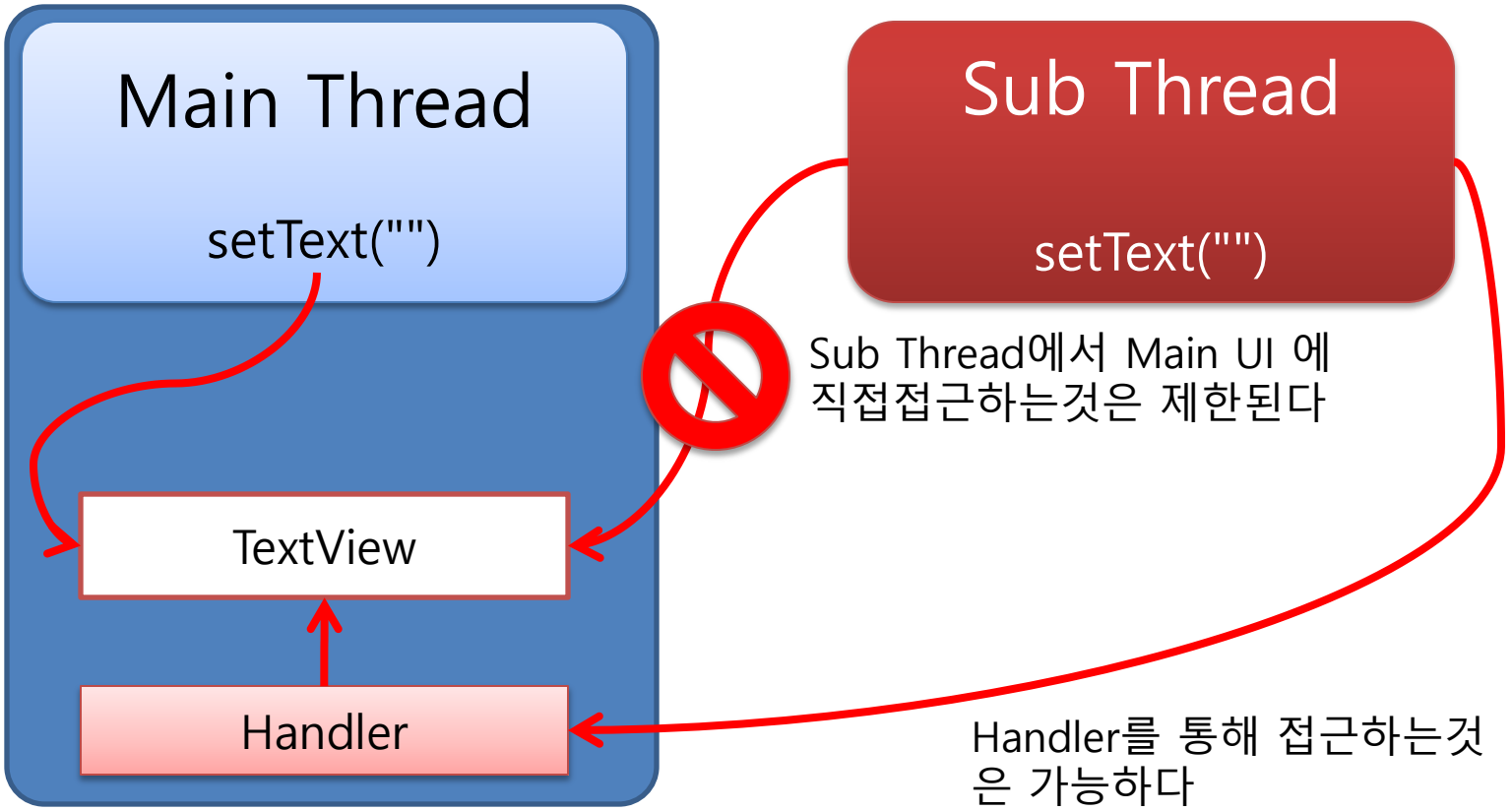
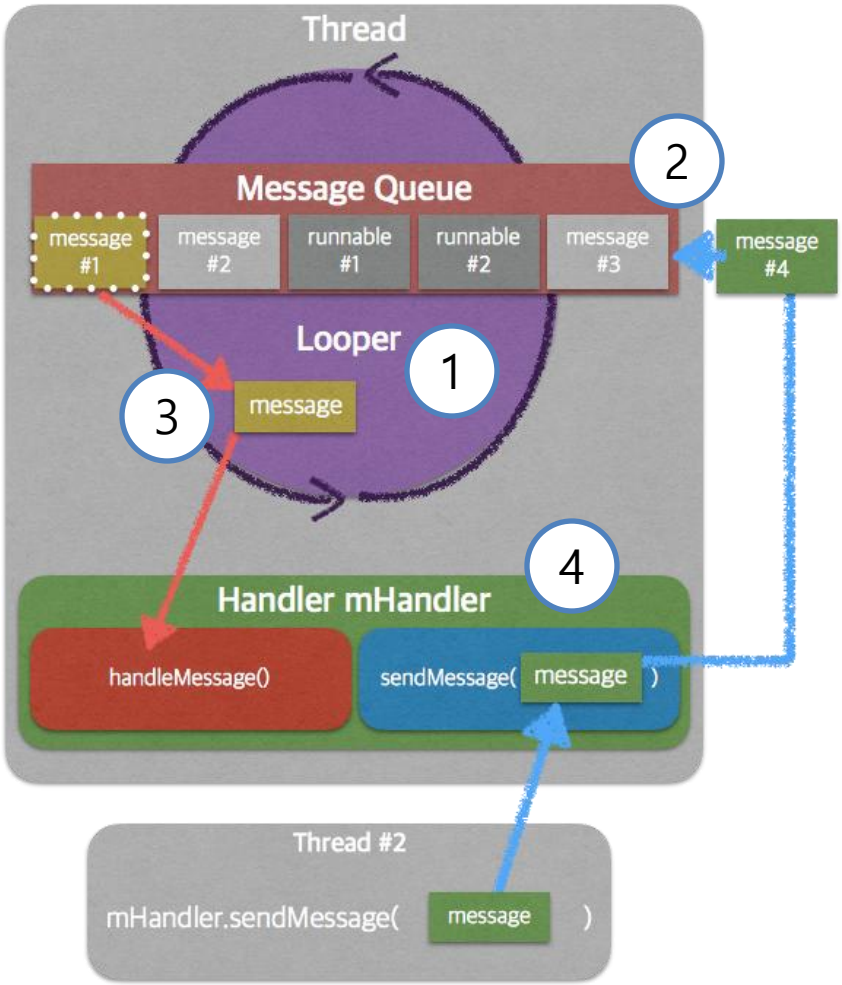
Android
023 Thread

1. Looper & Handler

1. 안드로이드에서는 두 개 이상의 스레드를 사용할 때의 동기화 이슈를 차단하기 위해서 Looper와 Handler를 사용한다.



2. Looper & Handler Architecture



① 안드로이드에서는 Main Thread 에는 기본적으로 Looper 라고 하는 메시지 전달객체가 동작하고 있다

② Looper는 Message Queue를 포함하고 있는데 Thread가 자기자신 혹은 다른 Thread로 부터 전달받은 Message를 FIFO 형태로 보관한다

③ Looper는 큐에서 Runnable 객체나 메시지를 순서대로 꺼내서 Handler가 처리할 수 있게 전달한다

④ Handler는 Looper로 부터 받은 메시지를 실행, 처리하거나 다른 Thread로 부터 받아서 Queue에 넣는 Thread간의 메신저 역할을 한다

3. Thread & Runnable

1. Thread 의 사용 - Thread를 상속받아서 사용한다

```
class CustomThread extends Thread {
    public void run( ){
        System.out.println("Running Thread!!");
    }
}
Thread thread = new CustomThread( );
thread.start( );
```

2. Runnable 인터페이스를 통해 implements 후 사용할 수 있다

```
class CustomRunnable implements Runnable {
    public void run( ){
        System.out.println("Running Thread!!");
    }
}
Thread thread = new Thread( new CustomRunnable( ));
thread.start( );
```


4. Thread 에서의 Looper구현

1. Thread 의 내부에서 Looper를 구현하여 메시지를 핸들링 할 수 있다

```
class CustomThread extends Thread {
    Handler threadHandler;
    public void run( ){
        Looper.prepare( );
        threadHandler = new Handler( );
        Looper.loop( ); // 루퍼가 시작되면서 Thread가 대기상태가 된다
    }
    public void callFunction( ){
        System.out.print("called wait Thread!!!");
    }
}
Thread thread = new CustomThread( );
thread.start( );

thread.callFunction( ); // run의 Looper로 인해 대기상태의 Thread를 호출할 수 있다
```


5. HandleThread

1. 안드로이드는 Looper가 구현되어 있는 HandlerThread를 사용할 수 있다

```
class CustomThread extends HandlerThread {
    Handler threadHandler;
    public void onLooperPrepared( ){ // run 대신에 사용된다
        super.onLooperPrepared( );

    }
    public void callFunction( ){
        System.out.print("called wait Thread!!!");
    }
}
Thread thread = new CustomThread( );
thread.start( );

thread.callFunction( ); // HandlerThread를 상속받으면 Looper를 내부적으로 구현해 주기 때문에 대기상태의 Thread를 호출할 수 있다
```

