Lecture 09

Object-Oriented Programming

{OOP} Thread

Prepared by **Hok Tin**

Content

- Thread

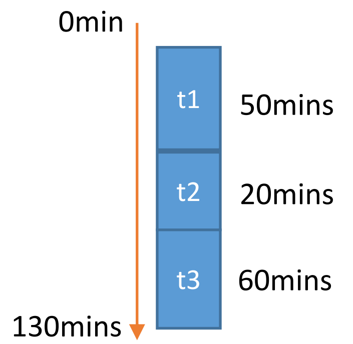
Java Thread

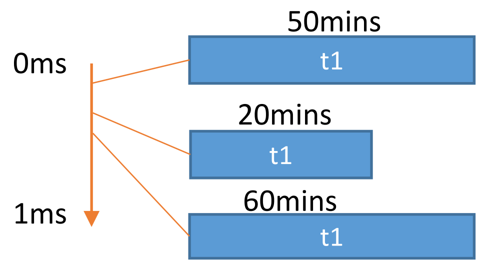
❑ Process

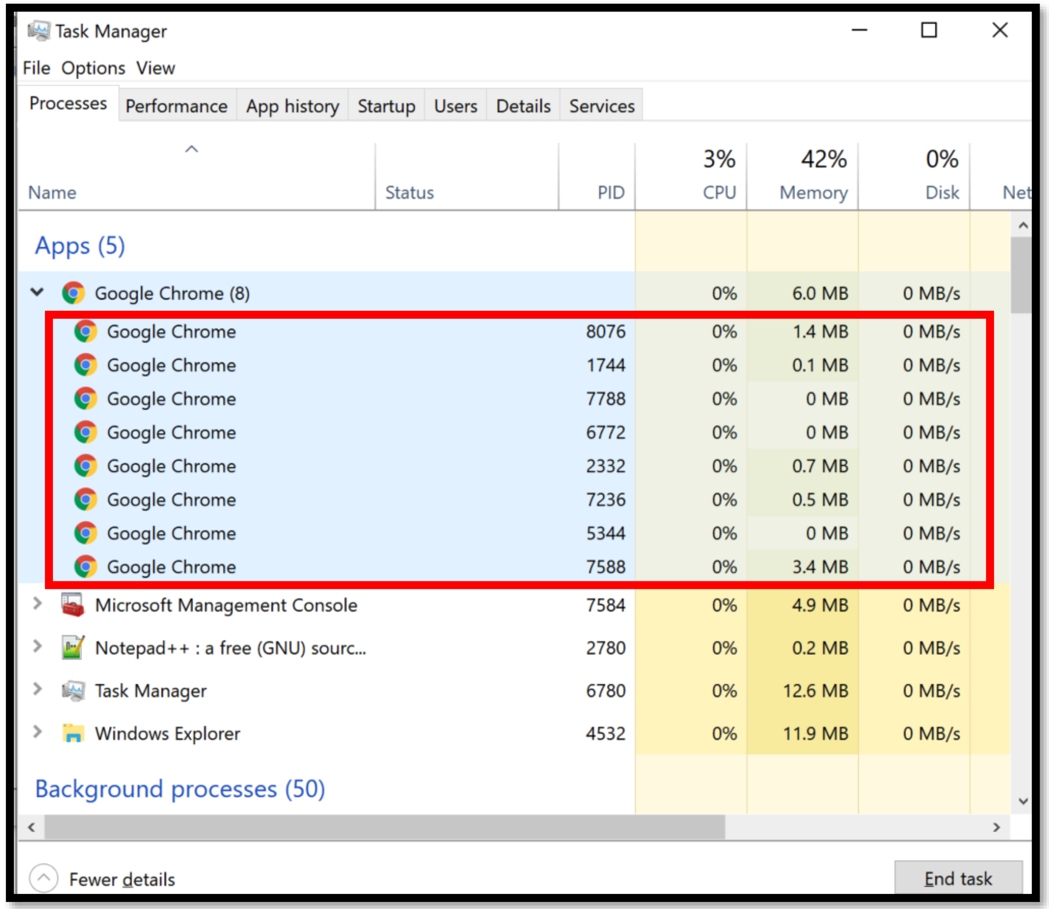
- A process is a self-contained execution environment, and it can be seen as a program or application
- A program itself contains multiple processes inside it

❑ Thread

- Thread can be called lightweight process
- Thread requires less resources to create and exists in the process, thread shares the process resources







Name	Status	PID	3% CPU	42% Memory	0% Disk	Net
Apps (5)						
Google Chrome (8)			0%	6.0 MB	0 MB/s	
Google Chrome		8076	0%	1.4 MB	0 MB/s	
Google Chrome		1744	0%	0.1 MB	0 MB/s	
Google Chrome		7788	0%	0 MB	0 MB/s	
Google Chrome		6772	0%	0 MB	0 MB/s	
Google Chrome		2332	0%	0.7 MB	0 MB/s	
Google Chrome		7236	0%	0.5 MB	0 MB/s	
Google Chrome		5344	0%	0 MB	0 MB/s	
Google Chrome		7588	0%	3.4 MB	0 MB/s	
Microsoft Management Console		7584	0%	4.9 MB	0 MB/s	
Notepad++ : a free (GNU) sourc...		2780	0%	0.2 MB	0 MB/s	
Task Manager		6780	0%	12.6 MB	0 MB/s	
Windows Explorer		4532	0%	11.9 MB	0 MB/s	
Background processes (50)						

Java provides two ways to create a thread programmatically.

1. Implementing the **java.lang.Runnable** interface
2. Extending the **java.lang.Thread** class

Java Thread

❑ Example: Thread.sleep

```
public class ThreadDemo2 {

    public static void main(String[] args) {
        long start = System.currentTimeMillis();
        System.out.println("Before sleep, ms = " + (System.currentTimeMillis() - start));
        try {

            Thread.sleep(2000);
            System.out.println("Sleep time in ms = " + (System.currentTimeMillis() - start));

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

}
```

```
public class ThreadDemo2 {
    public static void main(String[] args) throws InterruptedException {
        long start = System.currentTimeMillis();

        Thread.sleep(2000);

        System.out.println("Sleep time in ms = " + (System.currentTimeMillis() - start));
    }
}
```

Output:

```
Before sleep, ms = 0
Sleep time in ms = 2008
```

Java Thread

❑ Example: Implementing Runnable Interface

```java
class TaskRunnable implements Runnable {
    @Override
    public void run() {
        doDBProcessing();
    }

    private void doDBProcessing() {
        System.out.println("Heavy task processing - START " + Thread.currentThread().getName());
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Heavy task processing - END " + Thread.currentThread().getName());
    }
}
```

```java
public class ThreadDemo {
    public static void main(String[] args) {
        System.out.println("Main program - START");
        TaskRunnable task = new TaskRunnable();
        Thread taskProcess = new Thread(task, "t1");
        taskProcess.start();
        System.out.println("Main program - End");
    }
}
```

Output:

```
Main program - START
Main program - End
Heavy task processing - START t1
Heavy task processing - END t1
```

Java Thread

❑ Example: Implementing Thread class

```java
class TaskThread extends Thread {
    public TaskThread(String name) {
        super(name);
    }

    @Override
    public void run() {
        doDBProcessing();
    }

    private void doDBProcessing() {
        System.out.println("Heavy task processing - START " + Thread.currentThread().getName());
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Heavy task processing - END " + Thread.currentThread().getName());

    }
}
```

```java
public class ThreadDemo1 {
    public static void main(String[] args) {
        System.out.println("Main program - START");
        Thread taskProcess = new TaskThread("t1");
        taskProcess.start();
        System.out.println("Main program - End");
    }
}
```

Output:

```
Main program - START
Main program - End
Heavy task processing - START t1
Heavy task processing - END t1
```

Java Thread

❑ Example: Java Thread join

```java
class MyRunnable implements Runnable{
    @Override
    public void run() {
        System.out.println("Thread started:::"+Thread.currentThread().getName());
        try {
            Thread.sleep(4000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Thread ended:::"+Thread.currentThread().getName());
    }

}
```

Output:

```
Thread started:::t1
Thread started:::t2
Thread ended:::t1
Thread started:::t3
Thread ended:::t2
Thread ended:::t3
All threads are dead, exiting main
thread
```

```java
public class ThreadDemo3 {

    public static void main(String[] args) {
        Thread t1 = new Thread(new MyRunnable(), "t1");
        Thread t2 = new Thread(new MyRunnable(), "t2");
        Thread t3 = new Thread(new MyRunnable(), "t3");

        t1.start();

        //start second thread after waiting for 2 seconds or if it's dead
        try {
            t1.join(2000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        t2.start();

        //start third thread only when first thread is dead
        try {
            t1.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        t3.start();

        //let all threads finish execution before finishing main thread
        try {
            t1.join();
            t2.join();
            t3.join();
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        System.out.println("All threads are dead, exiting main thread");
    }
}
```

Good luck 👌

References:

https://www.digitalocean.com/community/tutorials/multithreading-in-java