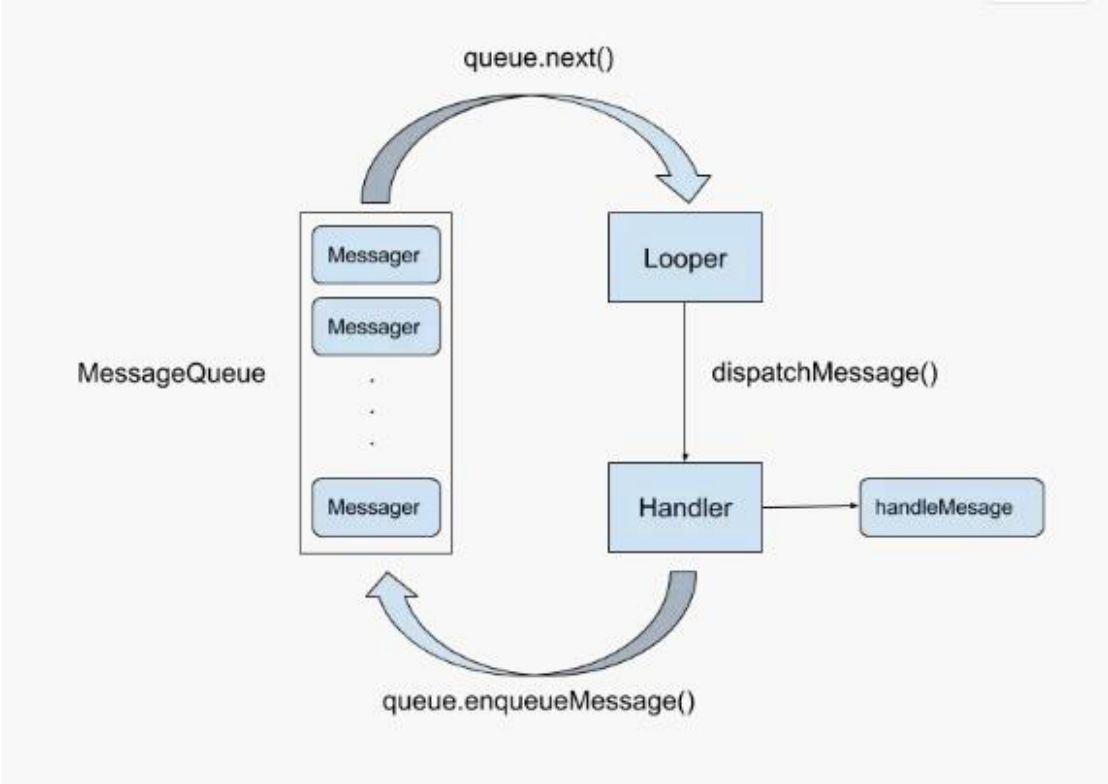
## Handler

### 1、谈谈消息机制 Handler 作用 ？有哪些要素 ？流程是怎样的 ？

- 参考回答：
    - 负责**跨线程通信**，这是因为**在主线程不能做耗时操作，而子线程不能更新 UI**，所以当子线程中进行耗时操作后需要更新 UI 时，通过 Handler 将有关 UI 的操作切换到主线程中执行。
    - 具体分为四大要素
        - **Message（消息）**：需要被传递的消息，消息分为硬件产生的消息（如按钮、触摸）和软件生成的消息。
        - **MessageQueue（消息队列）**：负责消息的存储与管理，负责管理由 Handler 发送过来的 Message。读取会自动删除消息，单链表维护，插入和删除上有优势。在其 next()方法中会无限循环，不断判断是否有消息，有就返回这条消息并移除。
        - **Handler（消息处理器）**：负责 Message 的发送及处理。主要向消息池发送各种消息事件（Handler.sendMessage()）和处理相应消息事件（Handler.handleMessage()），按照先进先出执行，内部使用的是单链表的结构。
        - **Looper（消息池）**：负责关联线程以及消息的分发，在该线程下从 MessageQueue 获取 Message，分发给

Handler，Looper 创建的时候会创建一个 MessageQueue，调用 loop()方法的时候消息循环开始，其中会不断调用 messageQueue 的 next()方法，当有消息就处理，否则阻塞在 messageQueue 的 next()方法中。当 Looper 的 quit()被调用的时候会调用 messageQueue 的 quit()，此时 next()会返回 null，然后 loop()方法也就跟着退出。

- 具体流程如下




  - 在主线程创建的时候会创建一个 Looper，同时也会在在 Looper 内部创建一个消息队列。而在创键 Handler 的时候取出当前线程的 Looper，并通过该 Looper 对象获得消息队列，然后 Handler 在子线程中通过

**MessageQueue.enqueueMessage** 在消息队列中添加一条 Message。

- 通过 **Looper.loop()** 开启消息循环不断轮询调用 **MessageQueue.next()**，取得对应的 Message 并且通过 **Handler.dispatchMessage** 传递给 Handler，最终调用 **Handler.handlerMessage** 处理消息。

## 2、一个线程能否创建多个 Handler，Handler 跟 Looper 之间的对应关系 ？

- 参考回答：
  - **一个 Thread 只能有一个 Looper，一个 MessageQueen，可以有多个 Handler**
  - **以一个线程为基准，他们的数量级关系是： Thread(1) : Looper(1) : MessageQueue(1) : Handler(N)**

## 3、软引用跟弱引用的区别

- 参考回答：
  - **软引用（SoftReference）**：如果一个对象只具有软引用，则内存空间充足时，垃圾回收器就不会回收它；如果内存空间不足了，就会回收这些对象的内存。只要垃圾回收器没有回收它，该对象就可以一直被程序使用。

- **弱引用（WeakReference）**：如果一个对象只具有弱引用，那么在垃圾回收器线程扫描的过程中，一旦发现了只具有弱引用的对象，不管当前内存空间足够与否，都会回收它的内存。
- 两者之间**根本区别**在于：只具有弱引用的对象拥有更短暂的生命周期，可能随时被回收。而只具有软引用的对象只有当内存不够的时候才被回收，在内存足够的时候，通常不被回收。

| 引用类型 | GC回收时间 | 用途 | 生存时间 |
| --- | --- | --- | --- |
| 强引用 | never | 对象的一般状态 | JVM停止运行时 |
| 软引用 | 内存不足时 | 对象缓存 | 内存不足时终止 |
| 弱引用 | GC时 | 对象缓存 | GC后终止 |
| 虚引用 | unknow | unknow | unknow |

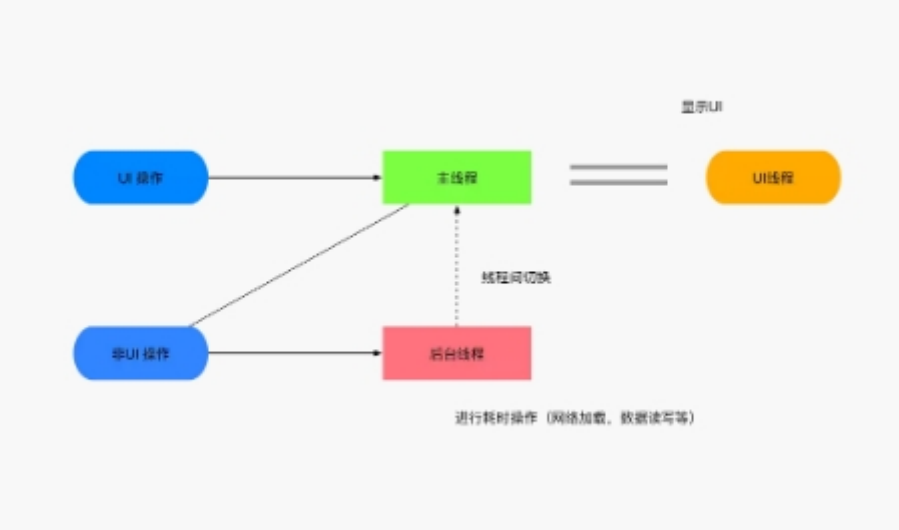
## 4、Handler 引起的内存泄露原因以及最佳解决方案

- 参考回答：
  - 泄露原因：
    - Handler 允许我们发送延时消息，如果在延时期间用户关闭了 Activity，那么该 Activity 会泄露。 这个泄露是因为 Message 会持有 Handler，而又因为 Java 的特性，内部类会持有外部类，使得 Activity 会被 Handler 持有，这样最终就导致 Activity 泄露。

- 解决方案：
  - **将 Handler 定义成静态的内部类，在内部持有 Activity 的弱引用，并在 Acitivity 的 onDestroy()中调用 handler.removeCallbacksAndMessages(null) 及时移除所有消息。**

## 5、为什么系统不建议在子线程访问 UI？

- 参考回答：
  - Android 的 UI 控件不是**线程安全**的，如果在多线程中并发访问可能会导致 UI 控件处于不可预期的状态
  - 这时你可能会问为何系统不对 UI 控件的访问加上锁机制呢？因为
    - 加锁机制会让 UI 访问逻辑变的复杂
    - 加锁机制会降低 UI 的访问效率,因为加锁会阻塞某些线程的执行



### 6、Looper 死循环为什么不会导致应用卡死？

- 参考回答：
    - 主线程的主要方法就是**消息循环**，一旦退出消息循环，那么你的应用也就退出了，Looer.loop（）方法可能会引起主线程的阻塞，但只要它的消息循环没有被阻塞，能一直处理事件就不会产生 ANR 异常。
    - 造成 **ANR** 的不是主线程阻塞，而是主线程的 Looper 消息处理过程发生了**任务阻塞**，无法响应手势操作，不能及时刷新 UI。
    - **阻塞与程序无响应**没有必然关系，虽然主线程在没有消息可处理的时候是阻塞的，但是只要保证有消息的时候能够立刻处理，程序是不会无响应的。

### 7、使用 Handler 的 postDealy 后消息队列会有什么变化？

- 参考回答：
    - 如果队列中只有这个消息，那么消息不会被发送，而是计算到时唤醒的时间，先将 Looper 阻塞，到时间就唤醒它。但如果此时要加入新消息，该消息队列的对头跟 delay 时间相比更长，则插入到头部，按照触发时间进行排序，队头的时间最小、队尾的时间最大

### 8、可以在子线程直接 new 一个 Handler 吗？怎么做？

- 参考回答：
    - 不可以，因为在**主线程**中，Activity 内部包含一个 Looper 对象，它会自动管理 Looper，处理子线程中发送过来的消息。而对于**子线程**而言，没有任何对象帮助我们维护 Looper 对象，所以需要我们自己手动维护。所以要在子线程开启 Handler 要先创建 Looper，并开启 Looper 循环

```
//代码示例
new Thread(new Runnable() {
            @Override
            public void run() {
                Looper.prepare();
                new Handler() {
                    @Override
                    public void handleMessage(Message msg) {
                        super.handleMessage(msg);
                    }
                };
                Looper.loop();
            }
        }).start();
```

### 9、Message 可以如何创建？哪种效果更好，为什么？

- 参考回答：可以通过三种方法创建：
    - 直接生成实例 **Message m = new Message**
    - 通过 **Message m = Message.obtain**
    - 通过 **Message m = mHandler.obtainMessage()**

- 后两者效果更好，因为 Android 默认的消息池中消息数量是 10，而后两者是直接在消息池中取出一个 Message 实例，这样做就可以避免多生成 Message 实例。