClose.setVisibility(View.GONE);

                dropClickTime = Calendar.getInstance().getTimeInMillis();

                long clickDuration = Calendar.getInstance().getTimeInMillis()-
startClickTime;


                layoutParams.x = initialX + (int)(initialTouchX -
motionEvent.getRawX());
                layoutParams.y = initialY + (int)(motionEvent.getRawY() -
initialTouchY);
```

```
                if(clickDuration < MAX_CLICK_DURATION)
                {
                    if(flag == 0){
                        fn_countdown();
                        flag = 1;
                    }
                    else if (flag == 1)
                    {
                        countDownTimer.cancel();
                        flag = 0;
                    }

                }else{
                    if(layoutParams.y > (height*0.6))
                    {
                        stopSelf();
                    }
                }
                return true;

            case MotionEvent.ACTION_MOVE:
                imageClose.setVisibility(View.VISIBLE);
                //claculate X & Y cordinates of view
                layoutParams.x = initialX + (int)(initialTouchX -
motionEvent.getRawX());
                layoutParams.y = initialY + (int)(motionEvent.getRawY() -
initialTouchY);

                //update layout with new condinates
                windowManager.updateViewLayout(mFloatingView, layoutParams);

                if(layoutParams.y > (height*0.6))
                {
                    imageClose.setImageResource(R.drawable.close_black);
                }else{
                    imageClose.setImageResource(R.drawable.close_white);
                }


                return true;
        }

        return false;
    }
});
```

다음 주는 중간보고서 발표준비와 함께 타이머 기능을 마무리할 생각입니다.

Github : https://github.com/hyeonjun414/Homenagement