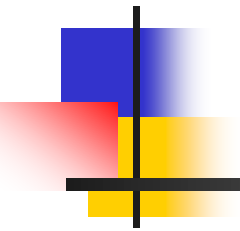
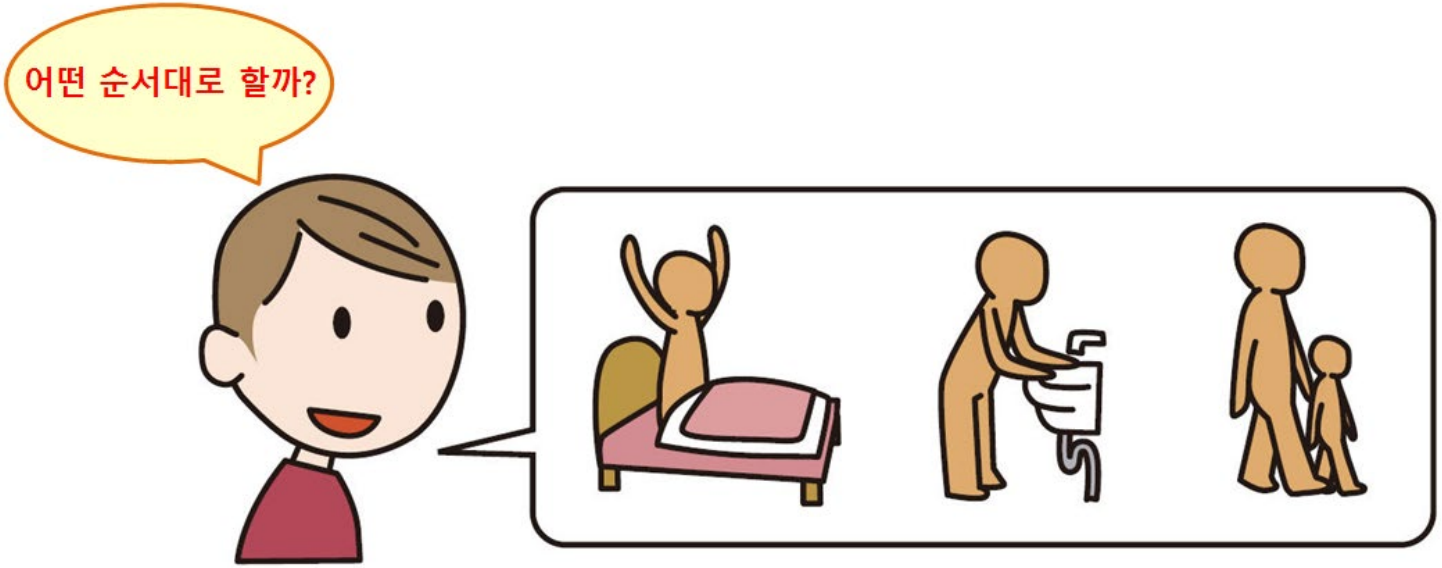


Process와 Thread

배 희호 교수
경복대학교
소프트웨어융합과

Program과 Thread

- Program
  - 어떤 기능을 어떤 순서대로 처리할 것인가를 정리하는 Logic



Program과 Thread

- 일과를 알려주는 App 기획하기

Program과 Thread

- 강아지와 원반 던지기 놀이하기

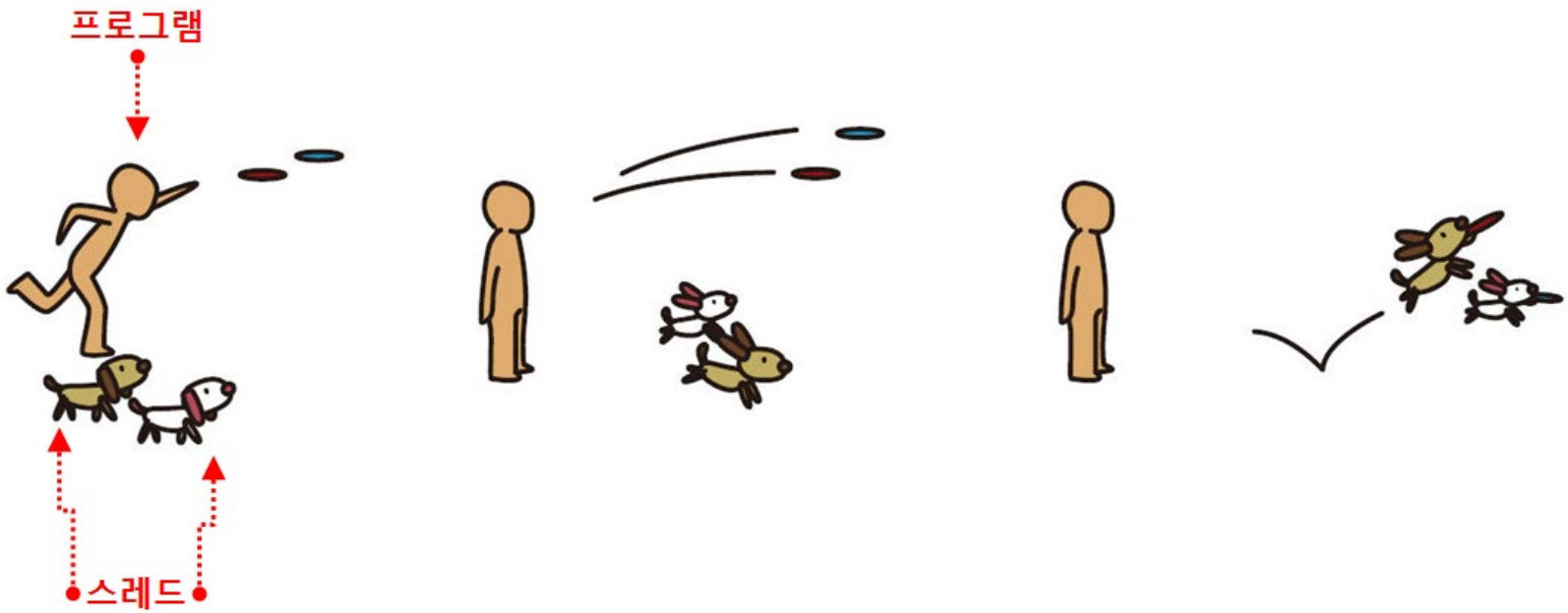
- Thread
  - Program 내부에서 실행의 단위
  - 하나의 Program은 여러 개의 Thfread로 나눌 수 있으며, 각 Thread는 독립적으로 실행

Program과 Thread

- 두 마리의 강아지가 동시에 원반을 가지고 오기

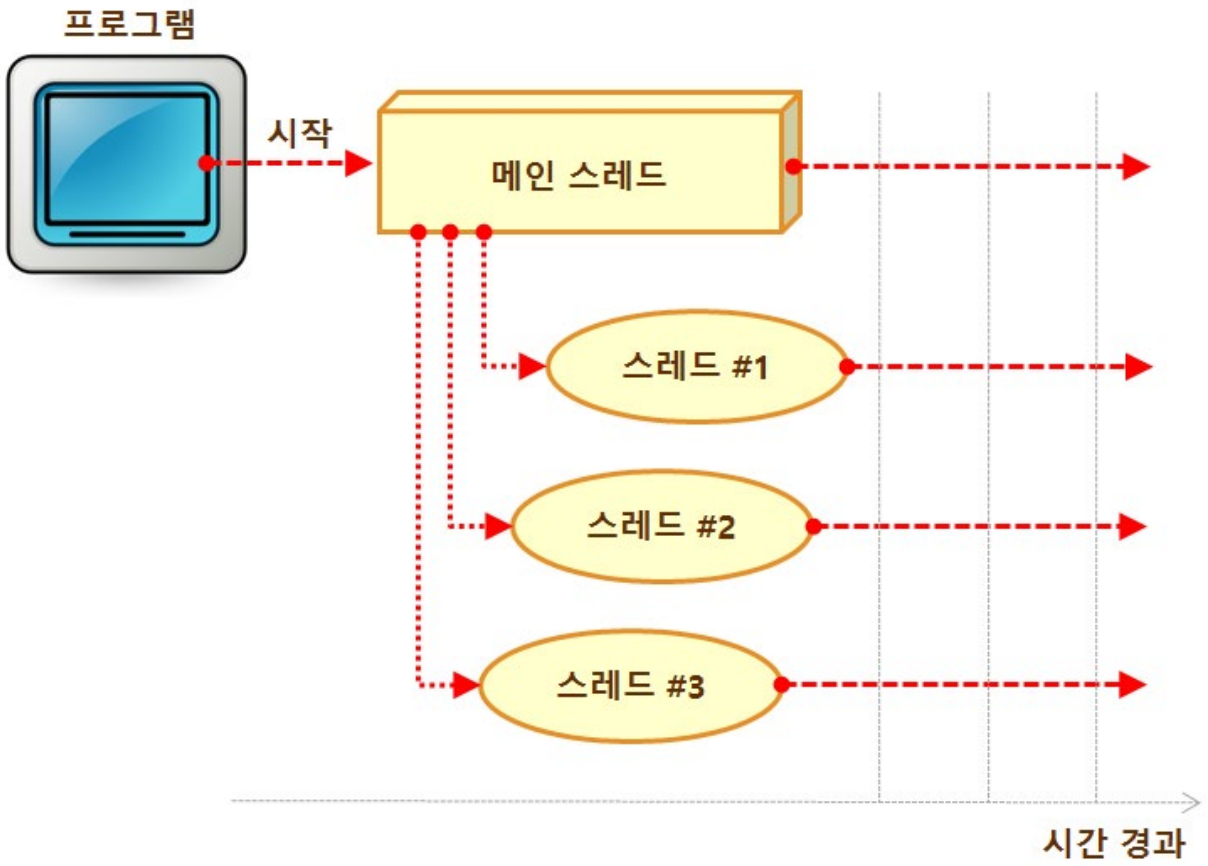
Program과 Thread

- 동시 작업을 하는 Thread



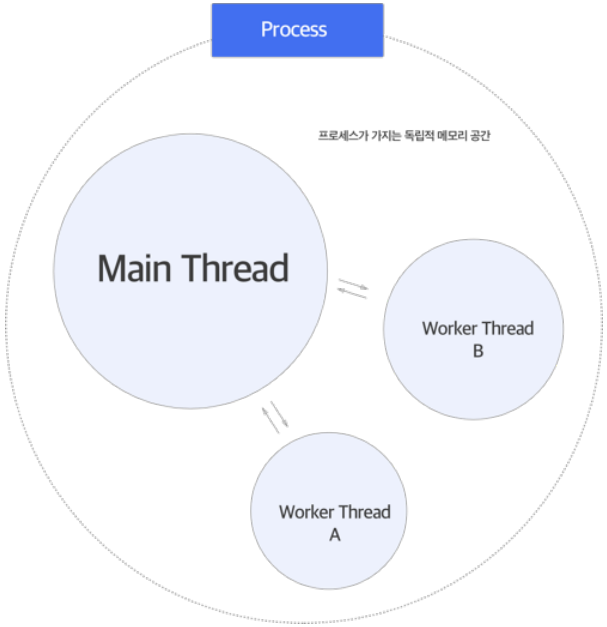
Program과 Thread

- Program과 Thread



Thread

- Android에서 Thread는 Application이 여러 작업을 병렬로 수행할 수 있도록 도와주는 중요한 개념
- Android는 기본적으로 Single Thread 환경에서 작동하며, Application의 기본 Thread는 UI Thread 또는 Main Thread라고 불림
- 이 Thread는 사용자 Interface(UI)와 상호 작용하고 화면을 Update하는 작업을 담당함



Thread

- Android Thread 종류
  - UI Thread/Main Thread
    - App 실행 시 생성되는 기본 Thread
    - UI 관련 작업(View Update, 사용자 입력 처리 등) 담당
    - 무거운 작업(Network 요청, File 처리 등)을 이 Thread에서 실행하면 App이 멈추거나 “App이 응답하지 않음(ANR)"이 발생할 수 있음
  - Worker Thread
    - 특정한 작업을 위하여 Program에서 만들어지는 Thread들
    - Background에서 실행되며, UI Update와는 독립적
    - 무거운 작업을 처리하여 UI Thread가 원활히 작동하도록 지원
    - Android에서는 UI 요소를 직접 Worker Thread에서 수정할 수 없으므로, UI Update는 UI Thread로 전달 함

Thread

- Android Thread를 위한 Class
  - JAVA의 기본 Thread Class
    - Thread Class를 확장하거나, Runnable Interface를 구현하여 사용
  - Handler
    - Worker Thread에서 처리된 결과를 UI Thread로 전달할 때 사용
  - AsyncTask (Deprecated in API 30)
    - UI Thread와 Worker Thread 간 작업 분리를 쉽게 할 수 있도록 도와주는 Class
    - 현재는 WorkManager, ExecutorService 등의 대체 기술을 권장
  - Executor 및 ExecutorService
    - ThreadPool을 관리하며 효율적인 Thread 실행을 지원

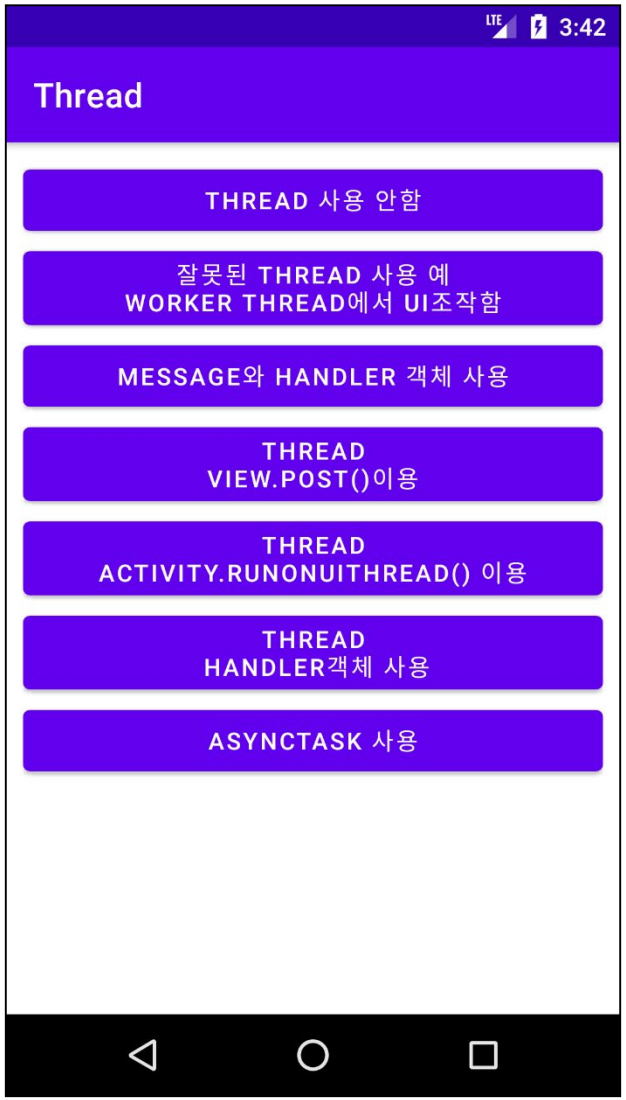
Thread

- Android Thread를 위한 Class
  - HandlerThread
    - Background 작업과 Message 처리를 위해 설계된 Thread
  - WorkManager
    - 장기 실행 작업이나 Background 작업을 관리하기 위한 Android Jetpack의 구성 요소
    - Network 요청, Data 동기화 등 지속적인 작업에 적합

Thread

- 주의 사항
  - UI 관련 작업은 반드시 UI Thread에서 처리해야 함. 그렇지 않으면 예외(ViewRootImpl$CalledFromWrongThreadException)가 발생
  - 무거운(시간이 오래 걸리는) 작업(File 처리, Network 통신 등)은 Worker Thread에서 처리해야 함. UI Thread에서 실행 시 ANR 발생 가능
  - Thread 관리는 복잡할 수 있으므로, Android의 Jetpack Library(예: WorkManager 또는 Coroutines와 같은 Kotlin 도구)를 적극 활용하는 것이 좋음

Thread 예제



Thread 예제

- 사용자 인터페이스

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Thread 사용 안함" />
```

Thread 예제

- 사용자 인터페이스

```
<Button
    android:id="@+id/button2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="잘못된 Thread 사용 예₩nWorker Thread에서 UI조작함" />

<Button
    android:id="@+id/button3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Thread₩nView.post()이용" />

<Button
    android:id="@+id/button4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Thread₩nActivity.runOnUiThread() 이용" />
```

Thread 예제

- 사용자 인터페이스

```
    <Button
        android:id="@+id/button5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Handler객체 사용" />

    <Button
        android:id="@+id/button6"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Thread₩n Handler객체 사용" />

    <Button
        android:id="@+id/button7"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="AsyncTask 사용" />
</LinearLayout>
```

Thread 예제

- Mainactivity.JAVA

```
public class MainActivity extends AppCompatActivity
                                        implements View.OnClickListener {

    Button button1, button2, button3, button4, button5, button6, button7;

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

        button1 = findViewById(R.id.button1);
        button2 = findViewById(R.id.button2);
        button3 = findViewById(R.id.button3);
        button4 = findViewById(R.id.button4);
        button5 = findViewById(R.id.button5);
        button6 = findViewById(R.id.button6);
        button7 = findViewById(R.id.button7);
```

Thread 예제

- Mainactivity.JAVA

```
        button1.setOnClickListener(this);
        button2.setOnClickListener(this);
        button3.setOnClickListener(this);
        button4.setOnClickListener(this);
        button5.setOnClickListener(this);
        button6.setOnClickListener(this);
        button7.setOnClickListener(this);
    }

    @SuppressLint("SetTextI18n")
    @Override
    public void onClick(View view) {
        Intent intent;
        if (view == button1) {
            intent = new Intent(getBaseContext(), DigitalClockActivity1.class);
        } else if (view == button2) {
            intent = new Intent(getBaseContext(), DigitalClockActivity2.class);
        } else if (view == button3) {
```

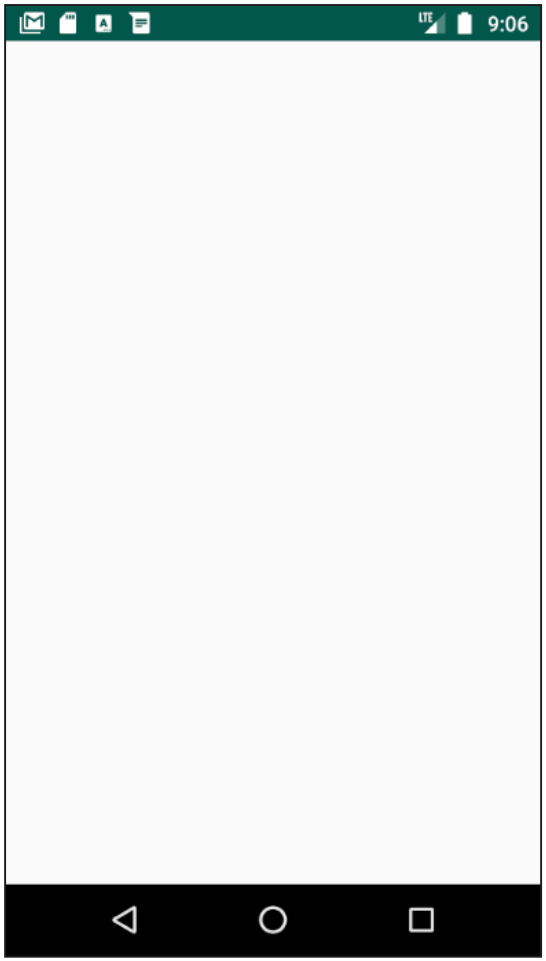
Thread 예제

- Mainactivity.JAVA

```
            intent = new Intent(getBaseContext(), DigitalClockActivity3.class);
        } else if (view == button4) {
            intent = new Intent(getBaseContext(), DigitalClockActivity4.class);
        } else if (view == button5) {
            intent = new Intent(getBaseContext(), DigitalClockActivity5.class);
        } else if (view == button6) {
            intent = new Intent(getBaseContext(), DigitalClockActivity7.class);
        } else
            intent = new Intent(getBaseContext(), DigitalClockActivity6.class);
        startActivity(intent);
    }
}
```

Thread 예제

- Thread를 사용하지 않은 경우



Thread 예제

- Calendar 클래스 사용해서 간단하게 오늘 날짜 가져오기
  - Calendar 객체 생성

```
Calendar cal = Calendar.getInstance();
```

  - SimpleDateFormat 객체 생성

```
SimpleDateFormat format = new
                SimpleDateFormat("yyyy-MM/dd");

String date =
    format.format(Calendar.getInstance().getTime());
```

Thread 예제

- SimpleDateFormat 기호

기호	의미	보기
G	연대(BC, AD)	AD
y	년도	2009
M	월 (1~12월 또는 1월~12월)	10또는 10월, OCT
E	요일	월
a	오전/오후(AM, PM)	PM
H	시간(0~23)	20
k	시간(1~24)	12
K	시간(0~11)	10
h	시간(1~12)	11
m	분(0~59)	35
s	초(0~59)	55
S	천분의 1초(0~999)	253

Thread 예제

- SimpleDateFormat 기호

기호	의미	보기
w	년의 몇 번째 주(1~53)	50
W	월의 몇 번째 주(1~5)	4
D	년의 몇 번째 일(1~366)	100
d	월의 몇 번째 일(1~31)	15
F	월의 몇번째 요일(1~5)	1
z	Time zone(General time zone)	GMT+9:00
Z	Time zone(RFC 822 time zone)	+0900
'	escape문자 (특수문자를 표현하는데 사용)	없음

Thread 예제

- 사용자 인터페이스

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/watch"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:textSize="50dp"
        android:text="AM 00:00:00" />
</LinearLayout>
```

Thread 예제

- DigitalClockActivity1.JAVA

```
public class DigitalClockActivity1 extends AppCompatActivity {
    private TextView clock;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_digitalclock1);

        clock = findViewById(R.id.watch) ;
        SimpleDateFormat form = new SimpleDateFormat("a HH:mm:ss") ;
```

Thread 예제

- DigitalClockActivity1.JAVA

```
        while (true) {
            Calendar calendar = Calendar.getInstance() ;
            String time = form.format(calendar.getTime());
            clock.setText(time);
            try {
                Thread.sleep(1000);
            } catch (Exception e) {
                Toast.makeText(this, e.getMessage(), Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

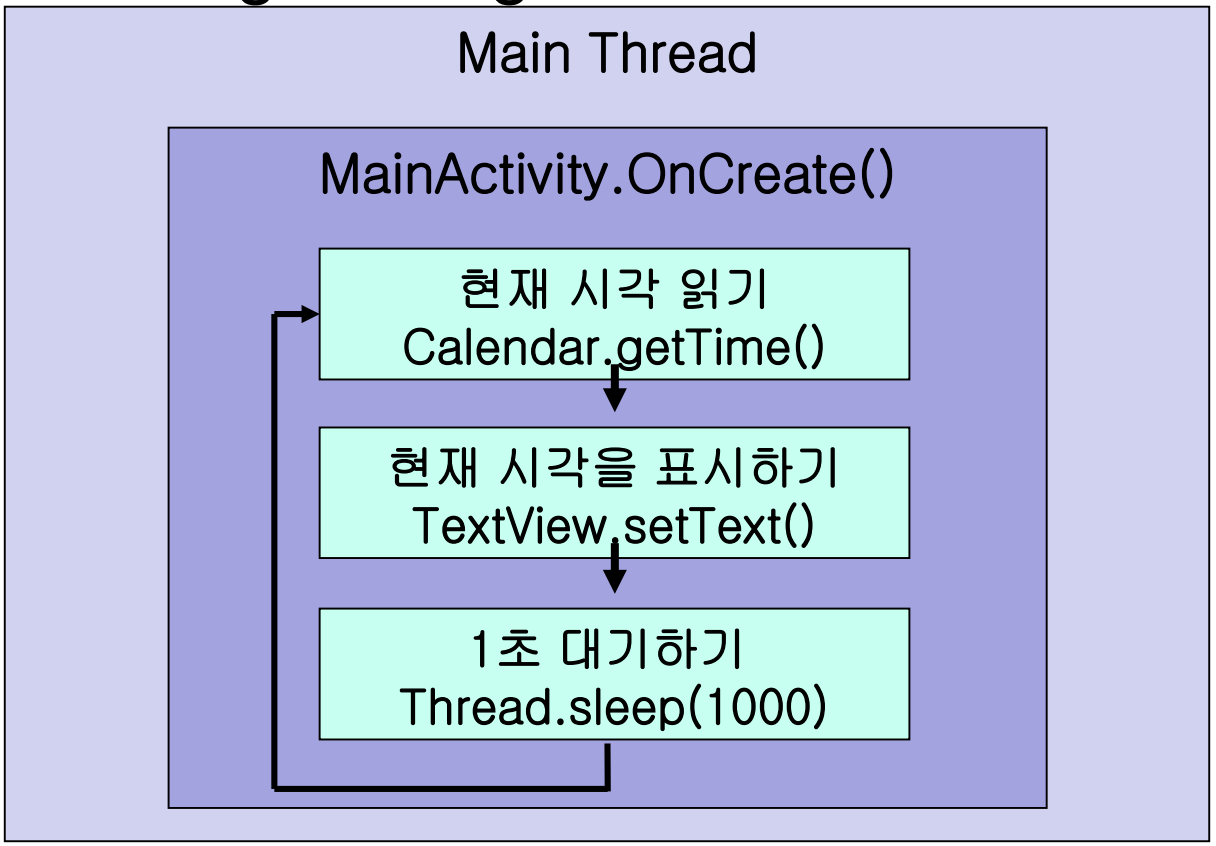
Main Thread에서 주기적인 UI 갱신
마지막 한번만 수행하는 것처럼 동작

Thread 예제

- 실행중인 Thread를 잠시 멈추게 하고 싶다면 Thread Class의 static Method인 sleep() Method를 사용하면 됨
  - Thread.sleep() Method를 호출한 Thread는 주어진 시간 동안 일시 정지 상태가 되고 다시 실행 대기 상태로 돌아감
  - 매개 변수 값에는 얼마 동안 일시 정지 상태로 있을 것인지 millisecond(1/1000) 단위로 시간을 알려주면 됨
  - 1,000이라는 값을 주면 Thread는 1초 동안 일시 정지 상태가 됨
  - 일시 정지 상태에서 주어진 시간이 되기 전에 interrupt() Method가 호출되면 InterruptedException이 발생하기 때문에 예외 처리가 필요함

Thread 예제

- 다음과 같이 Programming하면 안 되는 이유는 ?



- Main Thread에서는 시간이 오래 걸리는 작업을 하면 Android는 ANR을 발생시켜 자동으로 실행을 멈춤 따라서 시간이 오래 걸리는 작업은 별도의 Worker Thread를 만들어 처리해야 함

Thread 예제

- Main Thread의 한계
  - 개발자가 별도의 Worker Thread를 만들지 않는 이상 구현되는 모든 Code는 Main Thread에서 동작함
  - 그러나 아쉽게도 실질적으로 모든 작업을 Main Thread에서 처리할 수는 없음

  - 그렇다면 해결 방법은 무엇일까?
    - 간단함
    - Main Thread에서 무거운(시간이 오래 걸리는) 작업을 처리하지 말아야 함
    - 즉 긴 작업은 모두 Worker Thread에서 처리하면 됨

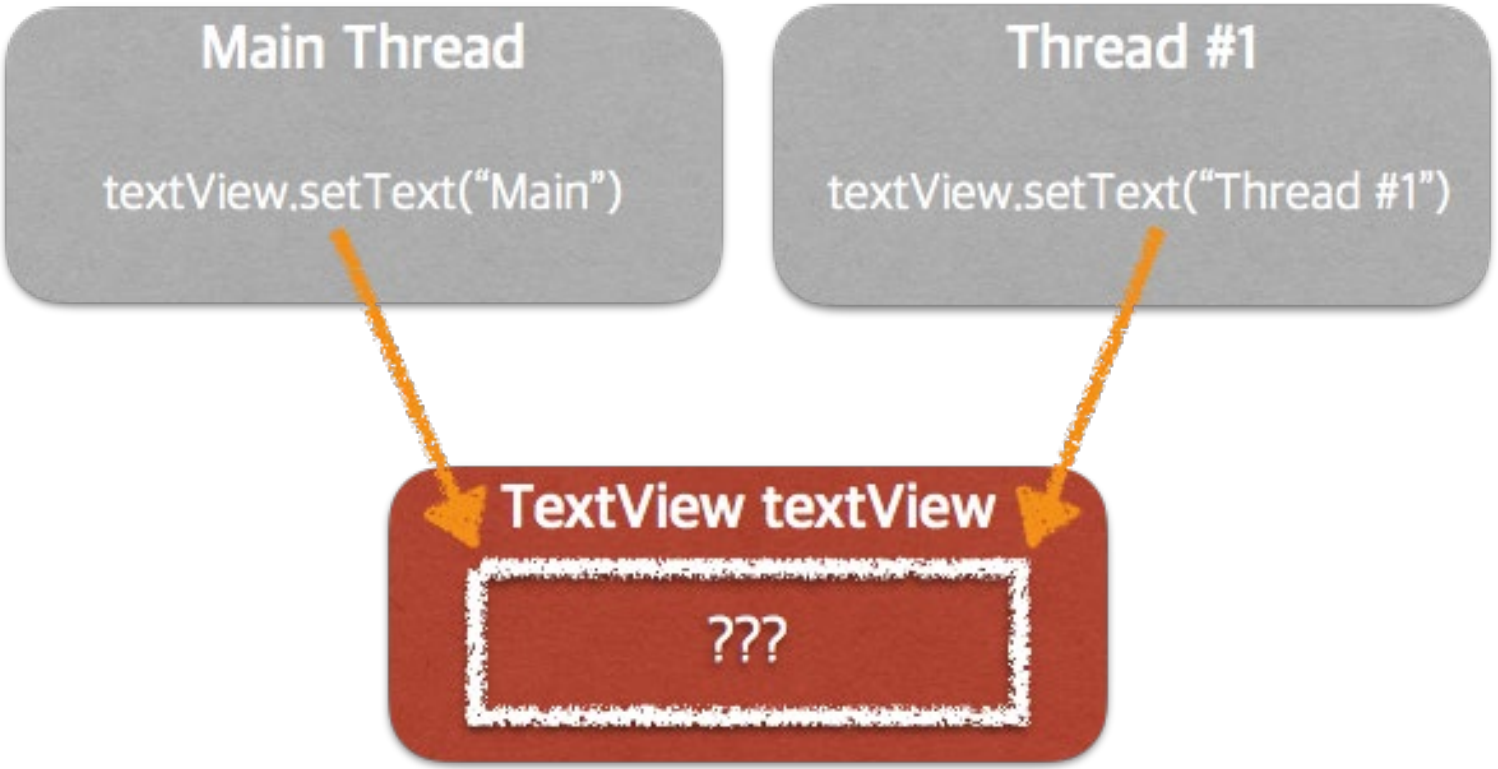
Thread 예제

- Main Thread의 한계
  - UI/Main Thread가 시간 오래 걸리는 작업을 하면 응답성이 떨어짐
  - 시간 오래 걸리는 작업은 별도의 Thread를 만들어 그 Thread가 수행하도록 함
    - Network 송수신
    - File 읽기 쓰기
    - 복잡한 계산 등
    - 별도의 Thread를 Worker Thread라고 부름

  - 여러 Thread로 작업할 때 어려운 점
    - Worker Thread는 UI Component에 접근하지 못함
      - Worker Thread가 한 일의 결과를 UI에 반영하려면 UI Thread의 도움을 받아야 함

Thread 예제

- Android는 Main Thread에서 만 UI 작업이 가능하도록 왜 제한할까요?



Thread 예제

- Worker Thread의 한계
  - Android에서는 Main Thread가 아닌 다른 Thread들에서 화면에 그리는 작업을 허용하지 않음
    - 여러 Thread에서 View를 변경하여 화면을 갱신한다면 동기화 문제가 발생될 수 있기 때문임
    - 동시에 실행되는 Thread는 어느 것이 더 빨리 처리될지 순서를 알 수 없고, 화면에 View를 그리는 것은 순서가 매우 중요함
    - 그러므로 처리되는 순서가 불규칙한 Thread에서 그리는 작업을 하면 화면은 뒤죽박죽이 될 수 있음
    - Android에서는 Main Thread에서만 그리는 작업을 허용하여 그리는 순서를 보장하며, 이를 Single Thread GUI(Graphical User Interface) Model이라 함

Thread 예제(Worker Thread)

- Thread를 사용한 경우(Digital 시계 출력을 Thread에서)

Thread 예제(Worker Thread)

- DigitalClockActivity2.JAVA

```
public class DigitalClockActivity2 extends AppCompatActivity {
    private int type = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_digitalclock1);
    }
```

Thread 예제(Worker Thread)

- DigitalClockActivity2.JAVA

```
@Override
protected void onStart() {
    super.onStart();
    TextView clock = findViewById(R.id.watch);
    if (type == 1) {
        TimerThread thread = new TimerThread(clock);
        thread.start();
    } else {
        TimerRunnable runnable = new TimerRunnable(clock);
        Thread thread = new Thread(runnable);
        thread.start();
    }
}
```

Thread 예제(Worker Thread)

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        switch (item.getItemId()) {
            case R.id.item1 :
                type = 1;
                break;
            case R.id.item2 :
                type = 2;
        }
        item.setChecked(true);
        onStart();
        return true;
    }
}
```

Thread 예제(Worker Thread)

- TimerThread.JAVA

```
public class TimerThread extends Thread {
    private TextView textView;
    private SimpleDateFormat form;

    public TimerThread(TextView textView) {
        this.textView = textView;
        form = new SimpleDateFormat("a HH:mm:ss");
        textView.setTextColor(Color.BLACK);
    }
```

Thread 예제(Worker Thread)

- TimerThread.JAVA

```
    @Override
    public void run() {
        while (true) {
            Calendar calendar = Calendar.getInstance();
            String time = form.format(calendar.getTime());
            textView.setText(time) ;
            try {
                Thread.sleep(1000) ;
            } catch (Exception e) {
                Toast.makeText(textView.getContext(), e.getMessage(),
                                                    Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

Thread 예제(Worker Thread)

- TimerRunnable.JAVA

```
public class TimerRunnable implements Runnable{
    private TextView textView;
    private SimpleDateFormat form;

    public TimerRunnable(TextView textView) {
        this.textView = textView;
        textView.setTextColor(Color.BLUE);
        form = new SimpleDateFormat("a HH:mm:ss", Locale.KOREA);
    }
```

Thread 예제(Worker Thread)

- TimerRunnable.JAVA

```
@Override
public void run() {
    while (true) {
        Calendar calendar = Calendar.getInstance();
        String time = form.format(calendar.getTime());
        textView.setText(time);
        try {
            Thread.sleep(1000) ;
        } catch (Exception e) {
            Toast.makeText(textView.getContext(), e.getMessage(),
                                                    Toast.LENGTH_SHORT).show();
        }
    }
}
}
```

Work Thread에서 주기적인 UI 갱신
하려고 해서 바로 죽음

Thread 예제(Worker Thread)

- menu.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group android:checkableBehavior="single">
        <item
            android:id="@+id/item1"
            android:checked="true"
            android:title="Thread" />
        <item
            android:id="@+id/item2"
            android:title="Runnable" />
    </group>
</menu>
```

Thread 예제 1(Work Thread)

- ClockThread.JAVA

```
    @Override
    public void run() {
        SimpleDateFormat dateFormat = new SimpleDateFormat("HH:mm:ss");
        while (true) {
            Calendar calendar = Calendar.getInstance();
            String time = dateFormat.format(calendar.getTime());
            textView.setText(time);
            try {
                Thread.sleep(1000);
            } catch (Exception e) {
                Toast.makeText(context, e.getMessage(),
                                                    Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

Thread 예제 1(Work Thread)

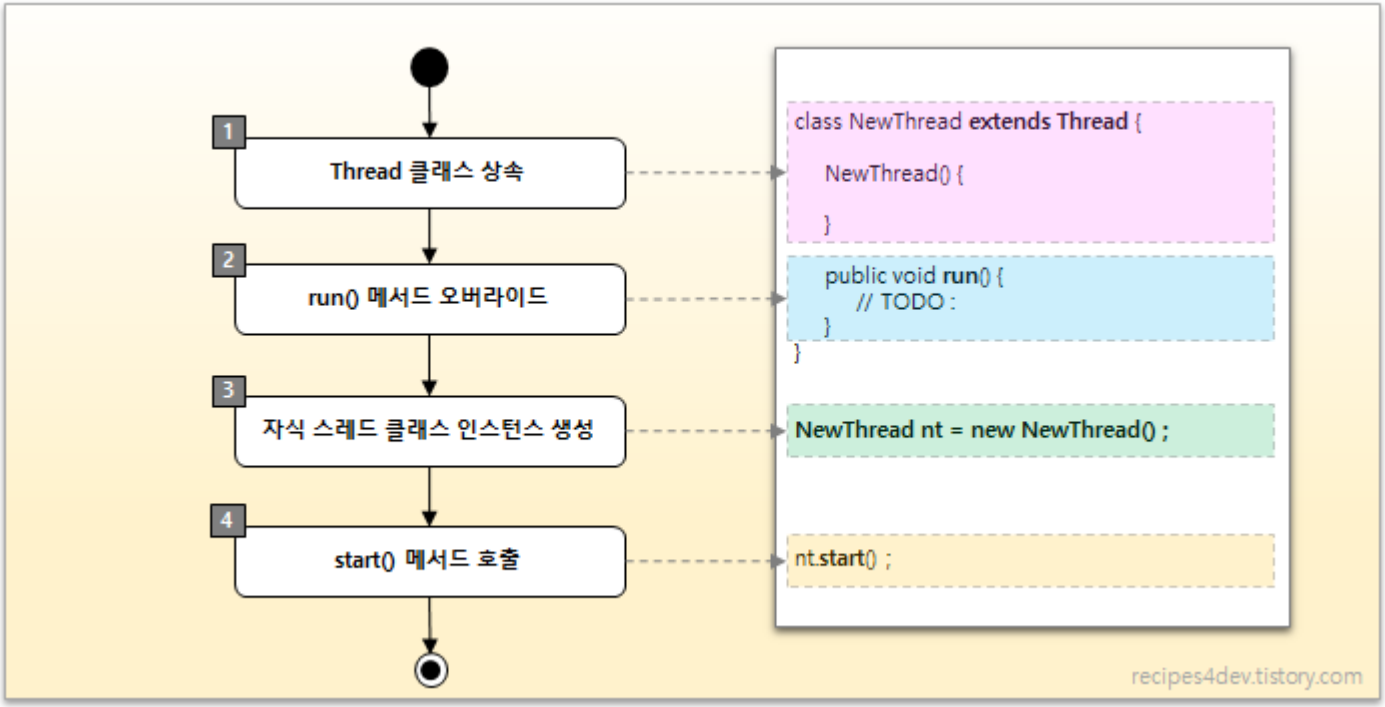
- Thread 만드는 2가지 방법
  - Thread 상속(extends) 방법
  - Runnable Interface 구현(implements)

- Program이 실행하면 바로 죽어버리는 원인은 ?
  - Activity의 UI는 Main Thread에서만 담당하고, 모든 Work Thread는 필요한 UI를 Handler를 통하여 Main Thread에서 처리할 수 있도록 각종 통신 방법을 이용하여 요구하여야 함

- Main Thread와 Work Thread사이의 통신 방법
  - Handler Message 사용
  - post() 메소드 사용
  - runOnUiThread() 메소드 사용

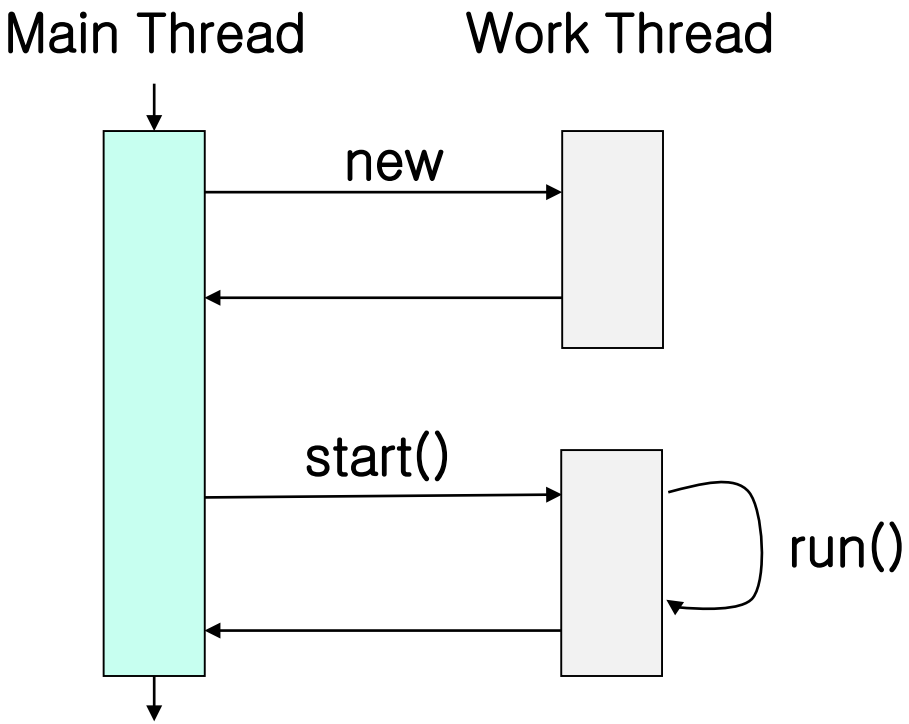
Thread 예제 1(Work Thread)

- Thread 상속(extends) 방법–White Box
  - Thread Class를 상속 받는 하위 Thread Class 생성
  - 하위 Thread에 run( )을 Override 함
    - run( )은 Thread가 실행되면 수행되는 곳임
  - Thread Object를 생성해주고 start( ) Method로 Tread의 run( ) Method를 실행시킴



Thread 예제 1(Work Thread)

- Thread 상속(extends) 방법 – White Box
  - Thread() 생성자로 만들어서 내부적으로 run()을 구현



Thread 예제 1(Work Thread)

- Thread 상속(extends) 방법–White Box
  - Thread() 생성자로 만들어서 내부적으로 run()을 구현

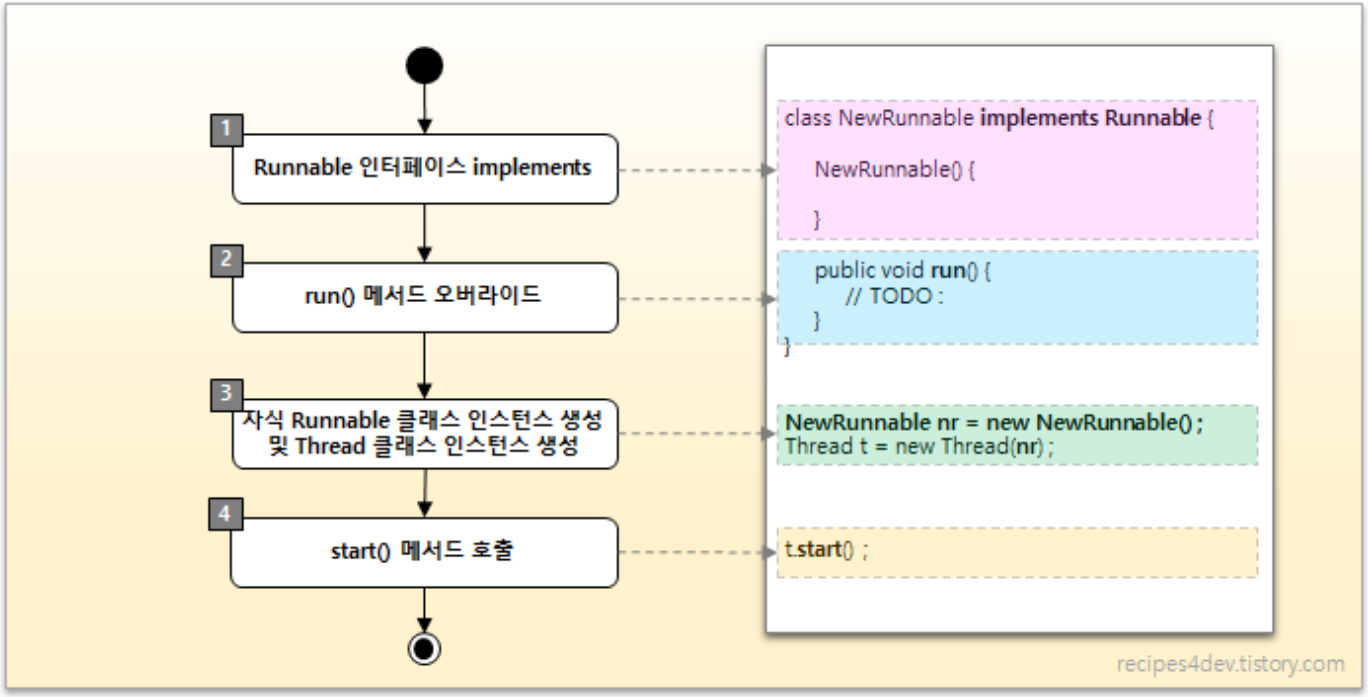
```
private class BackThread extends Thread {
    @Override
    public void run(){
        //구현 내용
    }
}
```

  - Thread를 사용할 Class를 만들었으니 사용하려면 Class를 만들어야 함

```
BackThread thread = new BackThread();
thread.start();
```

Thread 예제 1(Work Thread)

- Runnable Interface 구현(implements) – Black Box
  - Runnable Interface를 구현하는 Class를 선언
  - run() Method를 Override
  - Class Instance를 Thread Class Instance의 생성자에 전달하고 Thread Class Instance 생성
  - start() Method를 호출하면 됨



Thread 예제 1(Work Thread)

- Runnable Interface 구현(implements) -Black-Box
  - Thread(Runnable runnable) 생성자

```
private class Backthread implements Runnable {
        @Override
        public void run() {
            //구현 내용
        }
}
```

  - 작업이 끝난 후 UI 변경이 필요 없는 경우를 위한 사용법
  - Thread Class를 만들고 run( ) Method를 구현해주면 됨

```
BackThread backthread = new BackThread();
Thread thread = new Thread(backthread);
thread.start();
```

Thread 예제 1(Work Thread)

- Runnable Interface 구현(implements) -Black-Box
  - Thread(Runnable runnable) 생성자로 만들어서 Runnable Interface를 구현한 Object를 생성하여 전달
    - Runnable로 Thread의 run() Method를 분리한 것
    - Runnable Interface는 run() Abstract Method를 가지고 있으므로 상속받은 Class는 run() Code를 반드시 구현해야 함
    - JAVA는 다중 상속을 지원하지 않기 때문에 이러한 방식에서는 MyThread Class는 Thread Class 이외의 Ckass로 부터 상속을 받을 수가 없게 됨
    - Runnable Interface를 구현하여 Thread를 사용하는 방법이 많이 이용됨

Thread 예제 1(Work Thread)

- Thread 메소드

static void sleep(long msec) throws Interrupted Exception	msec에 지정된 밀리초 동안 대기
String getName()	Thread의 이름 가져오기
void setName(String s)	Thread의 이름을 s로 설정
void start()	Thread를 시작 run() 메소드 호출
int getPriority()	Thread의 우선 순위를 반환
void setpriority(int p)	Thread의 우선순위를 p값으로
boolean isAlive()	Thread가 시작되었고 아직 끝나지 않았으면 true 끝났으면 false 반환
void join() throws InterruptedException	Thread가 끝날 때 까지 대기
void run()	Thread가 실행할 부분 기술 (오버라이딩 사용)
void suspend()	Thread가 일시 정지 resume()에 의해 다시 시작 할 수 있음
void resume()	일시 정지된 Thread를 다시 시작
void yield()	다른 Thread에게 실행 상태를 양보하고 자신은 준비 상태로 변경

Thread 예제 1(Work Thread)

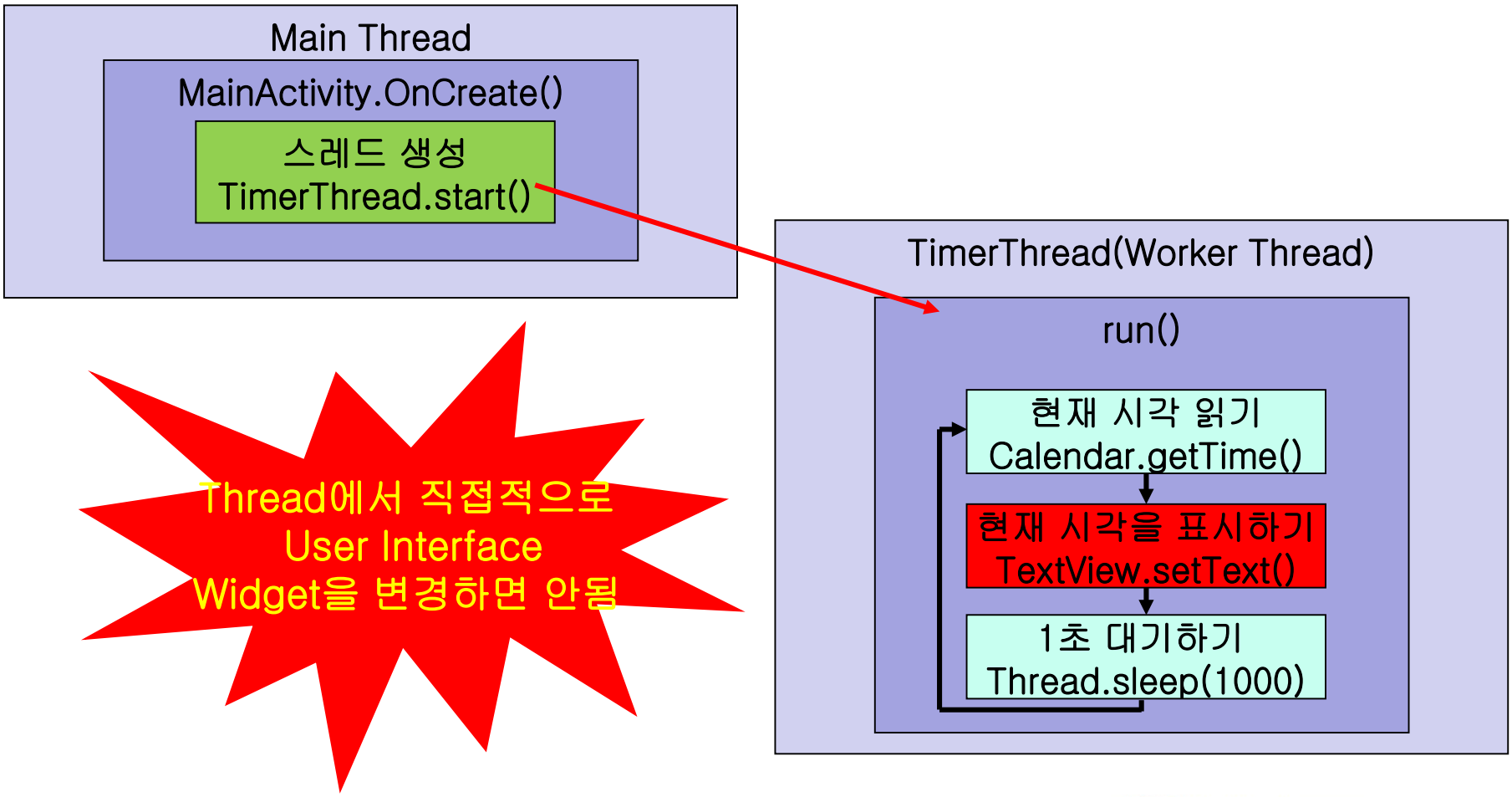
- Thread vs Runnable

항목	Runnable Interface 구현	Thread Class 상속
코드	implements Runnable	extends Thread
범위	단순히 run() Method만 구현하는 경우	Thread Class의 기능 확장이 필요한 경우
설계	논리적으로 분리된 태스크(Task) 설계에 장점	태스크(Task)의 세부적인 기능 수정 및 추가에 장점
상속	Runnable Interface에 대한 구현이 간결	Thread Class 상속에 따른 Overhead

- Thread Class를 상속(extends)해서 만든 Thread와 Runnable Interface를 구현(implements)해서 만든 Thread는, 작성된 run() 메소드 Code의 실행과 성능은 동일함

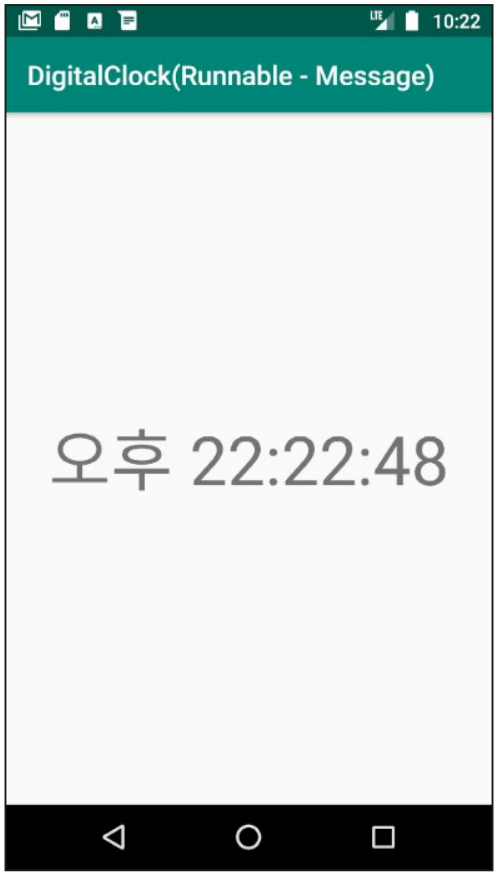
Thread 예제 1(Work Thread)

- 다음과 같이 Programming하면 안 되는 이유는 ?



Thread 예제(Message)

- Main Thread와 Work Thread사이의 통신 방법으로 Message를 사용



Thread 예제(Message)

- DigitalClockActivity3.JAVA

```
public class DigitalClockActivity3 extends AppCompatActivity {
    private int type = 1;
    private TextView clock;
    private Thread thread;
    private TimerThread1 thread1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_digitalclock1);
    }
```

Thread 예제(Message)

- DigitalClockActivity3.JAVA

```
@Override
protected void onStart() {
    super.onStart();
    clock = findViewById(R.id.watch);
    if (type == 1) {
        if (thread != null && thread.isAlive())
            thread.interrupt();
        thread1 = new TimerThread1(handler);
        thread1.start();
    } else {
        if (thread1 != null && thread1.isAlive())
            thread1.interrupt();
        TimerRunnable1 runnable = new TimerRunnable1(handler);
        thread = new Thread(runnable);
        thread.start();
    }
}
```

Thread 예제(Message)

- DigitalClockActivity3.JAVA

```
Handler handler = new Handler(Looper.getMainLooper()) {
    @Override
    public void handleMessage(Message msg) {
        SimpleDateFormat form = null;
        if (msg.what == 0) {
            form = new SimpleDateFormat("a HH:mm:ss");
            clock.setTextColor(Color.BLACK);
        } else if (msg.what == 1) {
            form = new SimpleDateFormat("a HH:mm:ss", Locale.KOREA);
            clock.setTextColor(Color.BLUE);
        }
        Calendar calendar = Calendar.getInstance();
        String time = form.format(calendar.getTime());
        clock.setText(time);
    }
};
```

Thread 예제(Message)

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        switch (item.getItemId()) {
            case R.id.item1:
                type = 1;
                break;
            case R.id.item2:
                type = 2;
        }
        item.setChecked(true);
        onStart();
        return true;
    }
}
```

Thread 예제(Message)

- TimerThread1.JAVA

```
public class TimerThread1 extends Thread{
    private Handler handler;

    public TimerThread1(Handler handler) {
        this.handler = handler;
    }

    @Override
    public void run() {
        while (!Thread.currentThread().isInterrupted()) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
            handler.sendEmptyMessage(0);
        }
    }
}
```

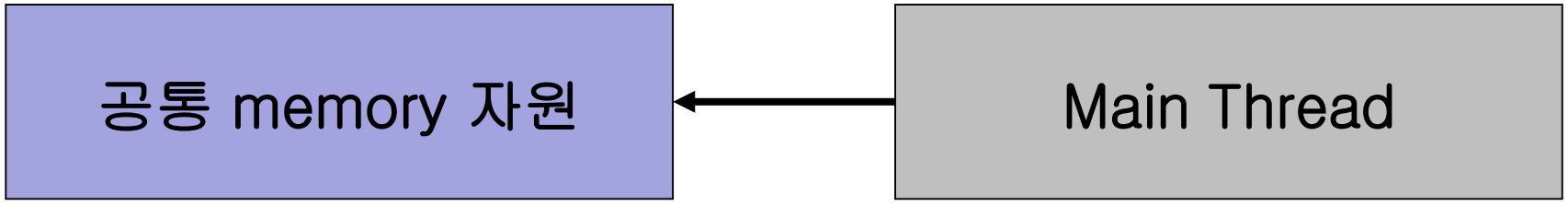
Thread 예제(Message)

- TimerRunnable1.JAVA

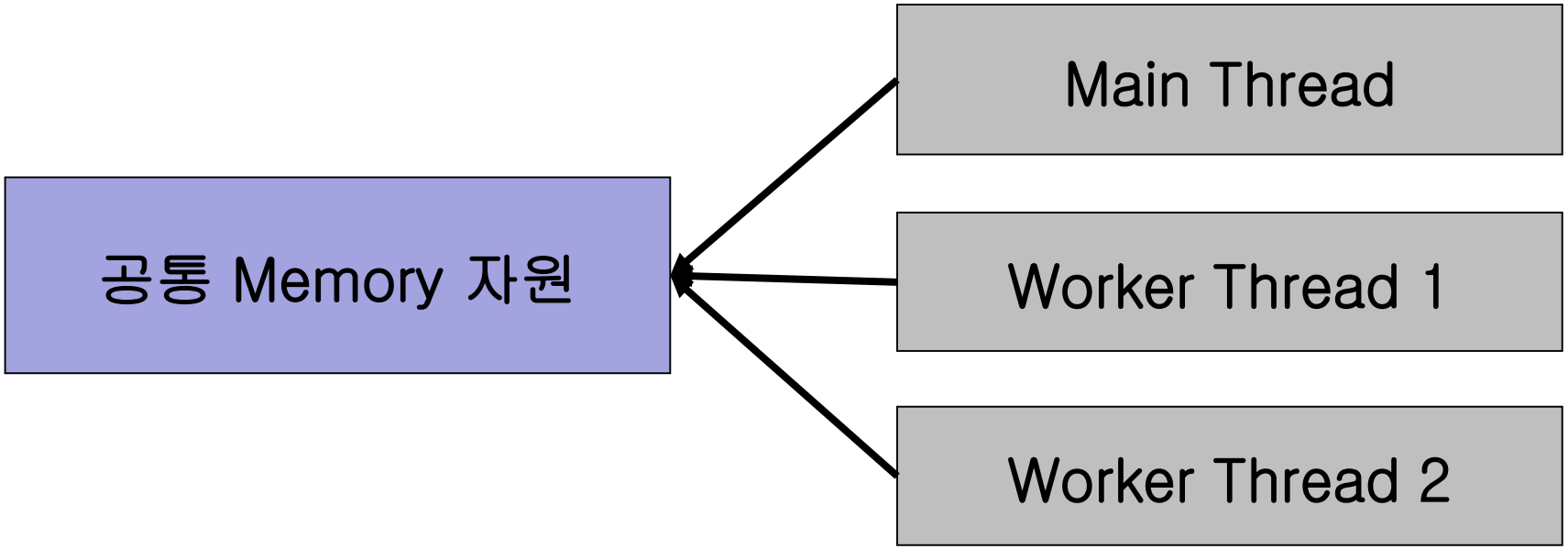
```
public class TimerRunnable1 implements Runnable {
    private Handler handler;

    public TimerRunnable1(Handler handler) {
        this.handler = handler;
    }

    @Override
    public void run() {
        while (!Thread.currentThread().isInterrupted()) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
            handler.sendEmptyMessage(1);
        }
    }
}
```

Thread 예제(Message)

- 공용 자원 활용 방법



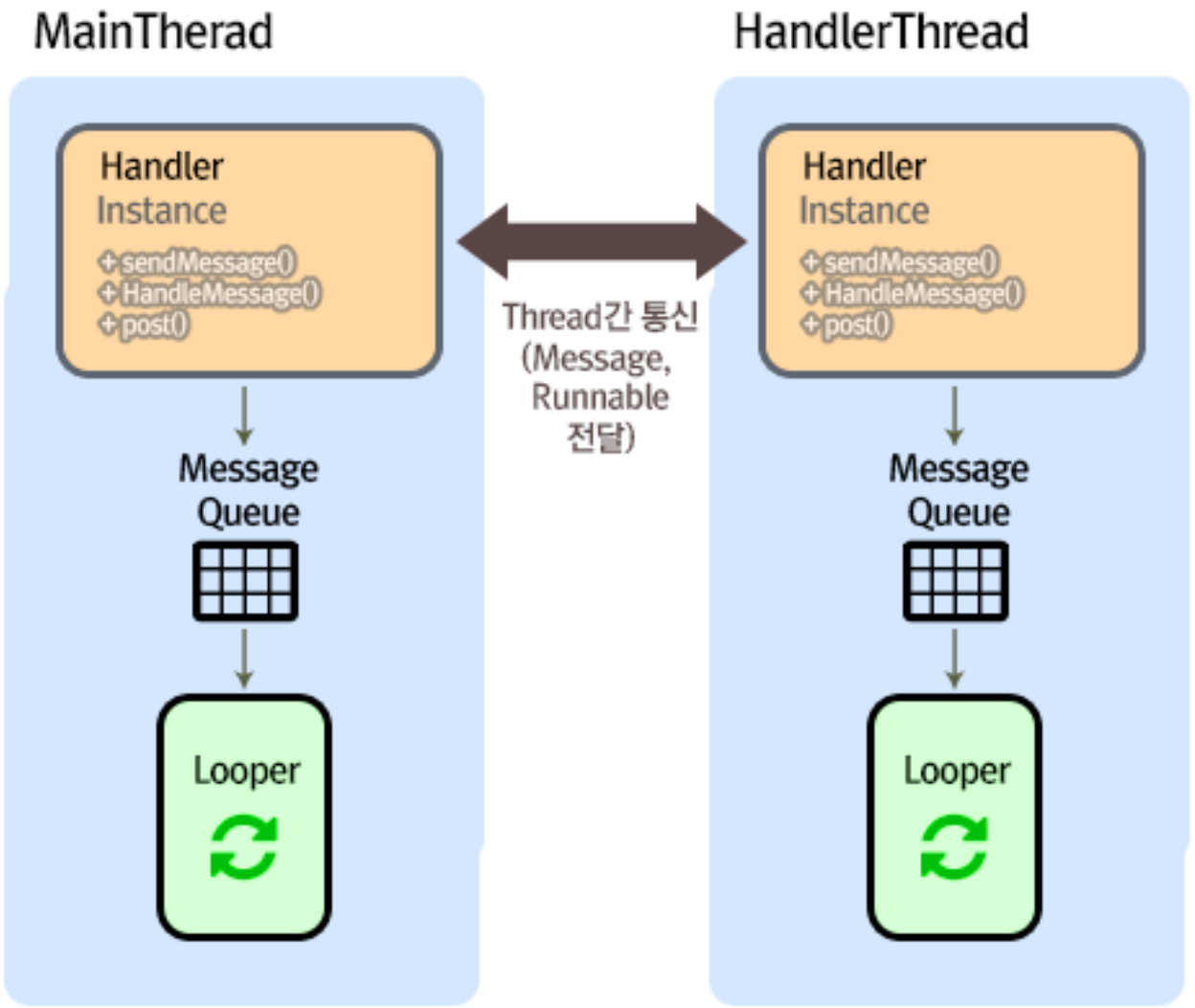
  - 일반적으로 Thread를 생성하지 않는 경우 Main Thread 한 개만 존재해서, Main Thread가 UI를 처리하게 됨



Thread 예제(Message)

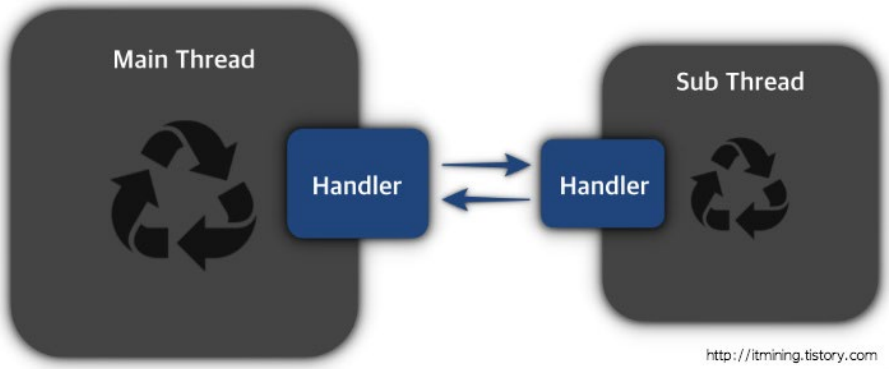
- Handler Thread
  - 정말 Main Thread가 아닌곳에서 화면에 그리는 작업을 할 수 없을까? 그렇다
    - 하지만 다른 Thread에서 Main Thread가 그림을 그리도록 요청할 수 있음
    - Main Thread는 구조적으로 그러한 것을 지원함
  - 그것을 Handler Thread 라고 부르고 Main Thread는 Handler Thread 구조임
  - Handler Thread는 내부적으로 Queue를 가짐
  - 그 Queue안에는 처리 해야 하는 일을 해당 Thread에서나 혹은 다른 Thread에서 넣을 수 있음
  - 그러므로 다른 Thread에서 그리는 작업을 시키기 위해 Handler Thread의 Queue에 Job를 넣을 수 있는 것임

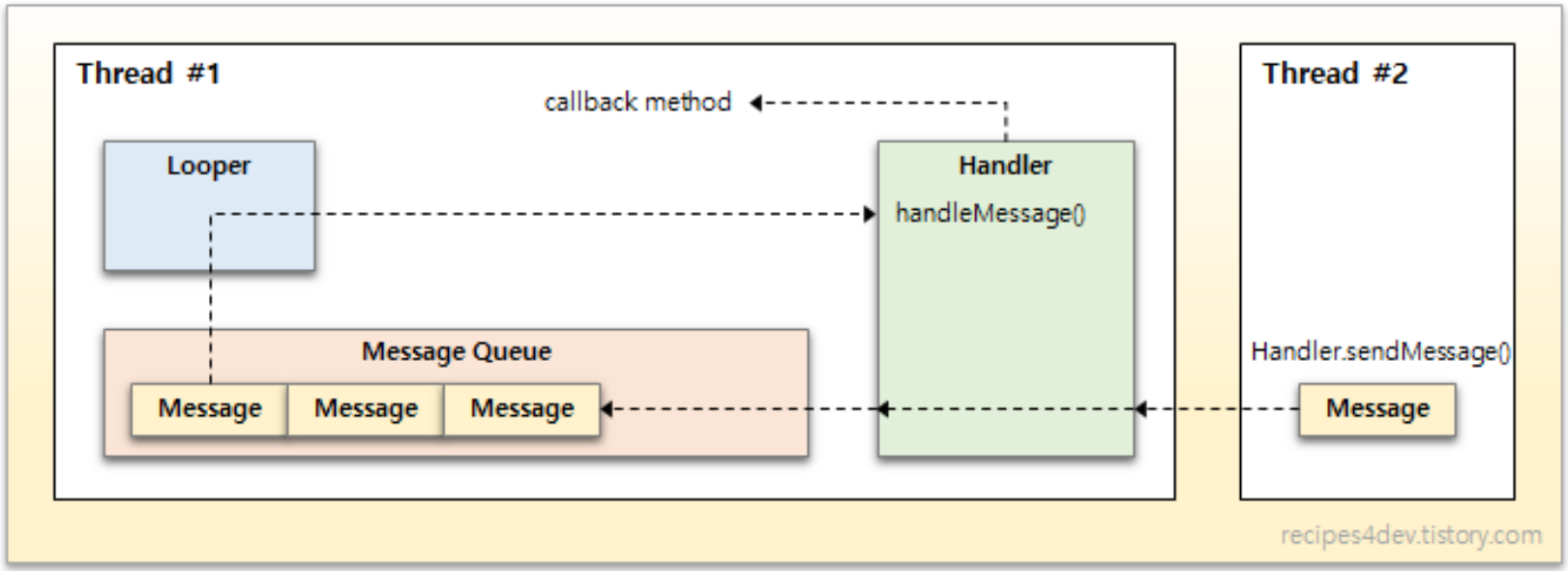
Thread 예제(Message)



Thread 예제(Message)

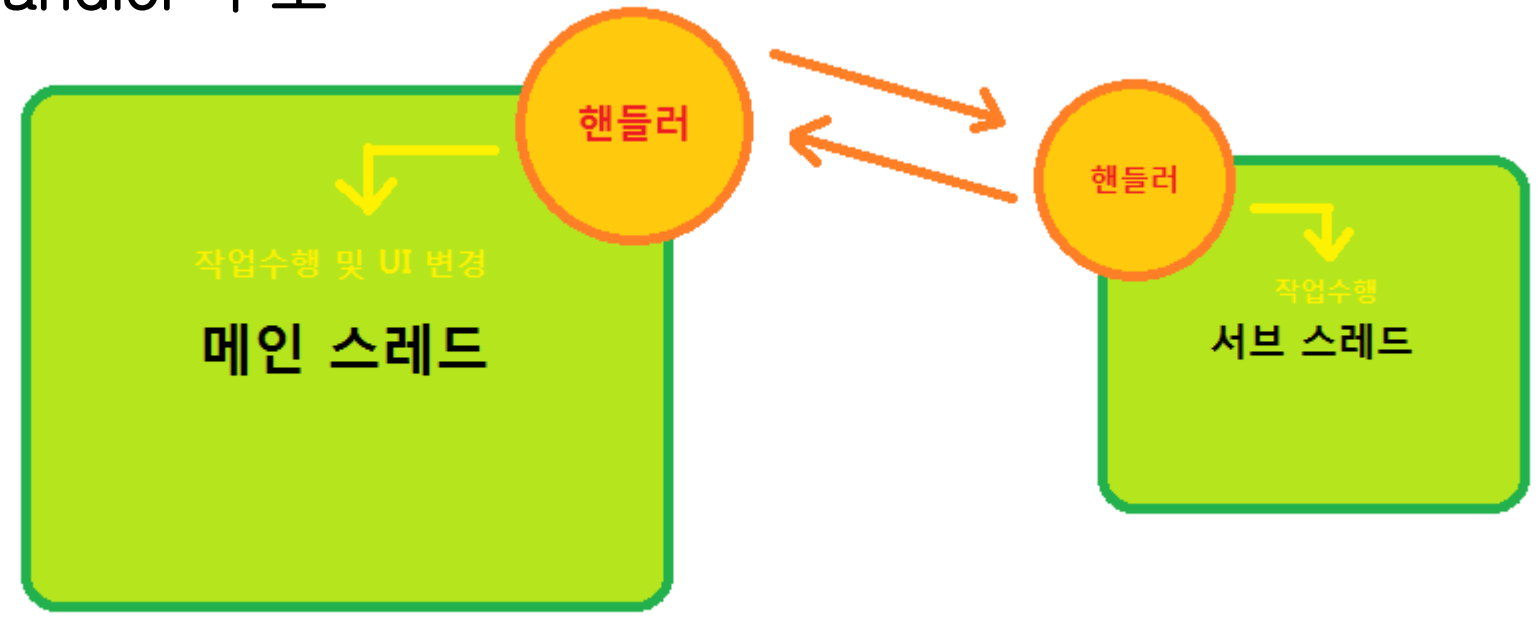
- Handler





Thread 예제(Message)

- Handler 구조



- Handler에 Message가 들어오면 순서대로 쌓여서 FIFO(First in First Out) 형태로 Message를 처리하게 됨
- 기본 생성자를 통해 Handler를 생성하면, 생성되는 Handler는 해당 Handler를 호출한 Thread의 MessageQueue와 Looper에 자동 연결

Thread 예제(Message)

- Message를 받는 쪽
  - void handleMessage(Message msg) 메소드
  - Message는 Thread간 통신 내용을 저장하는 Object이고 단순한 신호, 명령 뿐만 아니라 추가 정보도 전달받을 수 있음

Field	설명
int what	✓ Message의 의미를 설명 ✓ 의미가 정해져 있지는 않으며 Handler별로 지역적이므로 다른 Handler와 충돌할 위험은 없음
int arg1	✓ Message의 추가 정보
int arg2	✓ Message의 추가 정보
Object obj	✓ 정수만으로 Message를 기술할 수 없을 때 임의의 객체를 전달
Messenger replyTo	✓ Message에 대한 응답을 받을 객체를 지정

Thread 예제(Message)

- Message를 보내는 쪽
  - 전달하고자 하는 내용을 Message Object에 저장하여 Handler로 전송하는데, 이때 아래 Method를 사용
  - Message what(ID)를 사용할 경우

```
boolean Handler.sendEmptyMessage(int what)
```

  - Message what, arg1, obj 등 ID와 정보 등을 같이 사용하는 경우

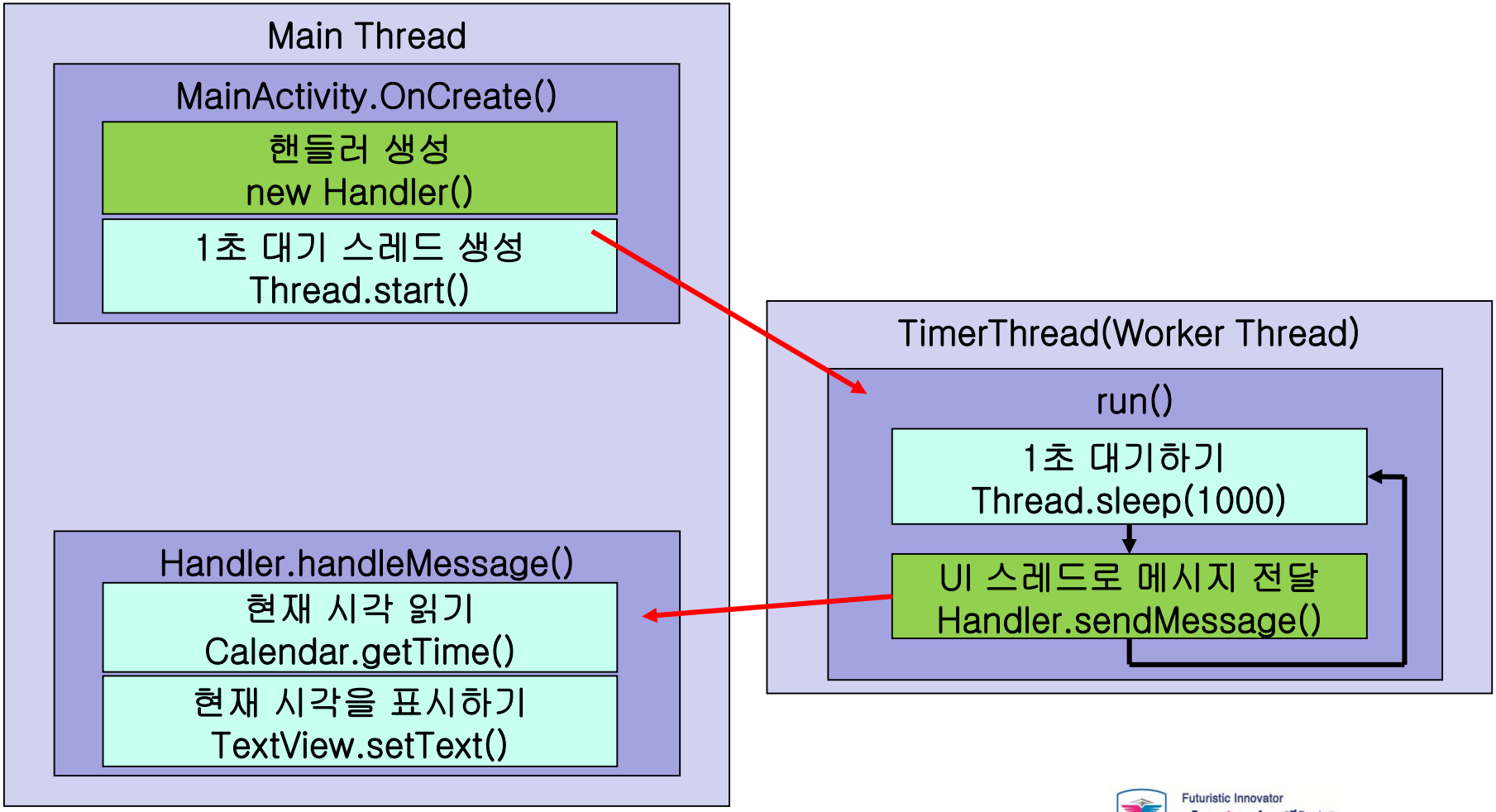
```
boolean Handler.sendMessage(Message msg)
```

  - Message는 Queue에 순서대로 쌓여 처리되나 급하게 처리해야 할 Message를 우선적으로 지정할 때 사용

```
boolean sendMessageAtFrontOfQueue(Message msg)
```

Thread 예제(Message)

- 정상적인 Thread Program 구조



Thread 예제(post())

- Main Thread와 Work Thread사이의 통신 방법으로 post() 메소드를 사용

Thread 예제(post())

- DigitalClockActivity4.JAVA

```
public class DigitalClockActivity4 extends AppCompatActivity {
    private int type = 1;
    private Thread thread;
    private TimerThread3 thread1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_digitalclock1);
    }
```

Thread 예제(post())

- DigitalClockActivity4.JAVA

```
@Override
protected void onStart() {
    super.onStart();
    TextView clock = findViewById(R.id.watch);
    if (type == 1) {
        if (thread != null && thread.isAlive())
            thread.interrupt();
        thread1 = new TimerThread3(clock);
        thread1.start();
    } else {
        if (thread1 != null && thread1.isAlive())
            thread1.interrupt();
        TimerRunnable3 runnable = new TimerRunnable3(clock);
        thread = new Thread(runnable);
        thread.start();
    }
}
```

Thread 예제(post())

- DigitalClockActivity4.JAVA

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu, menu);
    return true;
}
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    switch (item.getItemId()) {
        case R.id.item1:
            type = 1;
            break;
        case R.id.item2:
            type = 2;
    }
    item.setChecked(true);
    onStart();
    return true;
}
```

Thread 예제(post())

- TimerThread3.JAVA

```
public class TimerThread3 extends Thread{
    private TextView textView;
    private Handler handler = new Handler();
    private SimpleDateFormat form;

    public TimerThread3(TextView textView) {
        this.textView = textView;
        form = new SimpleDateFormat("a HH:mm:ss");
        textView.setTextColor(Color.BLACK);
    }
```

Thread 예제(post())

- TimerThread3.JAVA

```
    @Override
    public void run() {
        while (!Thread.currentThread().isInterrupted()) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
            handler.post(new Runnable() {
                @Override
                public void run() {
                    Calendar calendar = Calendar.getInstance();
                    String time = form.format(calendar.getTime());
                    textView.setText(time);
                }
            });
        }
    }
}
```

Thread 예제(post())

- TimerRunnable3.JAVA

```
public class TimerRunnable3 implements Runnable {
    private TextView textView;
    private Handler handler = new Handler();
    private SimpleDateFormat form;

    public TimerRunnable3(TextView textView) {
        this.textView = textView;
        form = new SimpleDateFormat("a HH:mm:ss", Locale.KOREA);
        textView.setTextColor(Color.BLUE);
    }
```

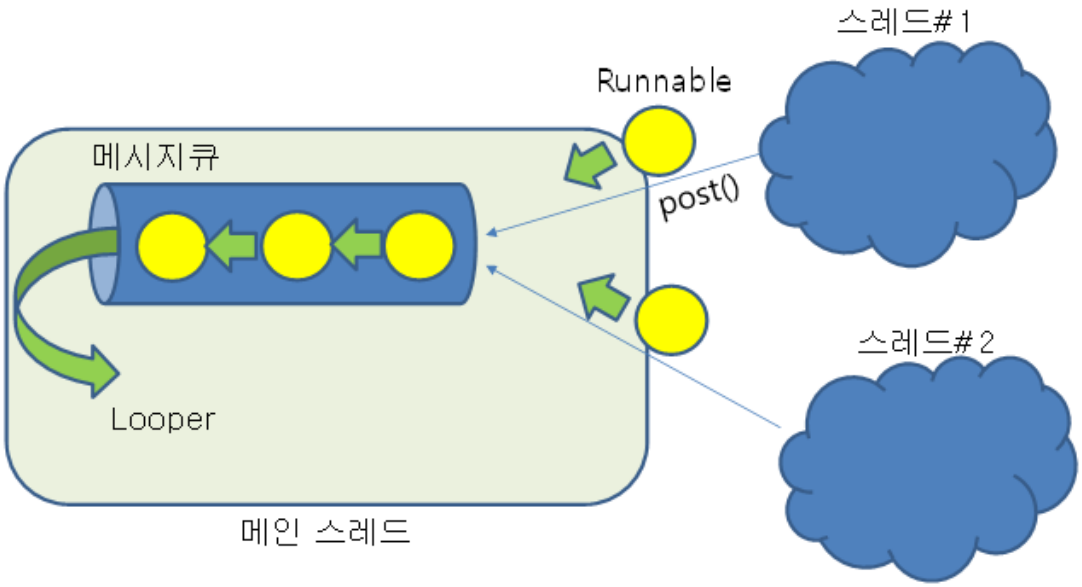
Thread 예제(post())

- TimerRunnable3.JAVA

```
@Override
public void run() {
    while (!Thread.currentThread().isInterrupted()) {
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
        handler.post(new Runnable() {
            @Override
            public void run() {
                Calendar calendar = Calendar.getInstance();
                String time = form.format(calendar.getTime());
                textView.setText(time);
            }
        });
    }
}
}
```

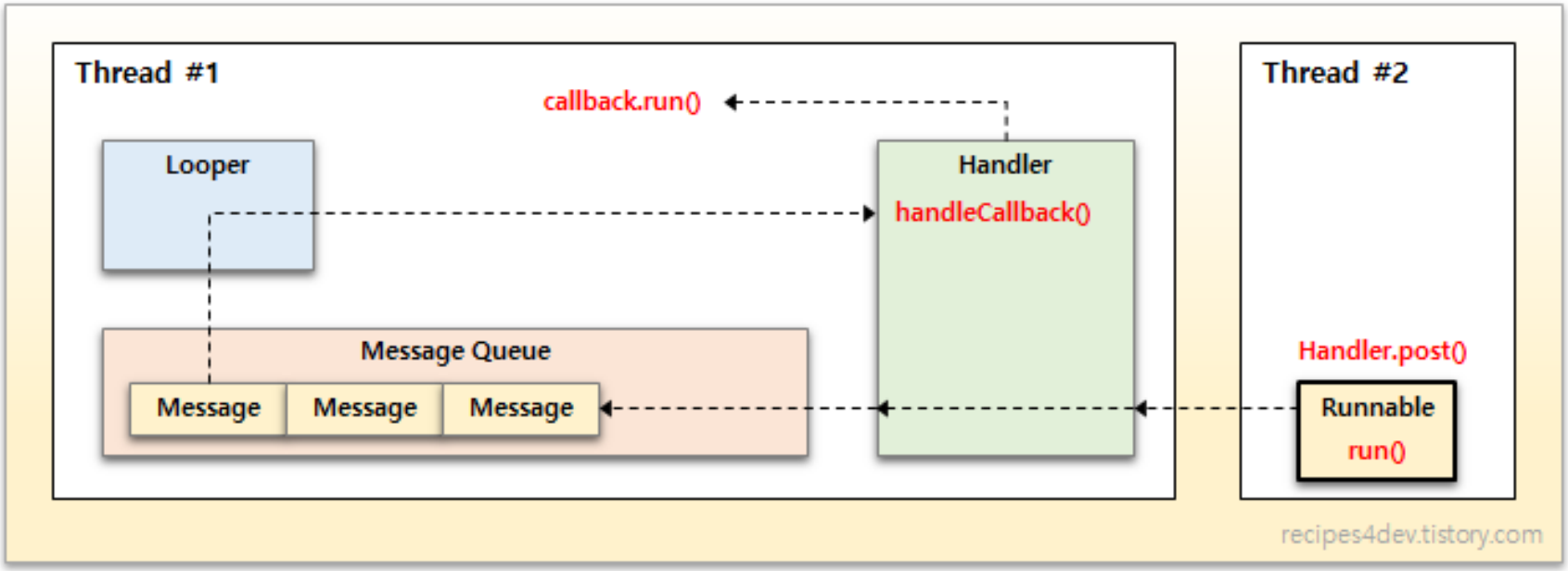
Thread 예제(post())

- post() 메소드
  - post() 메소드는 간단히 Handler에서 처리 해야 될 때 사용 할 수 있음
  - 이것은 new Runnable()의 run()을 이용
  - 그러면 sendMessage() 메소드를 쓰지 않아도 run() 메소드 내에서 직관적으로 처리 할 수 있음



Thread 예제(post())

- Runnable 객체를 만들고 run() 메소드를 Override하고 나면, 마지막으로 할 일은 Handler.post() 메소드를 사용하여, 앞서 생성한 Runnable Object를 수신 측 Thread로 보내는 것



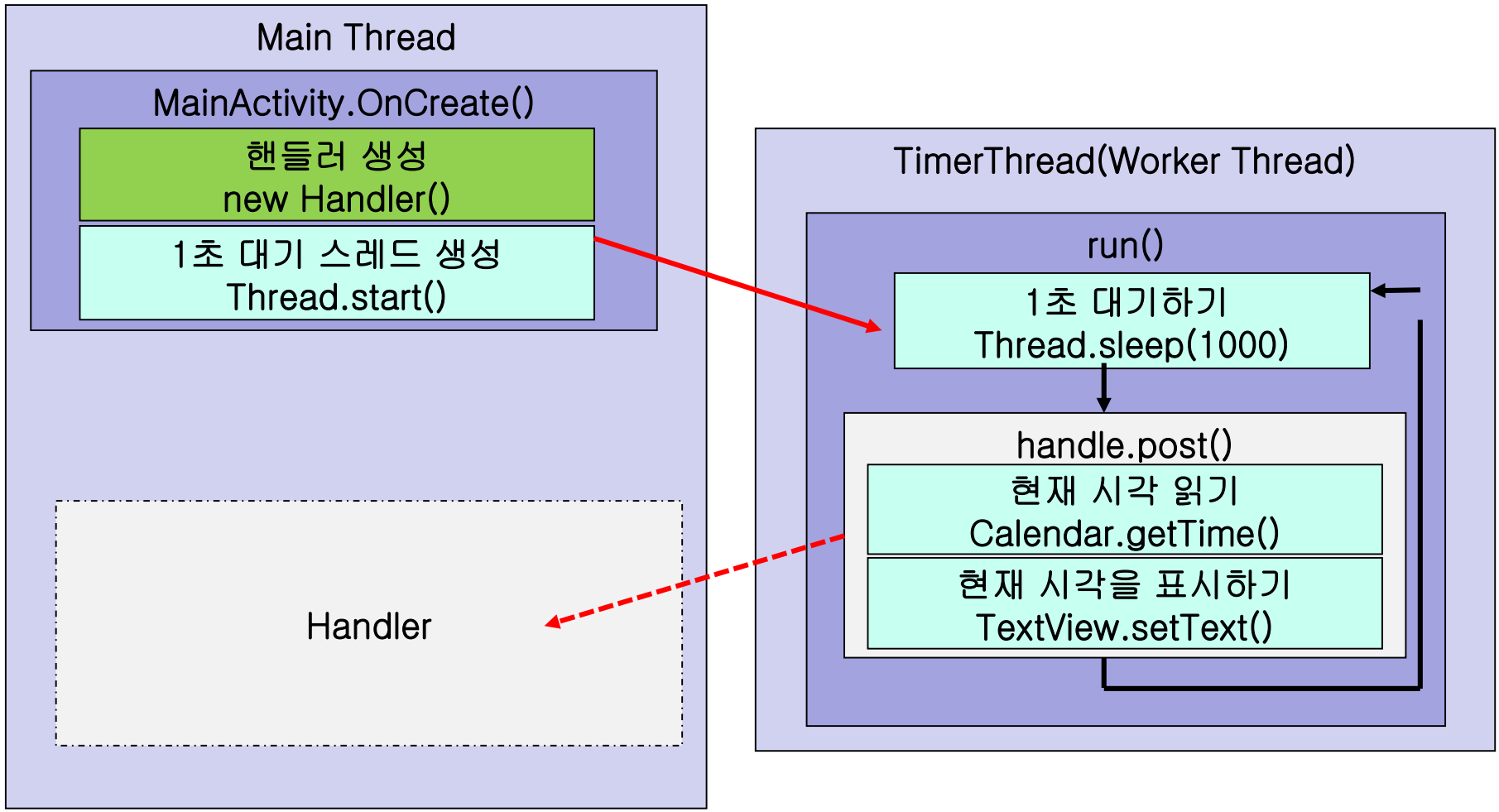
Thread 예제(post())

- Runnable Object를 보낼 때 사용하는 post() 메소드

메소드	설명
boolean post(Runnable r)	Runnable 객체를 전달 (핸들러에 연결된 메시지 큐에 추가)
boolean postAtFrontOfQueue (Runnable r)	Runnable 객체를 메시지 큐의 가장 앞에 추가
boolean postAtTime (Runnable r, long uptimeMillis)	uptimeMillis로 지정된 시각에, Runnable 객체 전달
boolean postAtTime (Runnable r, Object token, long uptimeMillis)	uptimeMillis로 지정된 시각에, Runnable 객체 전달. r을 취소하는데 사용될 수 있는 token 인스턴스 사용 가능
boolean postDelayed (Runnable r, long delayMillis)	현재 시각에서 delayMillis 만큼의 시간 후에, Runnable 객체 실행
boolean postDelayed (Runnable r, Object token, long delayMillis)	현재 시각에서 delayMillis 만큼의 시간 후에, Runnable 객체 실행. token 인스턴스를 통해 r의 실행 취소 가능

Thread 예제(post())

- 정상적인 Thread Program 구조



Thread 예제(runOnUiThread())

- Main Thread와 Work Thread사이의 통신 방법으로 runOnUiThread() 메소드를 사용

Thread 예제(runOnUiThread())

- DigitalClockActivity5.JAVA

```
public class DigitalClockActivity5 extends AppCompatActivity {
    private int type = 1;
    private Thread thread;
    private TimerThread thread1;
    private TextView clock;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_digitalclock1);
    }
```

Thread 예제(runOnUiThread())

- DigitalClockActivity5.JAVA

```
@Override
protected void onStart() {
    super.onStart();
    clock = findViewById(R.id.watch);

    if (type == 1) {
        if (thread != null && thread.isAlive())
            thread.interrupt();
        thread1 = new TimerThread();
        thread1.start();
    } else {
        if (thread1 != null && thread1.isAlive())
            thread1.interrupt();
        TimerRunnable runnable = new TimerRunnable();
        thread = new Thread(runnable);
        thread.start();
    }
}
```

Thread 예제(runOnUiThread())

```
class TimerRunnable implements Runnable {
    SimpleDateFormat form = new SimpleDateFormat("a HH:mm:ss", Locale.KOREA);
    @Override
    public void run() {
        while (!Thread.currentThread().isInterrupted()) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    Calendar calendar = Calendar.getInstance();
                    String time = form.format(calendar.getTime());
                    clock.setTextColor(Color.BLUE);
                    clock.setText(time);
                }
            });
        }
    }
}
```

Thread 예제(runOnUiThread())

```
class TimerThread extends Thread{
    SimpleDateFormat form = new SimpleDateFormat("a HH:mm:ss");
    @Override
    public void run() {
        while (!Thread.currentThread().isInterrupted()) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    Calendar calendar = Calendar.getInstance();
                    String time = form.format(calendar.getTime());
                    clock.setTextColor(Color.BLACK);
                    clock.setText(time);
                }
            });
        }
    }
}
```

Thread 예제(runOnUiThread())

- DigitalClockActivity5.JAVA

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu, menu);
    return true;
}
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    switch (item.getItemId()) {
        case R.id.item1:
            type = 1;
            break;
        case R.id.item2:
            type = 2;
    }
    item.setChecked(true);
    onStart();
    return true;
}
```

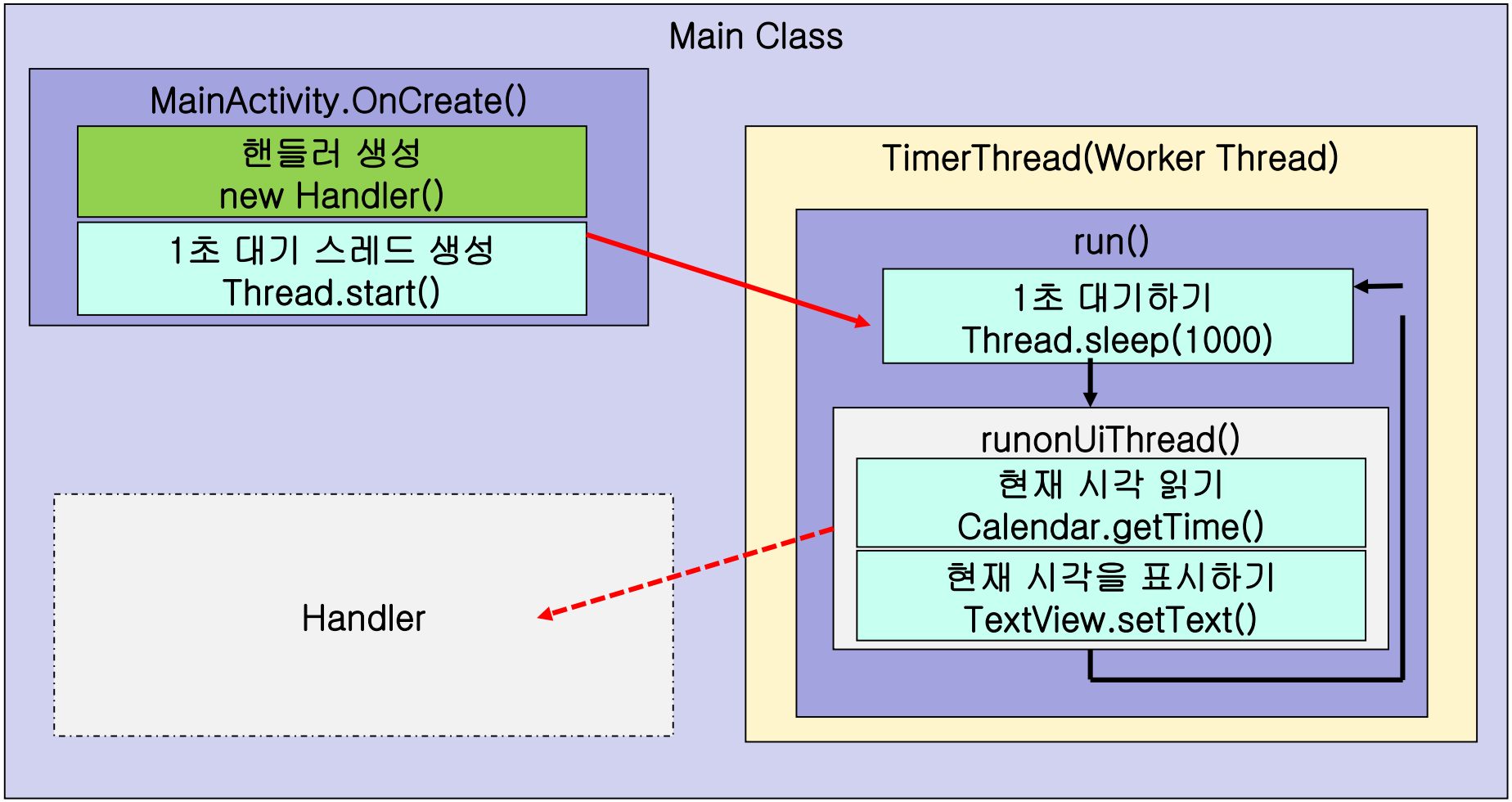
Thread 예제(runOnUiThread())

- UI Thread에서 지정된 작업을 실행
  - 현재 Thread가 UI Thread이면 작업이 즉시 실행
  - 현재 Thread가 UI Thread가 아닌 경우 조치는 UI Thread의 Event Queue에 게시
  - 지금 작업을 수행하는 Thread가 Main Thread라면 즉시 작업을 시작하고 Main Thread가 아니라면 Thread Event Queue에 쌓아두는 기능을 하는게 runOnUiThread()

- runOnUiThread()는 Activity Class에서 제공되는 Method
- 개발자가 만든 Runnable Object를 Main Thread에서 실행되게 하는 Method
- 현재 Method가 Main Thread인지 여부를 검사해 Main Thread가 아니라면 post()를 실행하고 맞으면 Runnable의 run()을 실행

Thread 예제(runOnUiThread())

- 정상적인 Thread Program 구조



Thread 예제(Main Thread만으로)

Thread 예제(Main Thread만으로)

- DigitalClockActivity6.JAVA

```
public class DigitalClockActivity6 extends AppCompatActivity {
    private TextView clock;
    private SimpleDateFormat form;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_digitalclock1);

        clock = findViewById(R.id.watch);
        form = new SimpleDateFormat("a HH:mm:ss", Locale.KOREA);
        handler.sendEmptyMessage(0);
    }
```

Thread 예제(Main Thread만으로)

- DigitalClockActivity6.JAVA

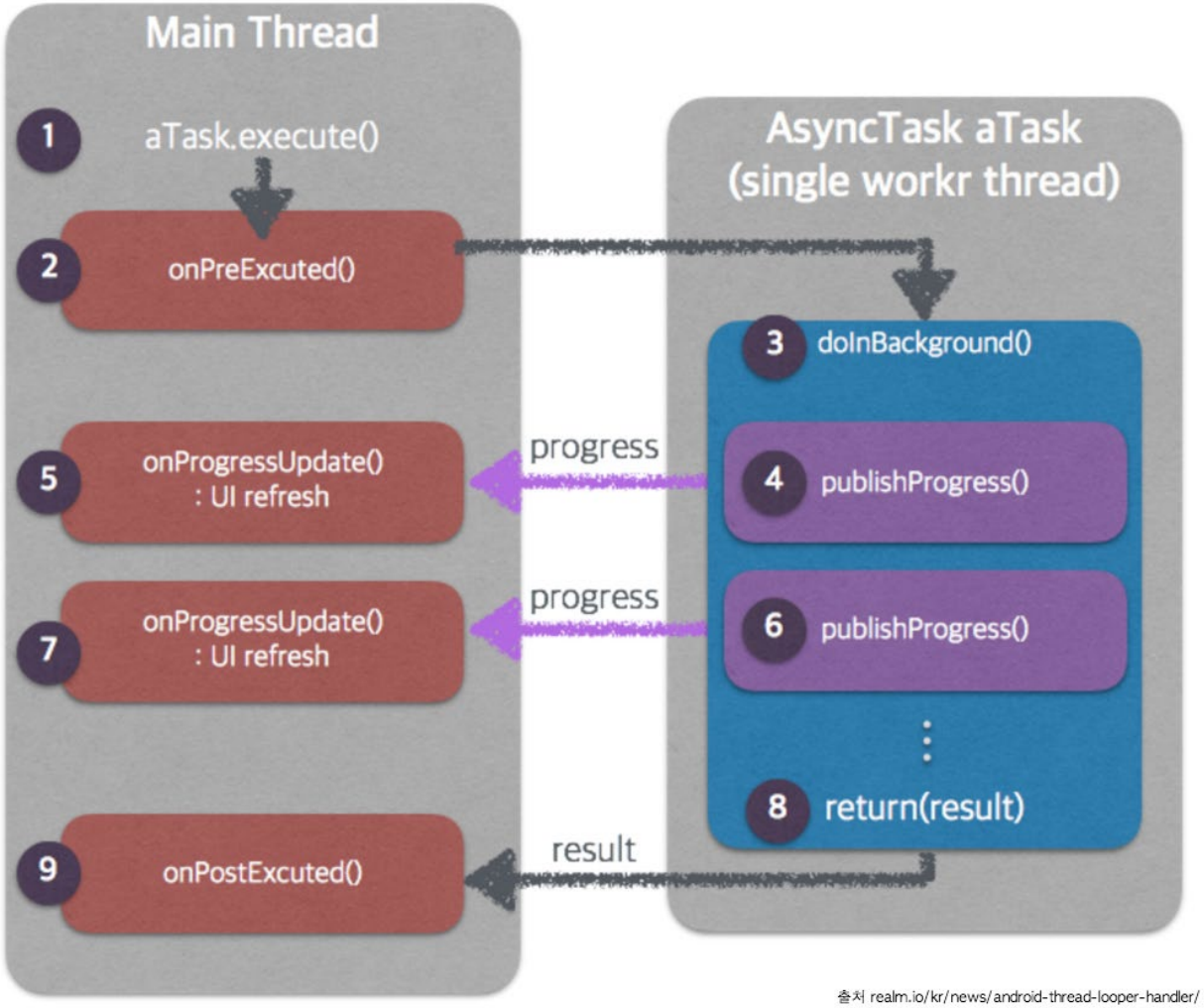
```
    Handler handler = new Handler(Looper.getMainLooper()) {
        @Override
        public void handleMessage(Message msg) {
            if (msg.what == 0) {
                Calendar calendar = Calendar.getInstance();
                String time = form.format(calendar.getTime());
                clock.setText(time);
                handler.sendEmptyMessageDelayed(0, 1000);
            }
        }
    };
}
```

AsyncTask 클래스

- Background에서 비동기 작업을 실행하기 위한 Class를 사용하기 위해서는 Sub Class를 구현
- 몇 초 정도 걸리는 짧은 작업에 사용하도록 권장
- Single Thread(3.0 이후)에서 동작, Multi Threading을 원하면 executeOrExcutor() UI Thread에서 생성, 로드, 실행
- cancel() 메소드를 사용해 언제든지 취소 가능
- 한번 생성해서 실행하고 나면, 다시 실행할 순 없음

AsyncTask 클래스

- AsyncTask의 개념



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

AsyncTask 클래스

- AsyncTask 사용 규칙
  - AsyncTask는 일회용 클래스. 2번 이상 사용하면 안 됨
  - 두 번 째 execute() 호출 시 Error 발생
  - AsyncTask 객체는 Main Thread에서 생성되어야 하고 실행되어야 함
  - AsyncTask의 콜백 메소드인 onPreExecute(), doInBackground(), onProgress() 등을 수동으로 호출 하면 Error 발생

- Google은 AsyncTask를 사용할 때 '수 초 내의 동작에만 사용'하는 것을 권장 하고 있음
- 그 이상의 작업을 하고 싶을 때는 Thread를 직접 구현을 하는 것을 권장하고 있음
  - 그 이유는 AsyncTask가 Activity에 종속되지 않기 때문임

AsyncTask 클래스

- AsyncTask의 동작 순서
  - execute( ) 명령어를 통해 AsyncTask을 실행
  - AsyncTask로 Background 작업을 실행하기 전에 onPreExcuted( ) 메소드가 실행
    - 이 부분에는 Image Loading 작업이라면 Loading 중 Image를 띄워 놓기 등, Thread 작업 이전에 수행할 동작을 구현
  - 새로 만든 Thread에서 Background 작업을 수행
    - execute( ) 메소드를 호출할 때 사용된 매개변수를 전달 받음
  - doInBackground( ) 메소드에서 중간 중간 진행 상태를 UI에 Update하도록 하려면 publishProgress( ) 메소드를 호출
  - onProgressUpdate( ) 메소드는 publishProgress( ) 메소드가 호출 될 때 마다 자동으로 호출

AsyncTask 클래스

- AsyncTask의 동작 순서
  - doInBackground( ) 메소드에서 작업이 끝나면 onPostExcuted( ) 메소드로 결과 매개 변수를 반환하면서 그 반환 값을 통해 Thread 작업이 끝났을 때의 동작을 구현

Thread 예제(AsyncTask)

```
public class DigitalClockActivity7 extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_digitalclock1);

        TextView clock = findViewById(R.id.watch);
        TimerAsyncTask task = new TimerAsyncTask(clock);
        task.execute();
    }
}
```

Thread 예제(AsyncTask)

- TimerAsyncTask.JAVA

```
public class TimerAsyncTask extends AsyncTask<Void, String, Void> {
    private TextView textView;
    private SimpleDateFormat form;

    public TimerAsyncTask(TextView textView) {
        this.textView = textView;
        form = new SimpleDateFormat("a HH:mm:ss");
    }

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    @Override
    protected void onPostExecute(String time) {
        textView.setText(time);
    }
```

Thread 예제(AsyncTask)

- TimerAsyncTask.JAVA

```
    @Override
    protected Void doInBackground(Void... Voids) {
        while (true) {
            try {
                Thread.sleep(1000);
            } catch (Exception e) {
                e.printStackTrace();
            }
            Calendar calendar = Calendar.getInstance();
            String time = form.format(calendar.getTime());
            publishProgress(time);
        }
    }

    @Override
    protected void onProgressUpdate(String... Voids) {
        textView.setText(Voids[0]);
     }
}
```

Thread 예제(AsyncTask)

- AsyncTask (삭제됨)
  - UI Thread에서 생성하고, 사용해야 한다는 제약사항 있음
  - 비교적 오래 걸리지 않는 작업에 유용
  - Callback 메소드들이 잘 정의되어 있음
  - Task cancel이 용이한 편임
  - 재사용이 불가능

- Handler + Thread
  - UI Thread 사용 제약이 없음 (Looper 전달 가능)
  - 작업 시간에 대한 제약이 없음
  - 구현이 AsyncTask에 비해 조금 복잡할 수 있음
  - Task cancel이 어려움
  - 재사용 가능

Thread 예제(AsyncTask)

- 언제 어떤걸 사용할까?
  - 주로 Async하게 처리해야 하는 경우는 I/O 처리를 할 때로, DB, Network, Bitmap 처리, File 처리 등
  - DB, Network은 Handler + Thread로 처리하는 것이 좋고, Bitmap은 AsyncTask, 그리고 File은 용량이나 처리 방법에 따라 조절해서 사용