


# Java 物件導向程式語言

## 多執行緒程式設計

正修科技大學資訊工程系


講師：林彥宏  
[lyhcode.info](mailto:lyhcode.info)  
[lyhcode@gmail.com](mailto:lyhcode@gmail.com)



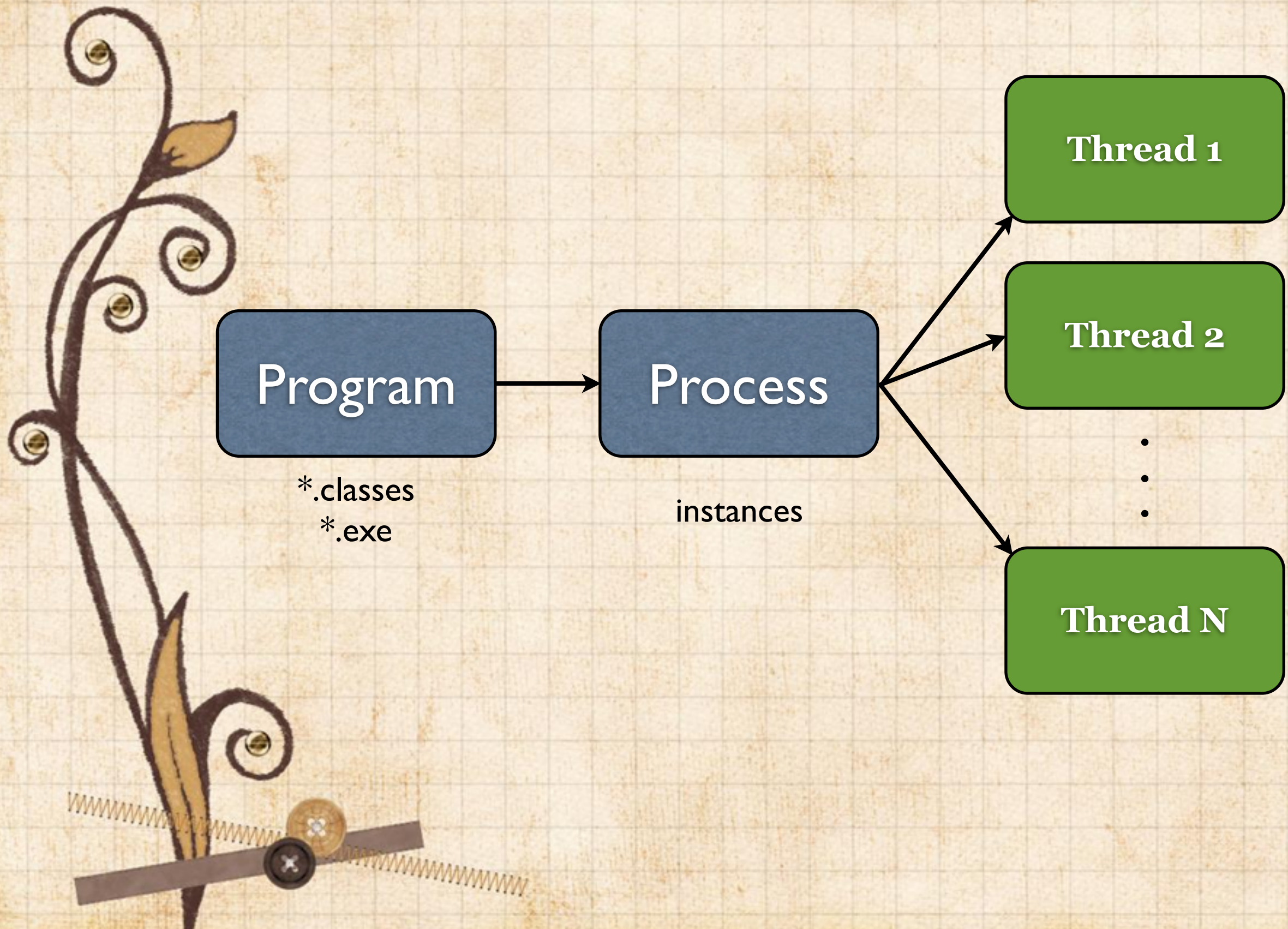


簡報下載

<http://goo.gl/8Hc1iX>

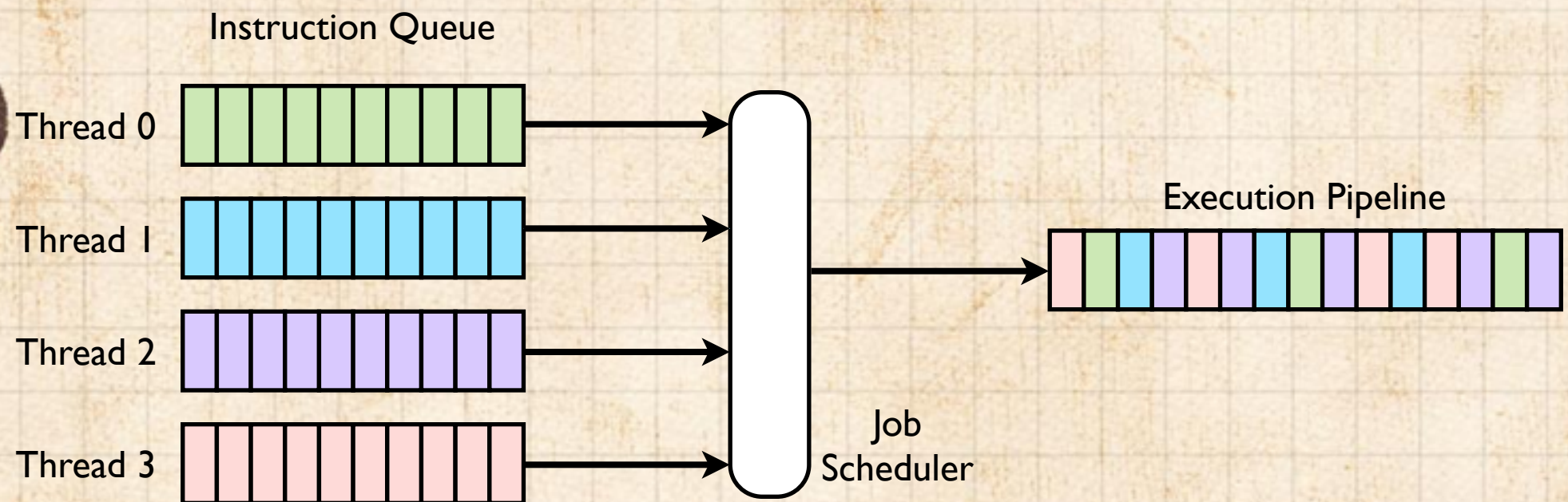






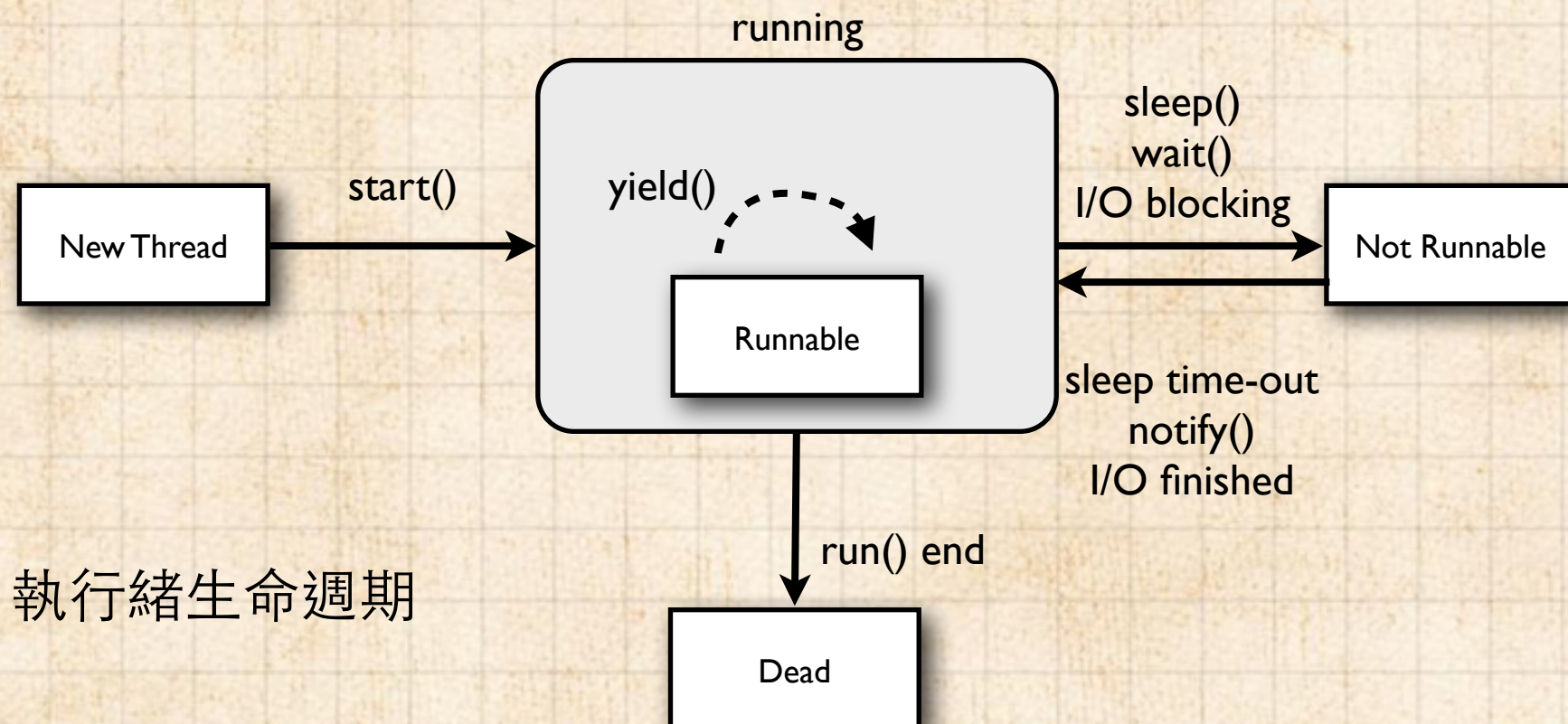


# 多執行緒處理佇列






# Thread Life Cycle



執行緒生命週期



# Thread and Runnable



```
new Thread();
```

```
new RunnableImpl();
```

```
new Thread(new RunnableImpl());
```



# Create a Thread

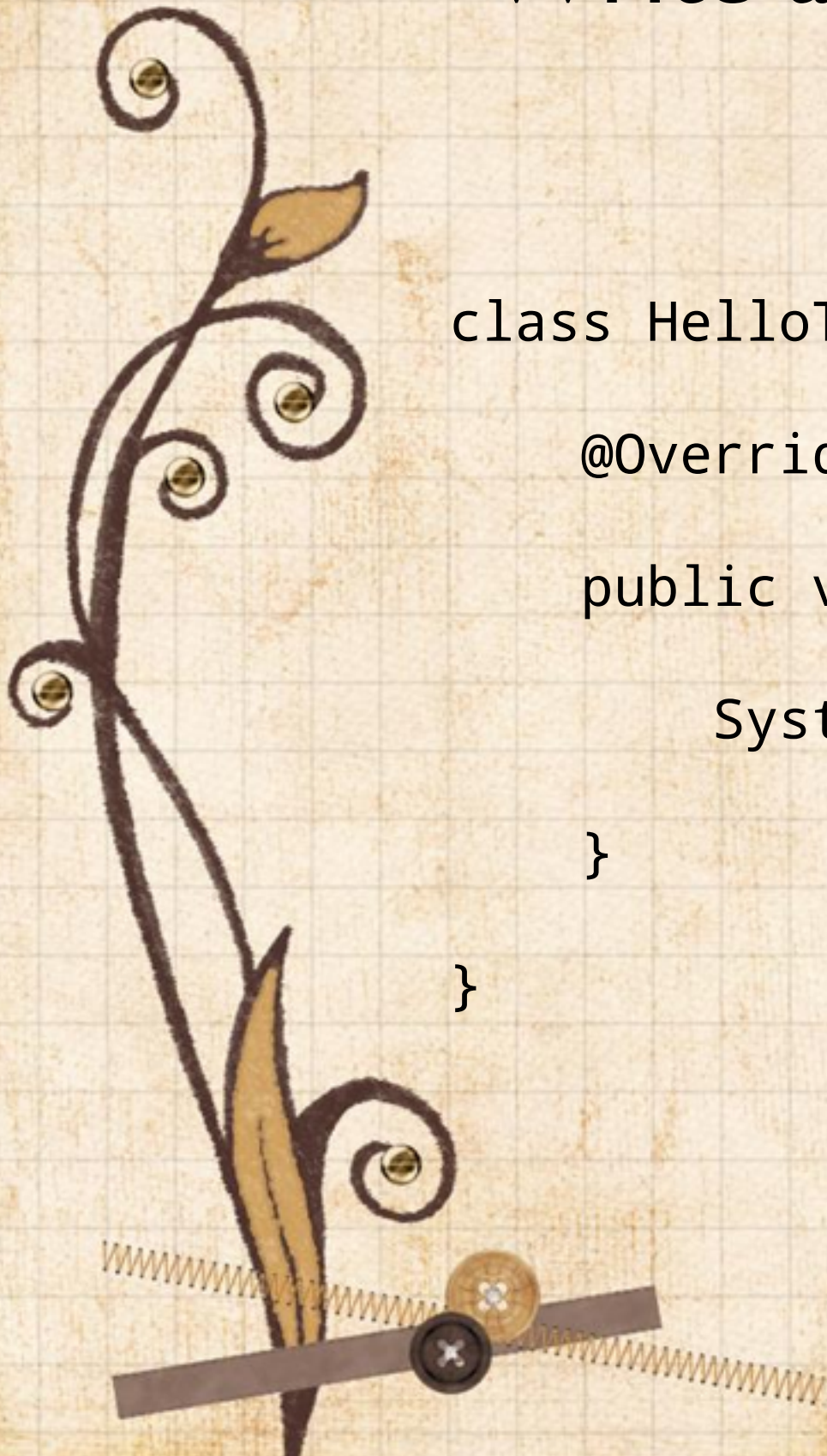
```
Thread thread = new Thread();
```





# Write a HelloThread class

```
class HelloThread extends Thread {  
    @Override  
    public void run() {  
        System.out.println("Hello");  
    }  
}
```



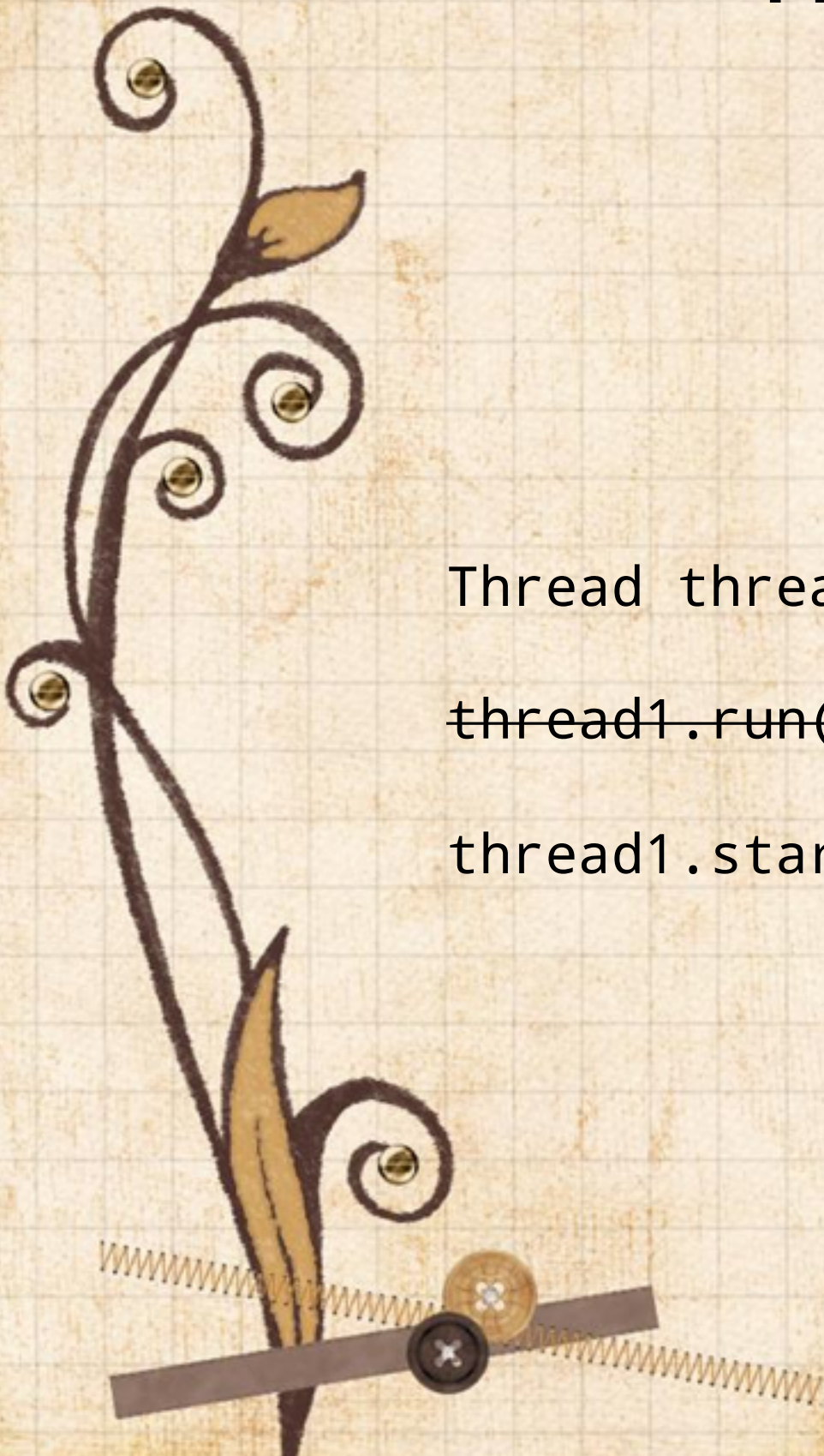


# Thread .start()

```
Thread thread1 = new HelloThread();
```


```
thread1.run(); //這不會以多執行緒方式執行
```

```
thread1.start();
```





# HelloThread with constructor



```
class HelloThread extends Thread {  
    private final String who;  
    public HelloThread(String name) {  
        who = name;  
    }  
    @Override  
    public void run() {  
        System.out.println("Hello " + who);  
    }  
}
```



# Start three thread

```
Thread thread1 = new HelloThread("John");
```

```
Thread thread2 = new HelloThread("Mary");
```

```
Thread thread3 = new HelloThread("Kyle");
```

```
// 猜猜執行結果？
```

```
thread1.start();
```

```
thread2.start();
```

```
thread3.start();
```





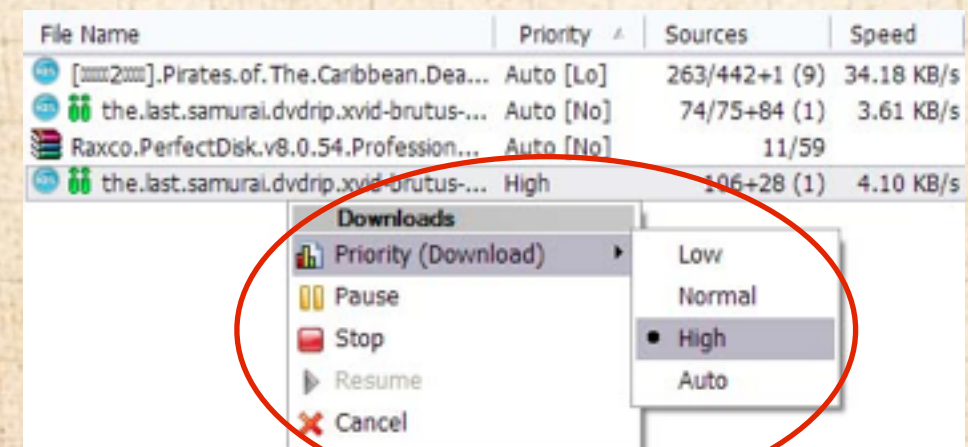
# Thread Priority

Thread.MAX\_PRIORITY //10

Thread.MIN\_PRIORITY //1

Thread.NORM\_PRIORITY //5

//優先權高，得到處理器時間的機會較高，但是“不保證”



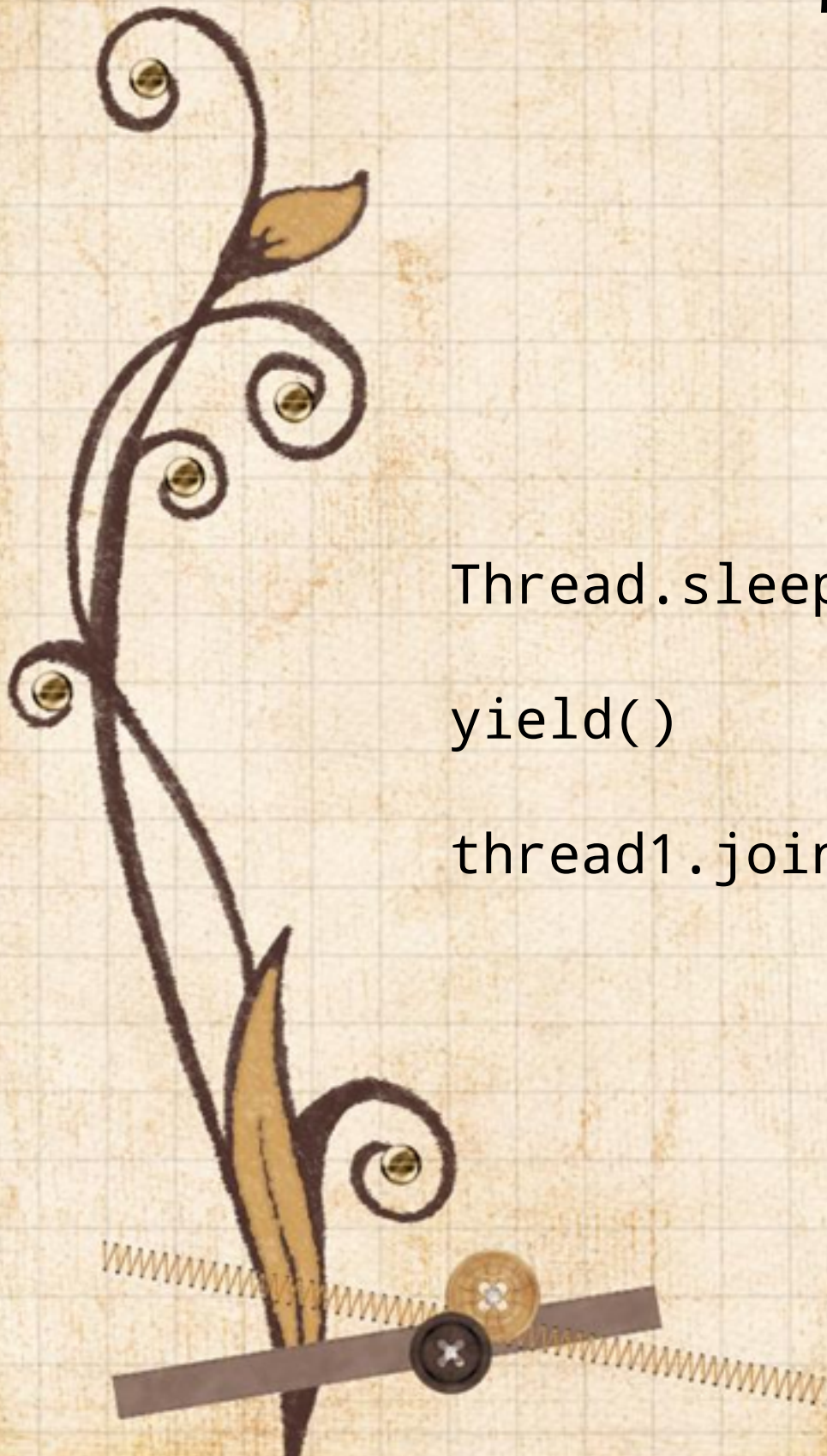


# 執行緒控制

`Thread.sleep(int time)` //讓目前的執行緒休息


`yield()` //讓給其他執行緒先處理

`thread1.join()` //等thread1結束再處理






# CountDownThread



```
class CountDownThread extends Thread {  
    private final int num;  
    public CountDownThread(int num) {  
        this.num = num;  
    }  
    @Override  
    public void run() {  
        for (int i = num; i >= 0; i--) {  
            System.out.println(i);  
        }  
    }  
}
```

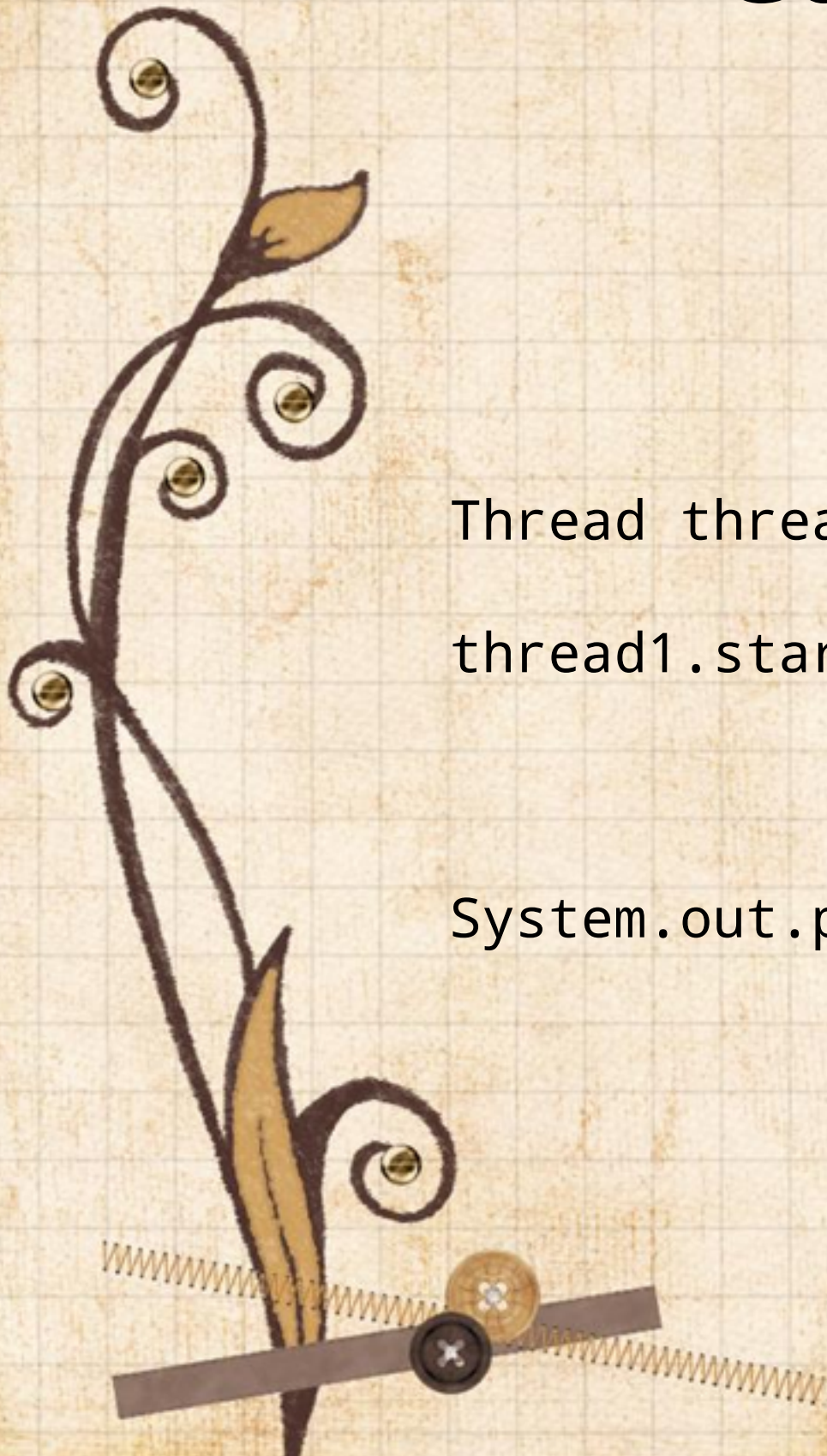




# Count Down !!!

```
Thread thread1 = new CountDownLatch(100);  
thread1.start();
```

```
System.out.println("END");
```





# Count Down with join()

```
Thread thread1 = new CountDownLatch(100);  
thread1.start();
```

```
thread1.join();
```

```
System.out.println("END");
```



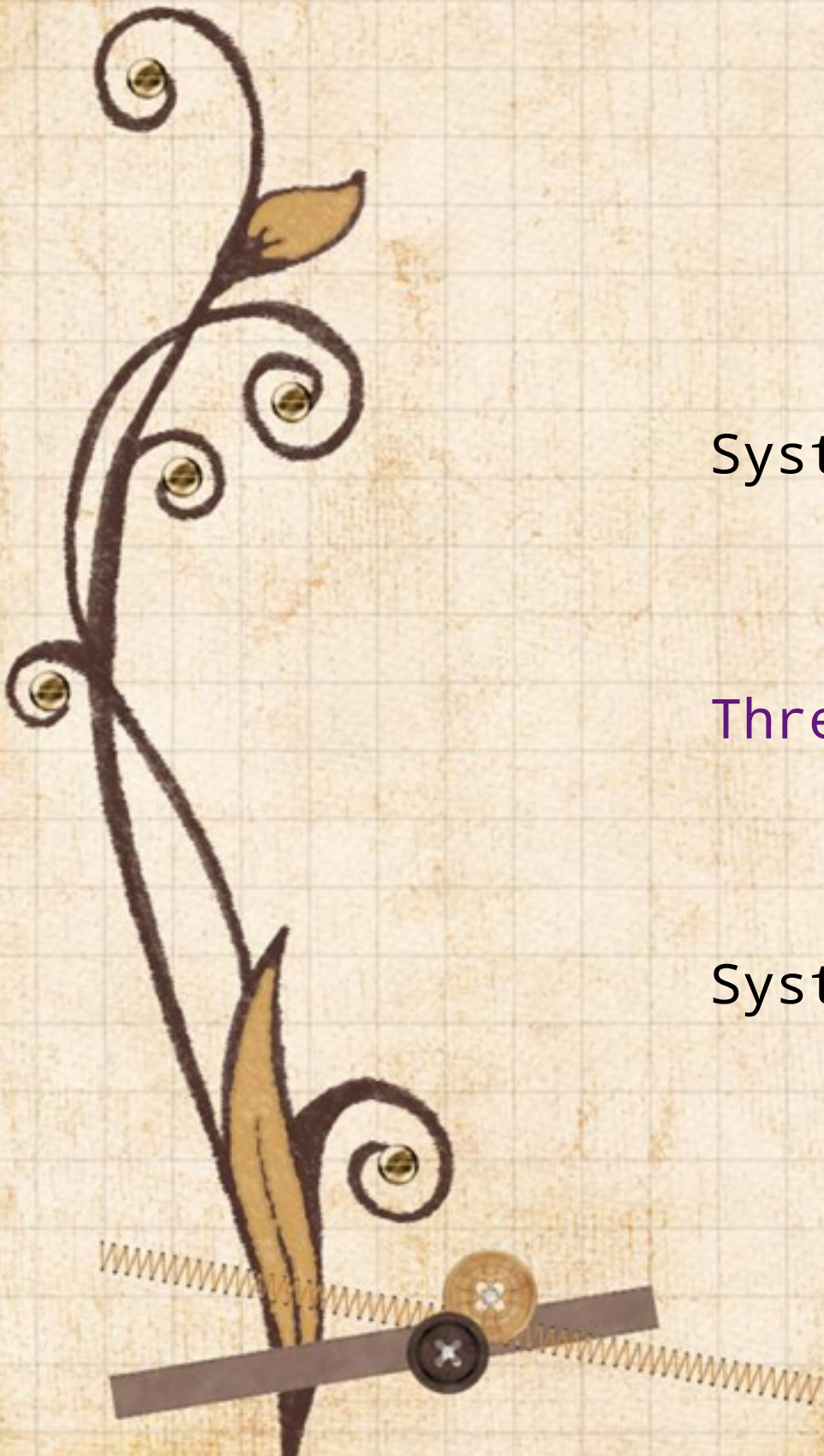


# Sleep ...

```
System.out.println(new Date());
```


```
Thread.sleep(5000);
```

```
System.out.println(new Date());
```






# Counter



```
class Counter {  
    public int count = 0;  
    private Object obj = new Object();  
    public void plus() { count++; }  
    public synchronized void syncPlus() { count++; }  
    public void syncPlus2() {  
        synchronized(obj) {  
            count++;  
        }  
    }  
}
```



# CounterThread



```
class CounterThread extends Thread {  
    private final Counter counter;  
    public CounterThread(Counter counter) {  
        this.counter = counter;  
    }  
    @Override  
    public void run() {  
        counter.plus();  
        //counter.syncPlus();  
        //counter.syncPlus2();  
    }  
}
```



# 停止執行緒

```
Thread thread1 = new SomeThread();
```

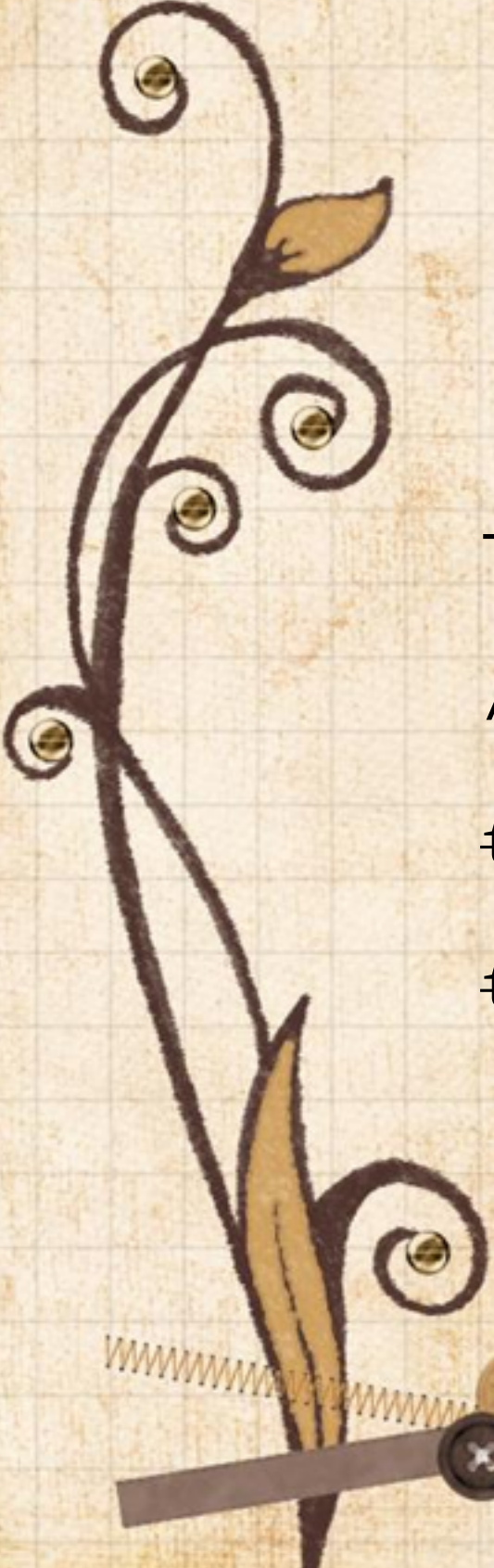
```
//deprecated
```

```
thread1.stop();
```





# 暫停、繼續執行緒



```
Thread thread1 = new SomeThread();
```


```
//deprecated
```

```
thread1.suspend();
```

```
thread1.resume();
```



# Runnable Interface



```
class SomeClass implements Runnable {  
    @Override  
    public void run() {  
        //...  
    }  
}  
  
new Thread(new SomeClass()).start();
```




# Anonymous Runnable

```
new Thread(new Runnable() {  
    @Override  
    public void run() {  
        //....  
    }  
}).start();
```






# Game




```
class Game implements Runnable {  
    private boolean isContinue = true;  
    private boolean isSuspend = false;  
    private int round = 0;  
  
    @Override  
    public void run() {  
        //...  
    }  
  
    public void stop() { isContinue = false; }  
    public void suspend() { isSuspend = true; }  
    public void resume() { isSuspend = false; }  
}
```



# Game run()



```
@Override
public void run() {
    while (isContinue) {
        while (isContinue && !isSuspend) {
            try {
                Thread.sleep(1000);
                System.out.println("round " + ++round);
            } catch (InterruptedException ex) {
                ex.printStackTrace();
            }
        }
    }
}
```






# Control the Game



```
Game game = new Game();  
Thread thread = new Thread(game);  
thread.start();  
Thread.sleep(2000);  
game.suspend();  
Thread.sleep(2000);  
game.resume();  
Thread.sleep(3000);  
game.stop();
```





# Swing Counter Practice



Start

1

2

3

4

5

6

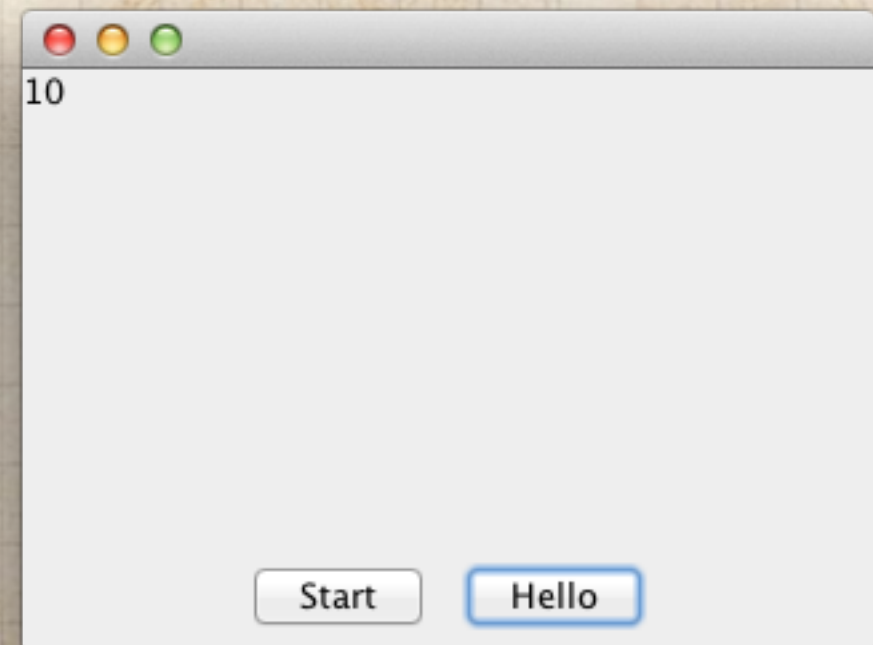
7

8

Hello

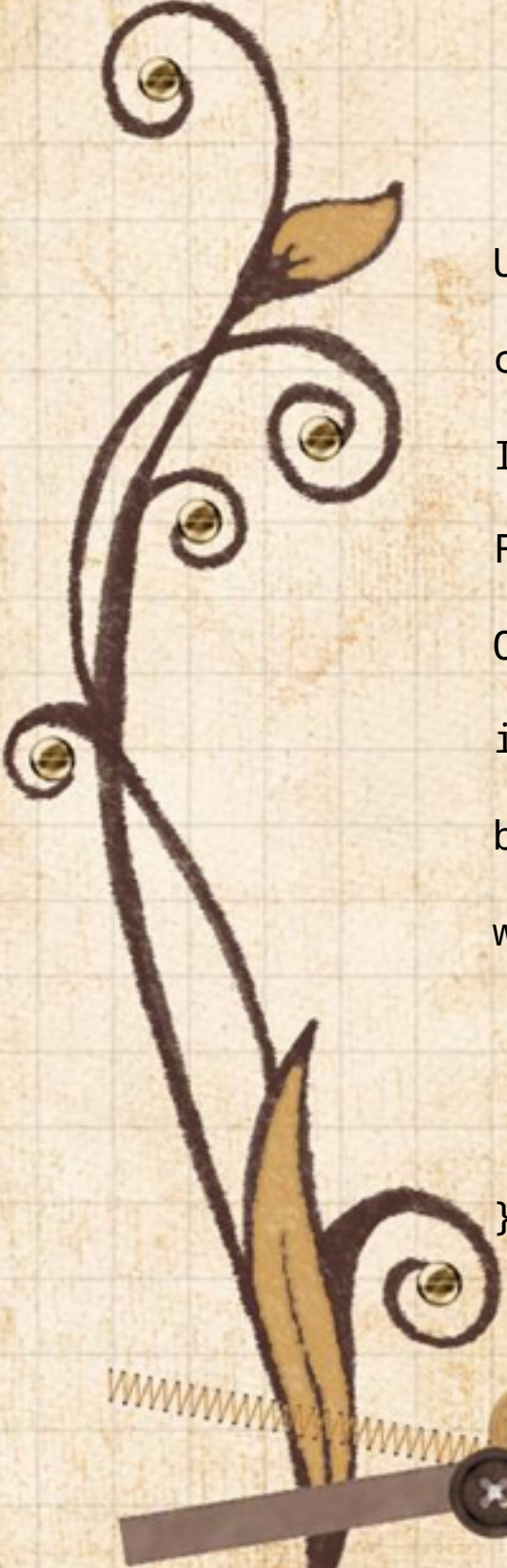
9

10





# File Downloader Practice



```
URLConnection conn = url.openConnection();  
conn.connect();  
InputStream stream = conn.getInputStream();  
File saveToFile = new File(destPath, url.getFile());  
OutputStream outputStream = new FileOutputStream(saveToFile);  
int len = 0;  
byte[] bytes = new byte[1024];  
while ((len = stream.read(bytes)) >= 0) {  
    System.out.printf("Get %d bytes from %s\n", len, url);  
    outputStream.write(bytes, 0, len);  
}
```



# 範例程式原始碼下載

<https://github.com/lyhcode/JavaThread>

