

- CS117 OOP

- Chapter 8

Java Thread



By Dr. Paween Khoenkaw
Computer Science MJU

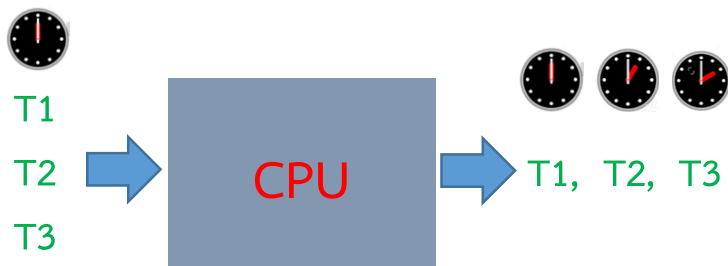


Task Execution Model

Task คืองาน หรือ หน่วยย่อยของโปรแกรมคอมพิวเตอร์

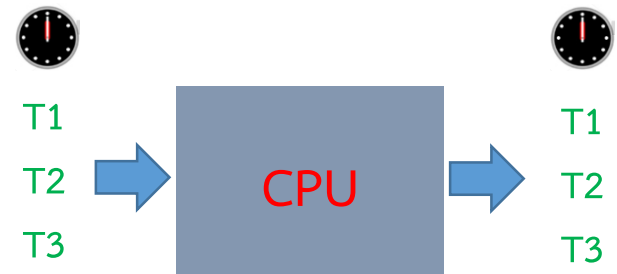
โดยหนึ่งโปรแกรมอาจจะแบ่งการทำงานได้เป็นหลาย Task

ระบบปฏิบัติการ จะนำ Task ไปทำการ Execute โดยมีรูปแบบหลักๆอยู่ 2 แบบคือ Single-Tasking และ Multi-tasking



Single-Tasking

ทำทีละงาน



Multi-Tasking

ทำพร้อมกันหลายงาน

Multi-Tasking Execution Model

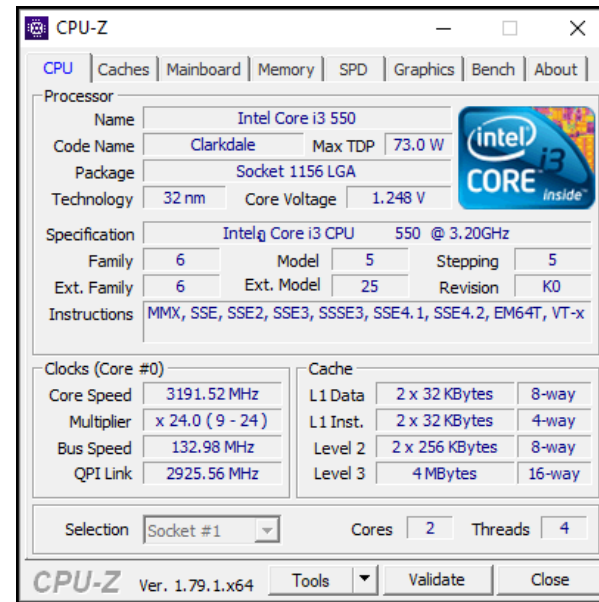
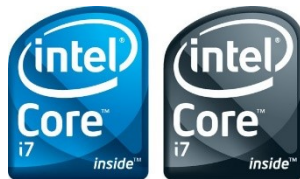
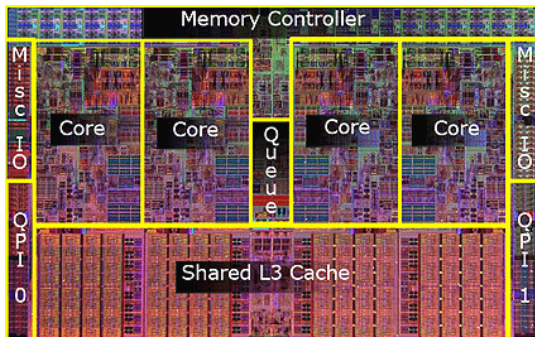
Multi-tasking มีรูปแบบการทำงานหลักอยู่ 2 แบบคือ

1 Concurrent Programming

- ใช้ CPU เดียว แต่แบ่งให้แต่ละโปรแกรมสลับกันใช้ Execute ทีละ Task
- Thread

2 Parallel Programming

- ใช้ CPU หลายตัว แยกกัน Execute แต่ละ Task ไปพร้อมๆ กัน
- Core



Thread

Task ที่ทำงานแบบ multi-tasking เรียกว่า Thread

Thread เป็นส่วนหนึ่งของ Process (คอมพิวเตอร์โปรแกรม)

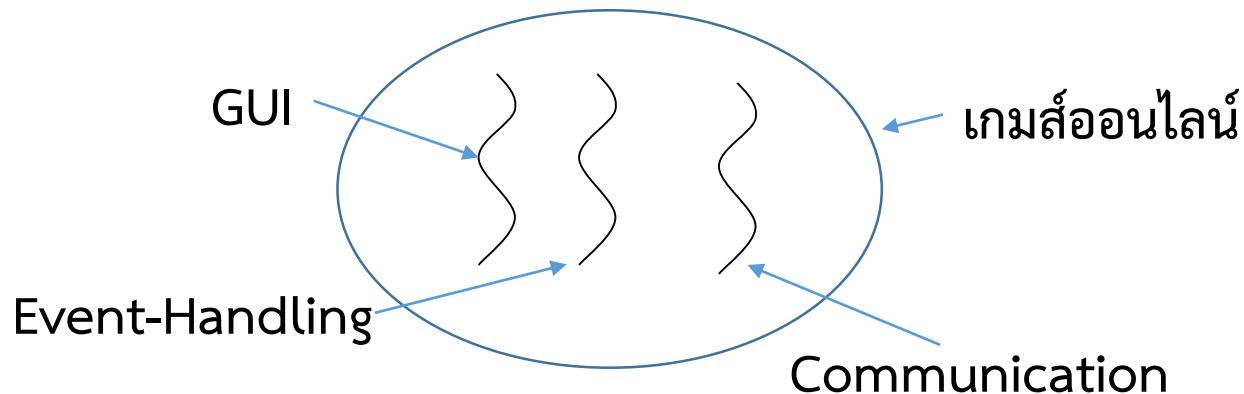
Thread หลายตัวอาจจะทำงานร่วมกัน เพื่อเป้าหมายเดียวกัน

ตัวอย่างเช่น Process เกมออนไลน์ จะประกอบไปด้วย

Thread ตัวที่ 1 สำหรับวาดภาพบนหน้าจอ

Thread ตัวที่ 2 สำหรับตรวจจับการกดแป้นพิมพ์ของผู้เล่น

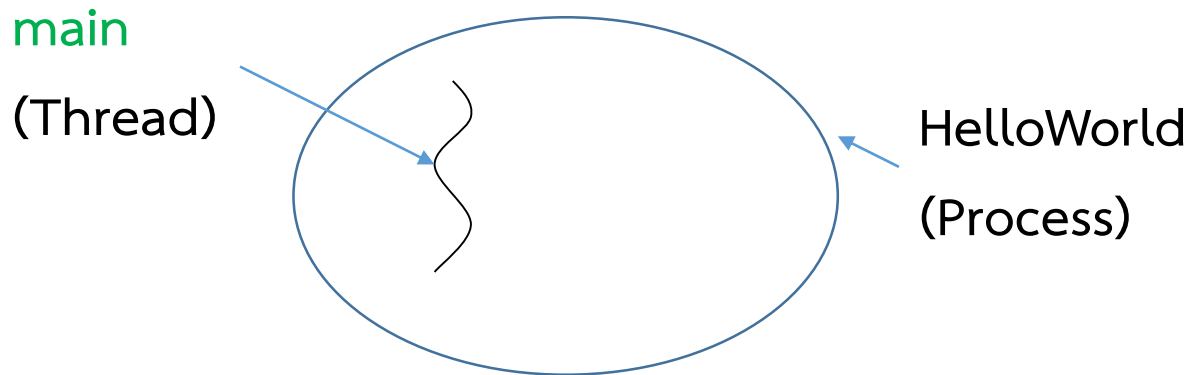
Thread ตัวที่ 3 สำหรับรับ-ส่งข้อมูลผ่านอินเทอร์เน็ต



Thread

Java มีการทำงานแบบ Thread โดยโปรแกรมที่เล็กที่สุดจะมี 1 Thread เรียกว่า Main Thread

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello World");  
    }  
}
```



Thread

ข้อดีของการใช้ Thread บน Java

1 เขียนโปรแกรมง่าย

- 1 thread ต่อ 1 task

2 โปรแกรมทำงานได้ดี

- thread จะทำงานเมื่อต้องการใช้งานเท่านั้น
- ไม่ต้องรอให้งานอื่นทำงานจนเสร็จก่อน

3 หลาย thread สามารถใช้ทรัพยากรร่วมกันได้

- ใช้ตัวแปรเดียวกันได้

4 ใช้หน่วยประมวลผลแบบหลาย core ได้อย่างมีประสิทธิภาพ

- แต่ละ thread สามารถทำงานพร้อมกันได้โดยไม่ต้องสลับกันใช้ CPU

Thread

ข้อเสียของการใช้ Thread บน Java

- 1 การสร้าง thread ใหม่จะใช้ทรัพยากรเพิ่ม
 - ใช้หน่วยความจำเพิ่ม
 - ใช้เวลาการทำงานเพิ่มในการสร้าง thread ใหม่
- 2 การสื่อสารระหว่าง thread ทำได้ยาก
 - ต้องตรวจสอบข้อมูลในตัวแปรว่าถูกต้องหรือไม่
- 3 หากมีหลาย thread อาจเกิด deadlock ขึ้นได้

Thread

Deadlock



1 Thread

Thread

Deadlock



ล้าง



ซัก



ปั่นหมาด



ล้าง → ซัก → ปั่น → ล้าง → ปั่น → ล้าง → ปั่น



รอ → ล้าง → ซัก → รอ → ปั่น → ล้าง → รอ → ปั่น → ล้าง → ปั่น

2 Threads (same task)

Thread

Deadlock

Resource



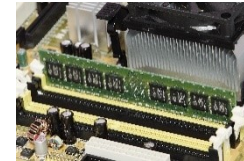
ล้าง



ชัก



ปั่นหมาด



ล้าง → รอ → รอ → รอ



รอ → รอ → รอ → รอ

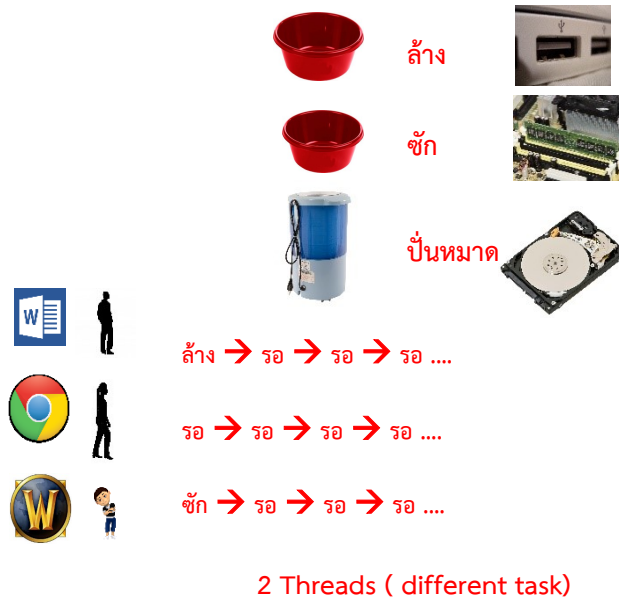


ชัก → รอ → รอ → รอ

3 Threads (different task)

Thread

Deadlock



ระบบปฏิบัติการ ป้องกันการเกิด deadlock ด้วยวิธีการ Preemption (บังคับ)

Preemption คือการยึด Resource กลับมาให้ process อื่นใช้ก่อน แล้วค่อยคืน Resource นั้นกลับไปให้ Process เดิมใช้ต่อจนกว่าจะเสร็จงาน

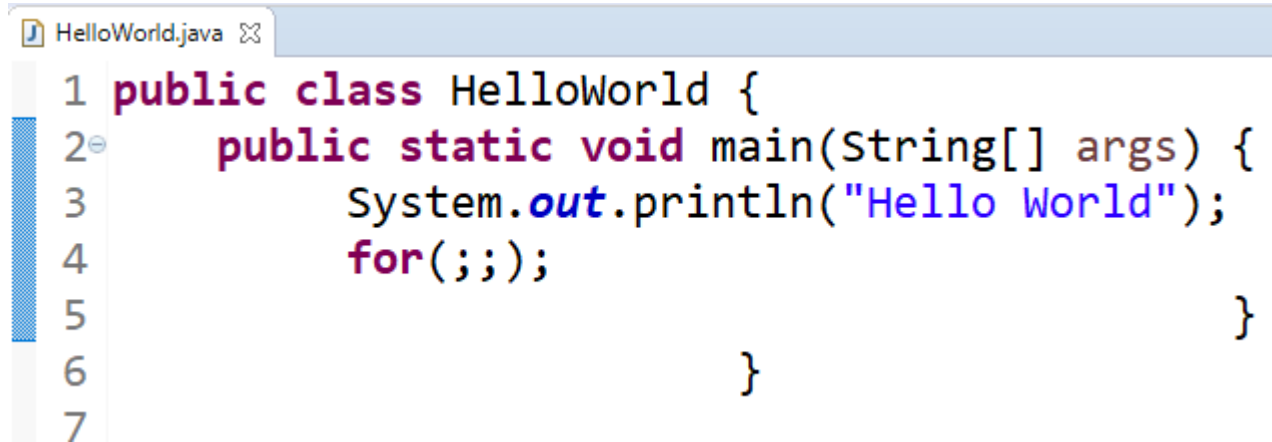
ไม่ใช่ทุก platform จะมี Preemption

ดังนั้นหากเราต้องใช้ resource ที่ใช้พร้อมกันไม่ได้ เราต้องไม่ใช้ resource นั้นนานเกินไป เพื่อป้องกันไม่ให้เกิด deadlock

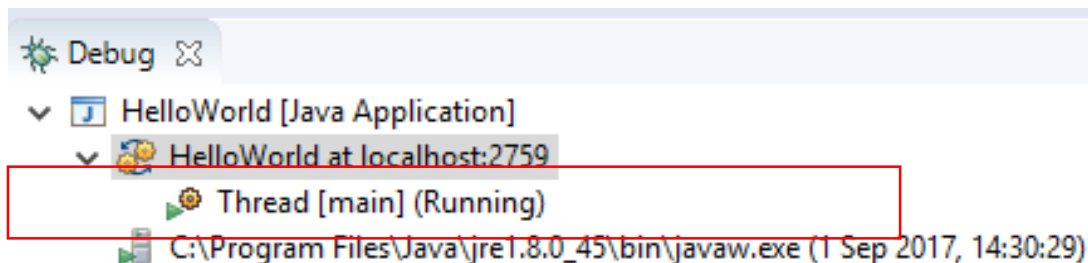
วิธีนี้เรียกว่า Voluntarily (เสียสละ)

Thread

หลาย class บน Java เป็น multi-thread อยู่แล้ว โดยเราไม่ต้องโปรแกรมเพิ่ม



```
1 public class HelloWorld {  
2     public static void main(String[] args) {  
3         System.out.println("Hello World");  
4         for(;;);  
5     }  
6 }  
7
```



Debug

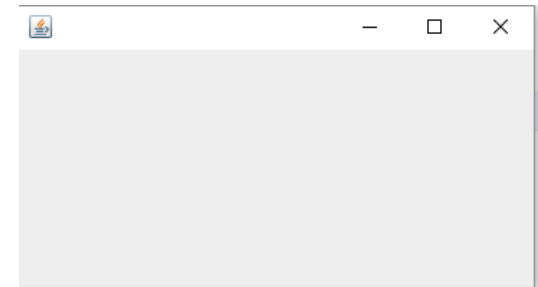
- HelloWorld [Java Application]
 - HelloWorld at localhost:2759
 - Thread [main] (Running)
- C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (1 Sep 2017, 14:30:29)

1 Thread

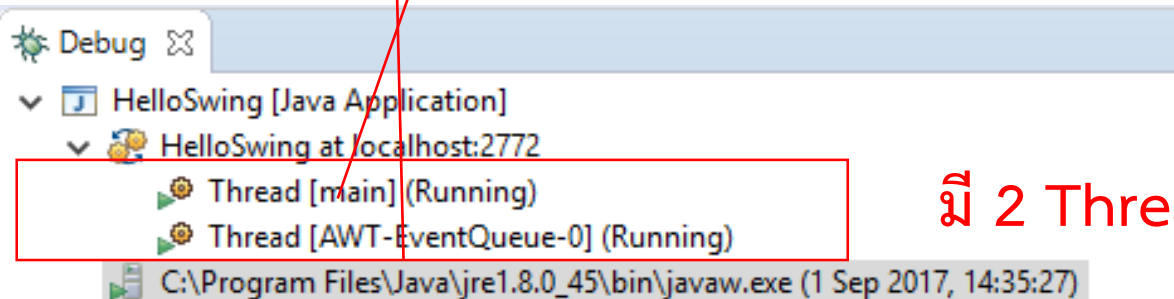
Thread

หลาย class บน Java เป็น multi-thread อยู่แล้ว โดยเราไม่ต้องโปรแกรมเพิ่ม

```
1 import javax.swing.JFrame;
2
3 public class HelloSwing {
4     public static void main(String[] args) {
5
6         JFrame j=new JFrame();
7         j.show();
8         System.out.println("Hello World");
9         for(;;);
10    }
11 }
12
```



Console Tasks
HelloSwing [Java Application] C:\Program File
Hello World



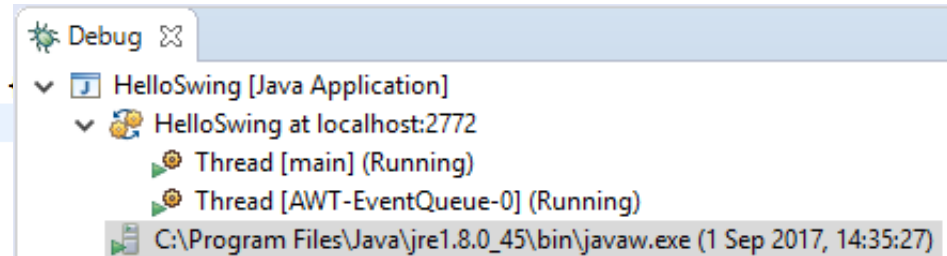
มี 2 Thread

Thread

งานประเภทไหนที่ต้องสร้างเป็น thread

- งานที่มี blocking loop (ต้องการค้างบางอย่างไว้บนจอ)
- งานที่ต้องมีการรออะไรบางอย่าง (รอให้ user กดปุ่ม)
- งานที่สามารถทำได้ล่วงหน้า โดยไม่ต้องรอให้งานก่อนหน้านั้นเสร็จก่อน
- Task มีลำดับความสำคัญไม่เท่ากัน

```
1 import javax.swing.JFrame;  
2  
3 public class HelloSwing {  
4     public static void main(String[] args) {  
5  
6         JFrame j=new JFrame();  
7         j.show();  
8         System.out.println("Hello World");  
9         for(;;);  
10    }  
11 }  
12
```



Blocking loop

ต้องคอยตรวจสอบว่ามีการ click หรือไม่

Thread

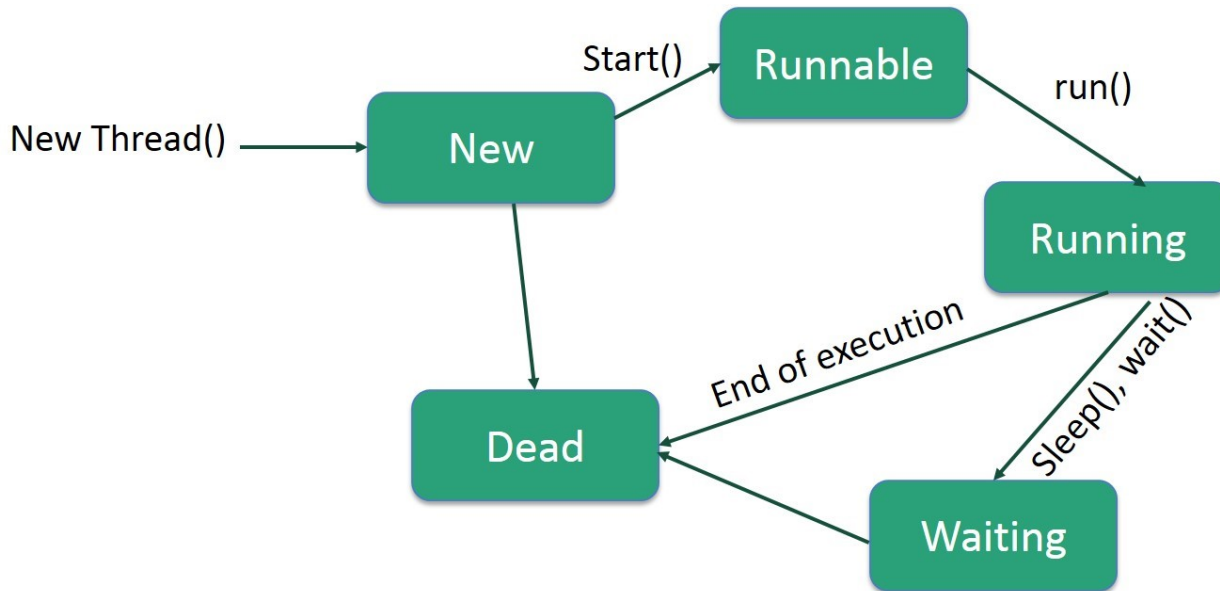
เราสามารถสร้าง class ของเราเองให้เป็น thread ได้

1) ใช้ Thread Class

2) ใช้ Runnable Interface

Thread	Runnable
- คลาส - สืบทอดคลาสไม่ได้ - ทำงานเสร็จแล้วจะถูกทำลาย - มีขนาดใหญ่ - ใช้งานได้ทันที - ควบคุมการทำงานได้ - เหมาะกับงานที่มีความซับซ้อน	- อินเตอร์เฟส - สืบทอดคลาสได้ - นำกลับมาใช้ใหม่ได้ - มีขนาดเล็ก - ทำงานเองไม่ได้ ต้องใช้คลาส thread ช่วยทำงาน - ควบคุมไม่ได้ ทำได้แค่สั่งให้เริ่มทำงาน - เหมาะกับงานง่ายๆ

Thread



Thread class และ Runnable interface จะมีวัฏจักรชีวิตที่คล้าย

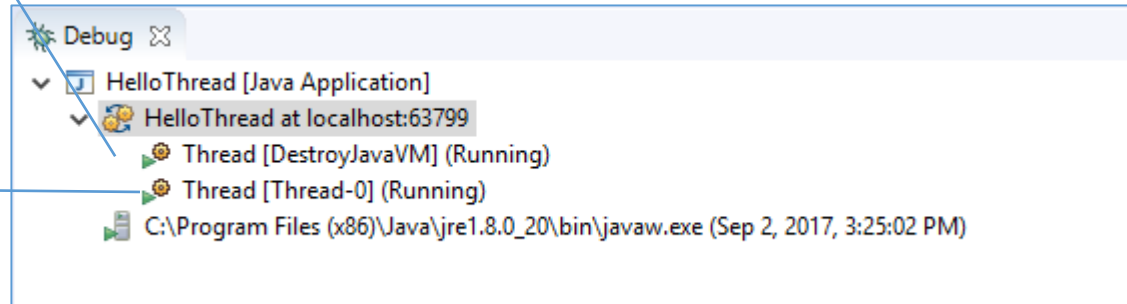
Thread class เมื่อถูกสร้าง instance (new) จะมีการเรียกใช้ constructor และ thread จะเริ่มทำงาน เมื่อถูกใช้ method start()

เมื่อเริ่มทำงาน method run() จะถูกเรียกใช้

Runnable interface จะไม่มี constructor จะเริ่มทำงาน method run() จะถูกเรียกใช้

Thread: Hello Thread

```
public class HelloThread {  
    public static void main(String[] args) {  
        Thread task=new Thread(){  
            @Override  
            public void run() {  
                while(true)  
                {  
                    System.out.println(Calendar.getInstance().getTime());  
                    try {  
                        Thread.sleep(1000);  
                    } catch (InterruptedException e) {  
                        e.printStackTrace();  
                    }  
                }  
            }  
        };  
        task.start();  
    }  
}
```



```
HelloThread [Java Application] C:\Program Files (x86)  
Sat Sep 02 15:22:26 ICT 2017  
Sat Sep 02 15:22:27 ICT 2017  
Sat Sep 02 15:22:28 ICT 2017  
Sat Sep 02 15:22:29 ICT 2017  
Sat Sep 02 15:22:30 ICT 2017
```

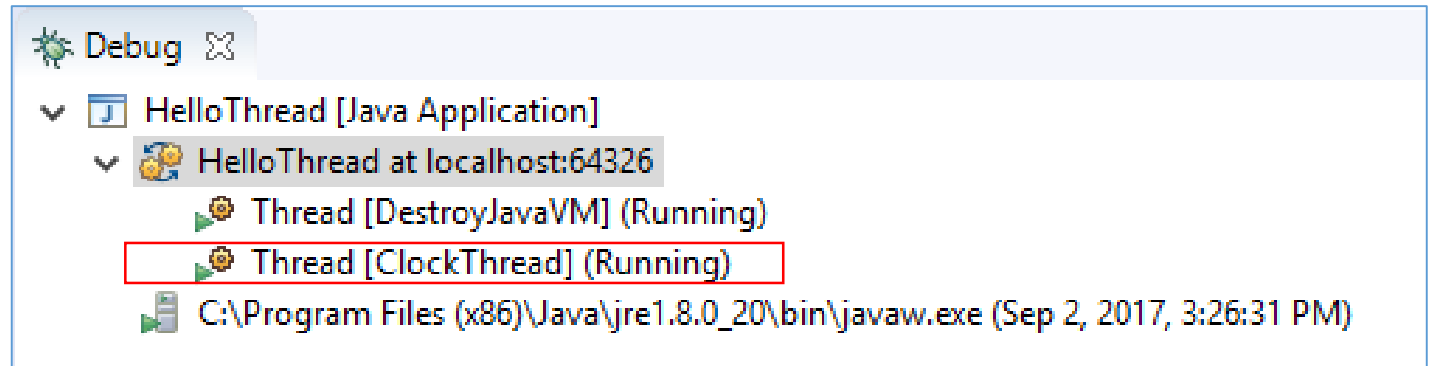
รู้ได้อย่างไรว่า Thread ไหน คือ code อะไร ?

ตอบ: ไม่รู้ เดาล้วนๆ

Thread: Hello Thread

```
public class HelloThread {  
    public static void main(String[] args) {  
        Thread task=new Thread("ClockThread"){  
            @Override  
            public void run() {  
                while(true)  
                {  
                    System.out.println(Calendar.getInstance().getTime());  
                    try {  
                        Thread.sleep(1000);  
                    } catch (InterruptedException e) {  
                        e.printStackTrace();  
                    }  
                }  
            }  
        };  
        task.start();  
    }  
}
```

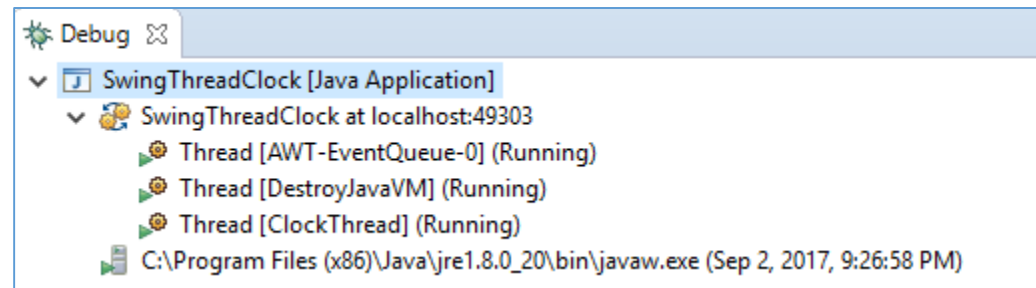
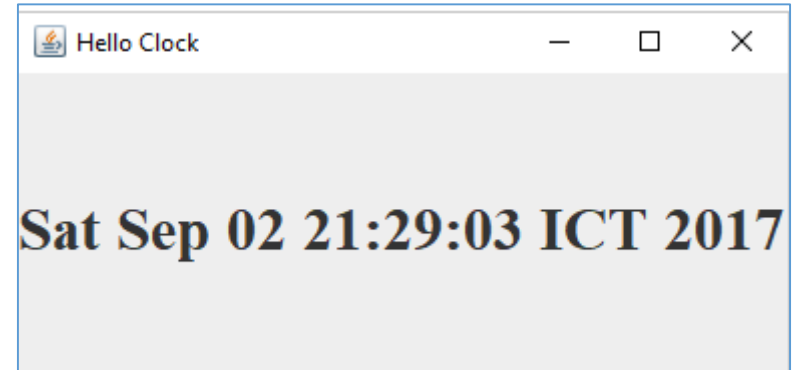
ทำไมต้องตั้งชื่อ Thread ?



ตอบ: เพื่อให้เช็คได้ว่า thread ไหนหยุดทำงานไปแล้ว หรือยังทำงานอยู่

Thread: Thread and Swing

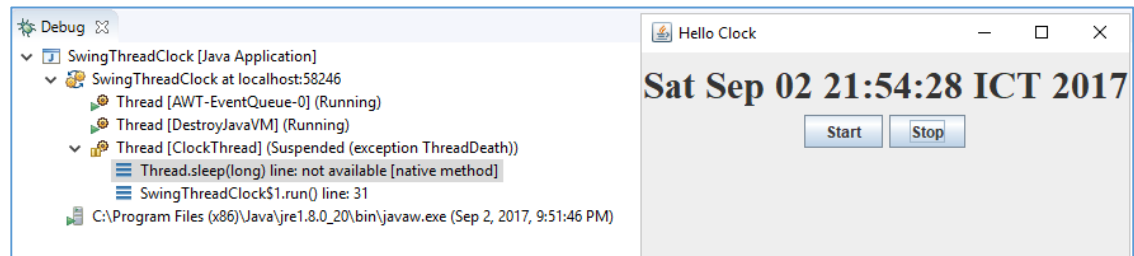
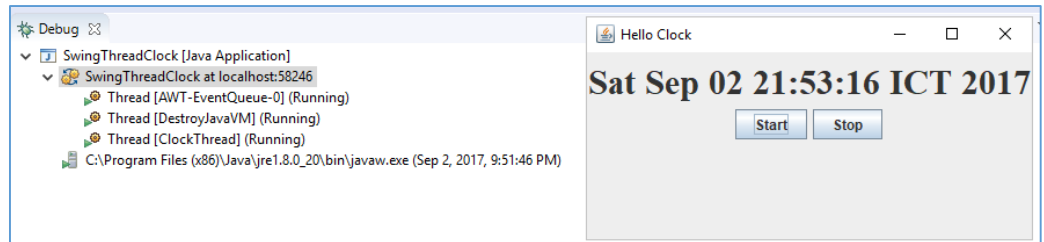
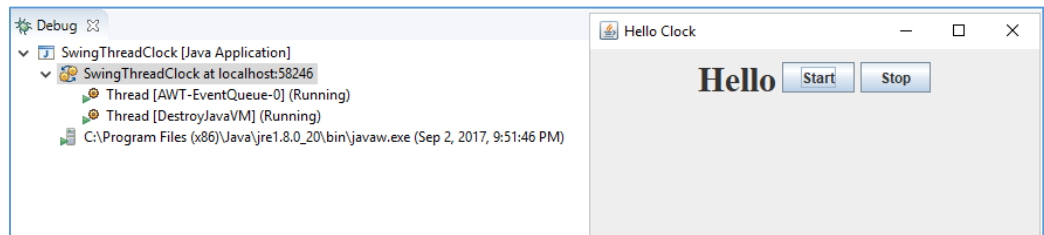
```
public class SwingThreadClock {
    public static void main(String[] args) {
        JFrame F=new JFrame("Hello Clock");
        final JLabel l=new JLabel("Hello");
        F.setBounds(1, 1, 400, 200);
        F.add(l);
        F.show();
        l.setFont(new Font("Serif", Font.BOLD, 30));
        Thread clock=new Thread("ClockThread"){
            @Override
            public void run() {
                while(true)
                {
                    l.setText(Calendar.getInstance().getTime().toString());
                    try {
                        Thread.sleep(1000);
                    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                }
            }
        };
        clock.start();
    }
}
```



Thread: Thread and Swing

```
public class SwingThreadClock {
    public static void main(String[] args) {
        JFrame F=new JFrame("Hello Clock");
        JPanel P =new JPanel();
        F.add(P);
        final JLabel l=new JLabel("Hello");
        JButton b1=new JButton("Start");
        JButton b2=new JButton("Stop");
        F.setBounds(1, 1, 400, 200);
        P.add(l);P.add(b1);P.add(b2);
        F.show();
        l.setFont(new Font("Serif", Font.BOLD, 30));
        Thread clock=new Thread("ClockThread"){
            @Override
            public void run() {
                while(true)
                {
                    l.setText(Calendar.getInstance().getTime().toString());
                    try {
                        Thread.sleep(1000);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        };
        b1.addMouseListener(new MouseAdapter(){
            @Override
            public void mouseClicked(MouseEvent arg0) {
                clock.start();
            }
        });

        b2.addMouseListener(new MouseAdapter(){
            @Override
            public void mouseClicked(MouseEvent arg0) {
                clock.stop();
            }
        });
    }
}
```



Console Tasks

SwingThreadClock [Java Application] C:\Program Files (x86)\Java\jre1.8.0_20\bin\javaw.exe (Sep 2, 2017, 9:51:46 PM)

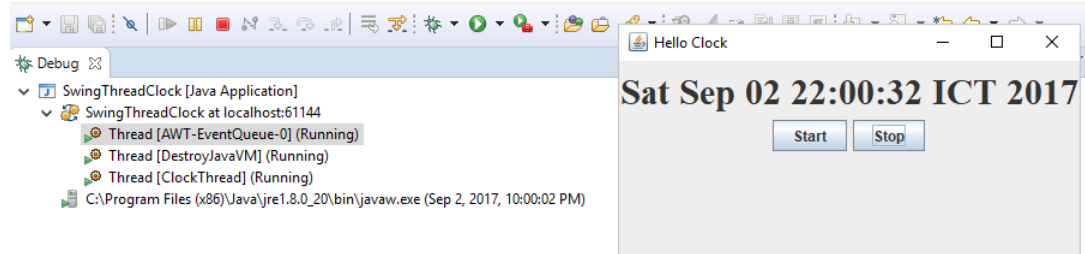
Exception in thread "AWT-EventQueue-0" java.lang.IllegalThreadStateException

```
at java.lang.Thread.start(Unknown Source)
at SwingThreadClock$2.mouseClicked(SwingThreadClock.java:40)
at java.awt.AWTEventMulticaster.mouseClicked(Unknown Source)
at java.awt.Component.processMouseEvent(Unknown Source)
at javax.swing.JComponent.processMouseEvent(Unknown Source)
at java.awt.Component.processEvent(Unknown Source)
at java.awt.Container.processEvent(Unknown Source)
at java.awt.Component.dispatchEventImpl(Unknown Source)
```

Thread: Thread and Swing

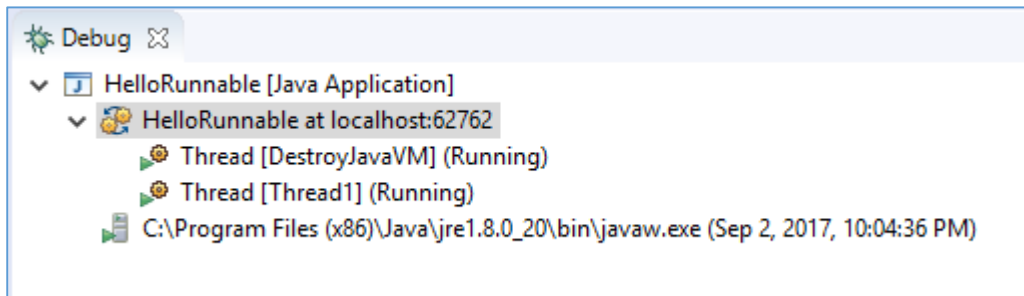
```
public class SwingThreadClock {
    public static void main(String[] args) {
        JFrame F=new JFrame("Hello Clock");
        JPanel P =new JPanel();
        F.add(P);
        final JLabel l=new JLabel("Hello");
        JButton b1=new JButton("Start");
        JButton b2=new JButton("Stop");
        F.setBounds(1, 1, 400, 200);
        P.add(l);P.add(b1);P.add(b2);
        F.show();
        l.setFont(new Font("Serif", Font.BOLD, 30));
        final Thread clock=new Thread("ClockThread"){
            @Override
            public void run() {
                while(true)
                {
                    l.setText(Calendar.getInstance().getTime().toString());
                    try {
                        Thread.sleep(1000);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        };
        clock.start();
        b1.addMouseListener(new MouseAdapter(){
            @Override
            public void mouseClicked(MouseEvent arg0) {
                clock.resume();
            }
        });

        b2.addMouseListener(new MouseAdapter(){
            @Override
            public void mouseClicked(MouseEvent arg0) {
                clock.suspend();
            }
        });
    }
}
```



Thread: Runnable

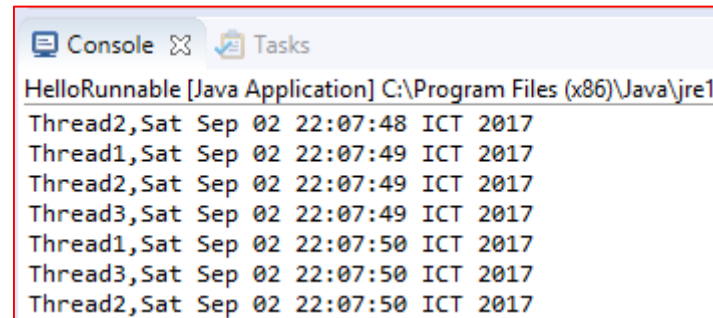
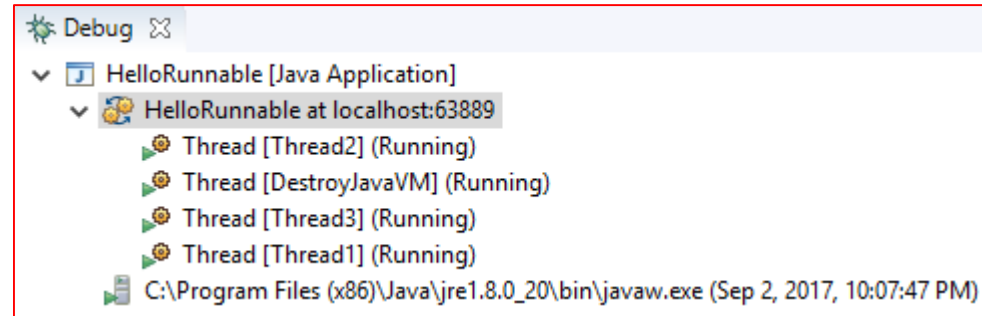
```
public class HelloRunnable {  
    public static void main(String[] args) {  
        Runnable task=new Runnable(){  
            @Override  
            public void run() {  
                while(true)  
                {  
                    System.out.println(Thread.currentThread().getName()+", "+  
                        Calendar.getInstance().getTime());  
                    try {  
                        Thread.sleep(1000);  
                    } catch (InterruptedException e) {  
                        e.printStackTrace();  
                    }  
                }  
            }  
        };  
        Thread t1=new Thread(task, "Thread1");  
        t1.start();  
    }  
}
```



```
HelloRunnable [Java Application] C:\Program Files (x86)\Java\jre1.8.  
Thread1,Sat Sep 02 22:04:36 ICT 2017  
Thread1,Sat Sep 02 22:04:37 ICT 2017  
Thread1,Sat Sep 02 22:04:38 ICT 2017  
Thread1,Sat Sep 02 22:04:39 ICT 2017  
Thread1,Sat Sep 02 22:04:40 ICT 2017  
Thread1,Sat Sep 02 22:04:41 ICT 2017  
Thread1,Sat Sep 02 22:04:42 ICT 2017  
Thread1,Sat Sep 02 22:04:43 ICT 2017
```

Thread: Runnable

```
public class HelloRunnable {
    public static void main(String[] args) {
        Runnable task=new Runnable(){
            @Override
            public void run() {
                while(true)
                {
                    System.out.println(Thread.currentThread().getName()+" "+
                    Calendar.getInstance().getTime());
                    try {
                        Thread.sleep(1000);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        };
        Thread t1=new Thread(task,"Thread1");
        Thread t2=new Thread(task,"Thread2");
        Thread t3=new Thread(task,"Thread3");
        t1.start();t2.start();t3.start();
    }
}
```



Thread ไหนจะเริ่มทำงานก่อน ?

ตอบ: ไม่รู้ ตัวไหนโหดเสร็จก่อนจะถูกรันก่อน ขึ้นอยู่กับระบบปฏิบัติการ
แล้วถ้า Thread3 จะทำงานได้ ก็ต่อเมื่อ Thread2 เริ่มทำงานแล้วจะทำอย่างไร ?

ตอบ: ให้เขียนโปรแกรมเช็คใน Thread3 ว่า Thread2 เริ่มทำงานหรือยัง ถ้ายังก็ให้รอ

Thread: Runnable and Swing

```
public class SwingRunnableClock {
public static void main(String[] args) {
JFrame F=new JFrame("Hello Clock");
JPanel P =new JPanel();
F.add(P);
final JLabel l=new JLabel("Hello");
JButton b1=new JButton("Start");
JButton b2=new JButton("Stop");
F.setBounds(1, 1, 400, 200);
P.add(l);P.add(b1);P.add(b2);
F.show();
l.setFont(new Font("Serif", Font.BOLD, 30));
Runnable task=new Runnable(){
@Override
public void run() {
while(true)
{
l.setText(Calendar.getInstance().getTime().toString());
try {
Thread.sleep(1000);
} catch (InterruptedException e) {
e.printStackTrace();
}}};
Thread clock=new Thread(task,"ClockThread");
clock.start();
b1.addMouseListener(new MouseAdapter(){
@Override
public void mouseClicked(MouseEvent arg0) {
clock.resume();
}});

b2.addMouseListener(new MouseAdapter(){
@Override
public void mouseClicked(MouseEvent arg0) {
clock.suspend();
}}); }
}
```



ถึงแม้ว่า Runnable interface จะต้องใช้ Thread class ช่วยในการทำงาน
แต่จำนวน Thread ที่เกิดขึ้นในระบบ ก็ไม่ได้แตกต่างกัน

ต้องการรัน Task ละ 1 Thread ควรใช้ Thread Class
ต้องการรัน Task หลาย Thread ควรใช้ Runnable Interface