BACKGROUND TASK IN ANDROID

TOPICS

- Review Room database
- Intro android Threads
  - Main Thread
  - Running background task
    - Thread
    - Async Task
- Examples

Introduction

- Threads
  - :independent sequences of execution
  - Pros.
    - Concurrency
    - Performance
    - Good user experience
  - Cons.
    - Data sharing
    - Correctness
    - Deadlock

ANDROID THREAD

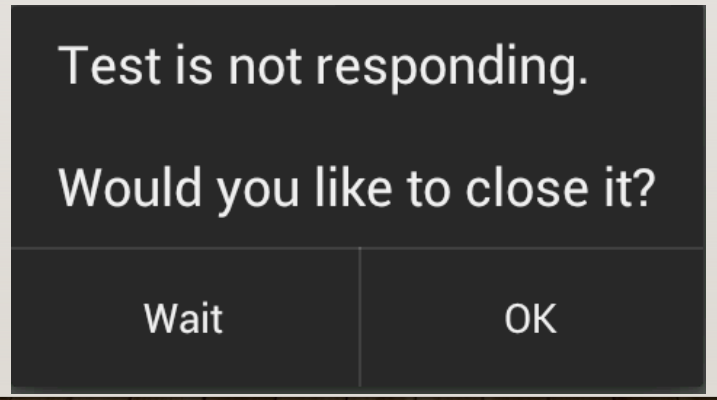
- Each application run in separate process(sandbox)
- There may be many threads of execution.
- All Threads share allocated slices of CPU time managed by the operating system.
- Each thread
  - is a separate sequential flow of control within the overall program
  - executes its instructions in order, one after the other.

MAIN THREAD

- When we start the application, the system creates a thread of execution called **main thread** also known as **UI Thread**
- **Main Thread:**
  - **handles all** interactions with the App UI elements, updating the state and their look on the device screen.
  - Handles all lifecycle and UI elements events.

LONG-RUNNING TASKS IN ANDROID

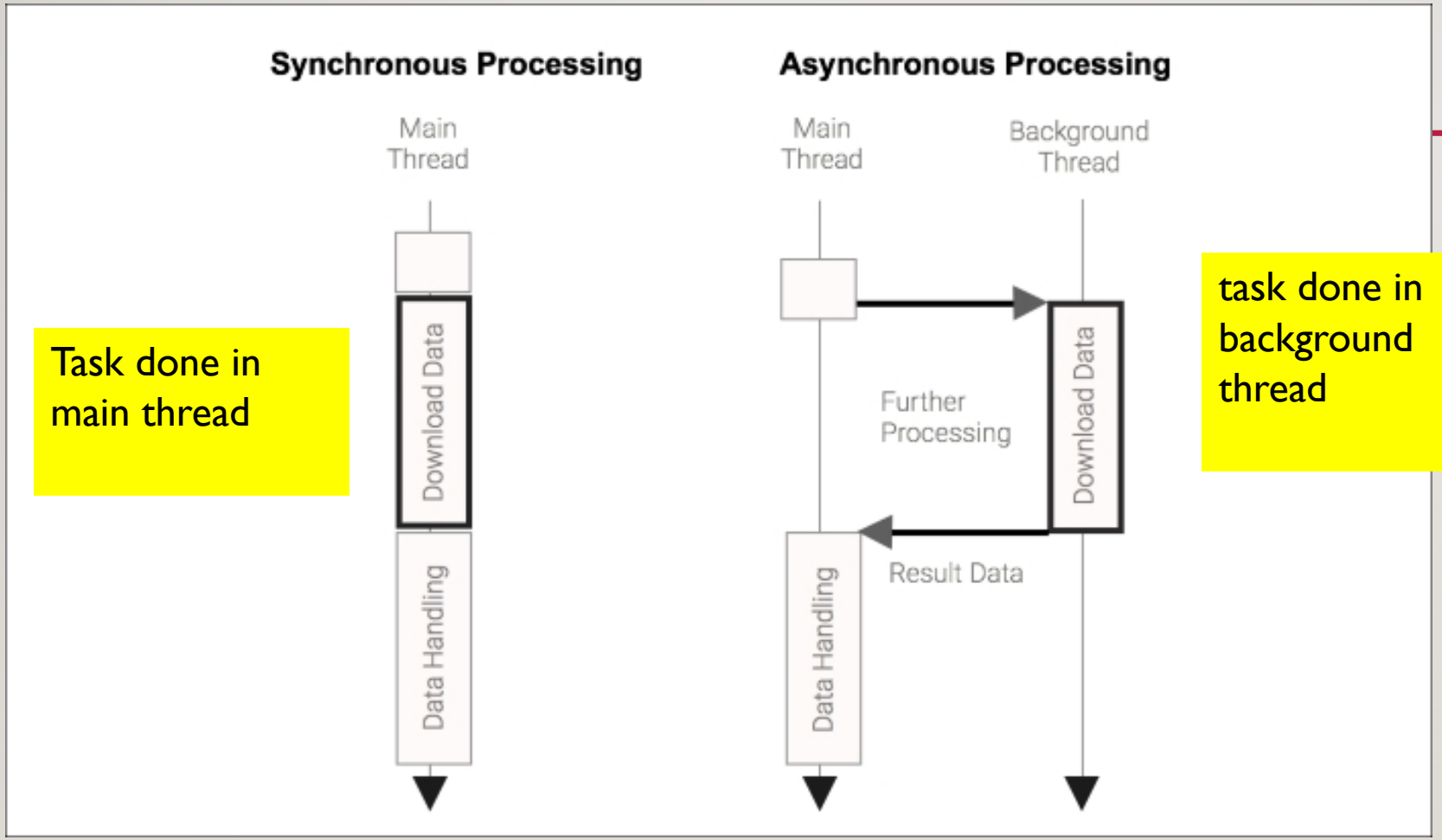
- Avoid any kind of long execution task, such as input/output (I/O) that could block the Main Thread
- If your app doesn't respond to user input within a few seconds, the Android run-time can display the dreaded Application Not Responding (ANR) Dialog:



MAINTAINING RESPONSIVENESS

Time-consuming tasks that should be handled on a background thread

- Network communications
- Debase operations
- Input and output file operations
- Image and video processing
- Complex math calculations



Task done in main thread

task done in background thread

Src: Android: Programming for Developers by John Horton; Helder Vasconcelos; Raul Portales Published by Packt Publishing, 2016

EXAMPLE

```
public void taskA(View v) {

    try {

        Thread.sleep(3000);
        Log.d( ”task",“done“);

     } catch (InterruptedException e) {
            Log.d( ”task",“error“);
             e.printStackTrace();
         }

}
```

EXAMPLE

```
public void taskA(View v) {

    new Thread(new Runnable(){
            public void run(){

            //  task


        }).start();
    }
```

EXAMPLE

```
public void taskA(View v) {

  new Thread(new Runnable(){
        public void run(){

      try {

          Thread.sleep(3000);
          Log.d( "Item","done");

          } catch (InterruptedException e) {
                e.printStackTrace();
           }
        }
   }) .start();
}
```

EXAMPLE

```
public class MyTask implements Runnable {
public void run(){
        Log.d( ”task",“MyTask running“);
                //task

    }
 }

    public void taskA(View v) {
       MyTask myTask = new MyTask()
        Thread thread = new Thread(myTask);
        thread.start();
    }
```

EXAMPLE ADDING USER TO DATABASE

```
public void addUser(User user) {
    new Thread(new Runnable() {
        public void run() {
            // add new User
            database.userDao().addUser(user);

        }
    }).start();
}
```

GETTING DATA FROM DATABASE

- MutableLiveData<List<User>> users = new MutableLiveData<>();
- public void getUsers() {
    new Thread(new Runnable() {
        public void run() {
            users.postValue(database.userDao().getAll());
        }
    }).start();

}

Example Using AsyncTask (now deprecated by android)

- public class MyAsyncTask extends AsyncTask<Void,Void, String> {
  @Override protected String doInBackground(Void... params) {...}
  @Override protected void onPostExecute(String result) {...}
  }

References

- https://developer.android.com/topic/performance/threads
- https://developer.android.com/reference/java/lang/Thread
- https://developer.android.com/reference/android/os/AsyncTask