
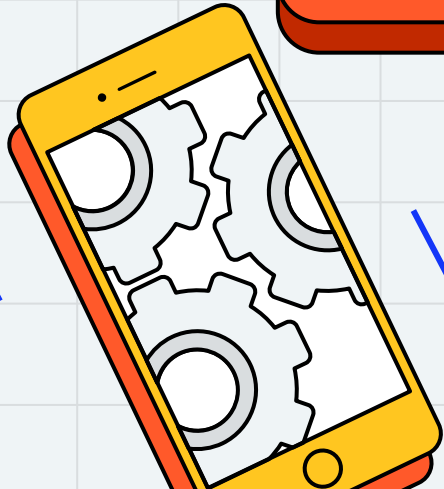




# Application

# Programming



Hend Alkittawi





# Multi-threaded Apps

Building applications the utilize  
threads

# INTRODUCTION

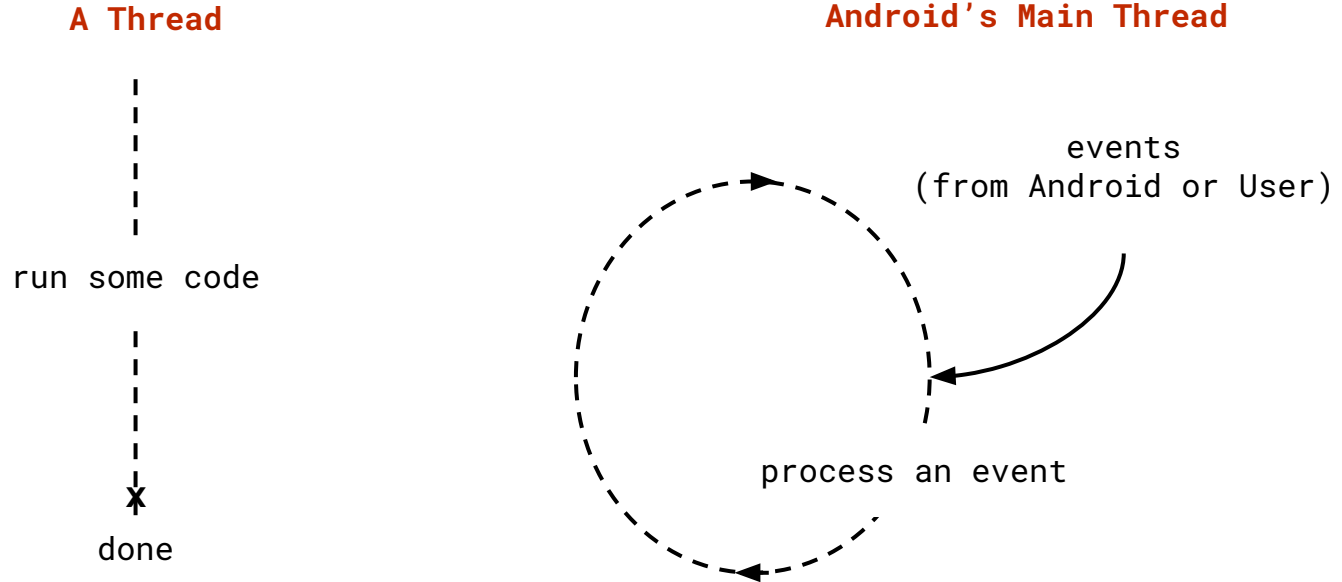
- Recall that a thread is a single sequence of execution, and that code running within a single thread will execute one step after another.
- Every Android app starts life with a main thread. The main thread, however, is not a pre-ordained list of steps. Instead it sits in an infinite loop and waits for events initiated by the user or the system. Then it execute code and response to those events as they occur.
- So far, all of our Android apps has been executed on the main thread.

# INTRODUCTION

- One of the key rules of Android development is to never perform time-consuming operations on the main thread of an application.
- The second rule is that the code within a separate thread must never, under any circumstances, directly update any aspect of the user interface.
  - Any changes to the user interface must always be performed from within the main thread.
  - The Android UI toolkit is not thread-safe. Attempts to work with non-thread-safe code from within multiple threads will typically result in intermittent problems and unpredictable application behavior!
- If the code executing in a thread needs to interact with the user interface, it must do so by synchronizing with the main UI thread. This is achieved by creating a handler within the main thread, which, in turn, receives messages from another thread and updates the user interface accordingly.

# ANDROID'S MAIN THREAD

- Regular threads vs the main thread



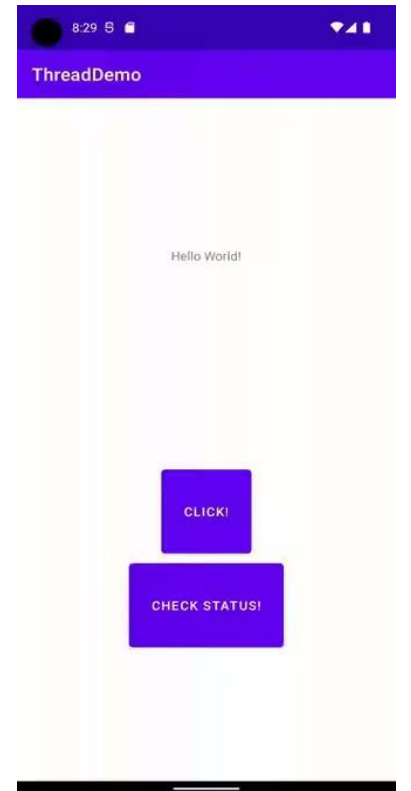
• reproduced from book figure

# ANDROID'S MAIN THREAD

- The main thread in an Android application runs all the code that updates the UI, including the code executed and response to the different UI related events (activity startup, button presses, ...).
  - because the events are all related to the UI and some way, the main thread is sometimes called the UI thread.
- The event loop keeps the UI code in sequence. It makes sure that none of these operations step on each other while still ensuring that the code is executed in a timely fashion.

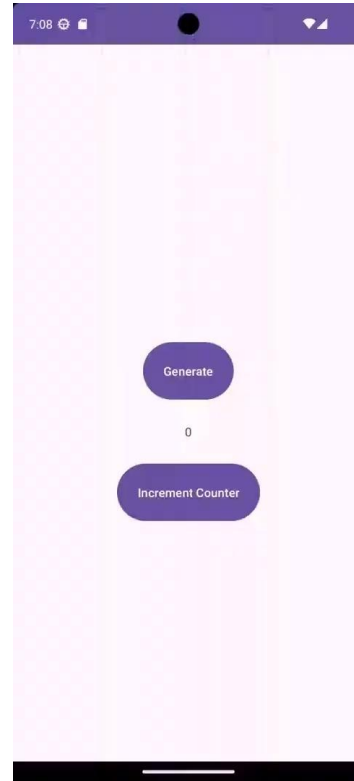
# BACKGROUND THREADS

- Sometimes a task takes long time compared to other tasks. During that time, the UI will be completely unresponsive, which might result in an Application Not Responding or ANR.
- An ANR occurs when Android's watchdog determines that the main thread has failed to respond to an important event, like pressing the back button.
- This is not the desired behavior for any application.



# WORD GENERATOR APP

- Let us look at an app that utilizes threads





```

public class WordGenerator {
    public final int wordLength;

    public WordGenerator(int wordLength){
        this.wordLength = wordLength;
    }

    public String generateWord(){
        String generatedWord = "";
        Random rand = new Random();
        for(int i = 0; i < wordLength; i++){
            char randChar = (char) rand.nextInt(256);
            generatedWord = generatedWord + randChar;
        }
        return generatedWord;
    }
}

```

**runOnUiThread(runnable)** runs the specified action on the UI thread. If the current thread is the UI thread, then the action is executed immediately. If the current thread is not the UI thread, the action is posted to the event queue of the UI thread.

```

public class MainActivity extends AppCompatActivity {
    private TextView generatorText;
    private TextView counterText;
    private Button generatorButton;
    private Button counterButton;
    private WordGenerator generator;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        generatorButton = findViewById(R.id.generator_button);
        counterButton = findViewById(R.id.counter_button);
        generatorText = findViewById(R.id.generator_text);
        counterText = findViewById(R.id.counter_text);

        generatorButton.setOnClickListener((view) -> startCounter());

        counterButton.setOnClickListener((view) -> {
            int count = Integer.parseInt(counterText.getText().toString());
            counterText.setText(String.valueOf(count + 1));
        });
    }

    public void startCounter() {
        Thread thread = new Thread(() -> {
            generator = new WordGenerator(5);
            while (true) {
                updateCounterText(generator.generateWord(), generatorText);
                try {
                    Thread.sleep(500); // Update every half second
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });
        thread.start();
    }

    public void updateCounterText(String text, TextView textView) {
        runOnUiThread(() -> textView.setText(text));
    }
}

```



**THANK**

**YOU!**



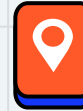
## DO YOU HAVE ANY QUESTIONS?



hend.alkittawi@utsa.edu



By Appointment



Online