JAVA THREAD 2

030523313 - Network programming

Asst. Prof. Dr. Choopan Rattanapoka

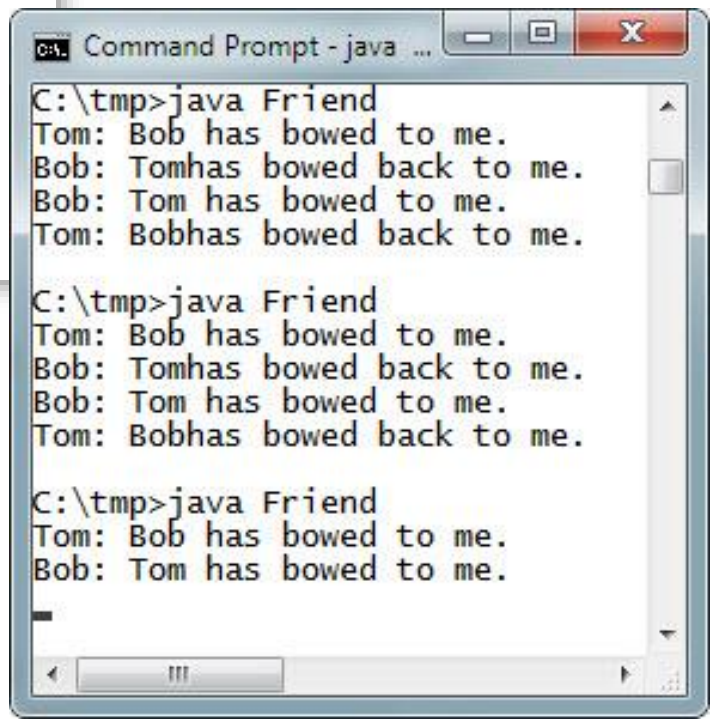
DeadLock (1)

- Deadlock ที่สถานการณ์ที่เมื่อมี thread จำนวนตั้งแต่ 2 ตัวขึ้นไป ถูก block ตลอดเวลาเพื่อรอกันเอง

```
public class Friend {
    private String name;

    public Friend(String name) {
        this.name = name;
    }

    public String getName() {
        return this.name;
    }

    public synchronized void bow(Friend bower) {
        System.out.println(name + ": " + bower.getName() + " has bowed to me.");
        bower.bowBack(this);
    }
    public synchronized void bowBack(Friend bower) {
        System.out.println(name + ": " + bower.getName() + "has bowed back to me.");
    }
    public void run() {

    }
```

Deadlock (2)

```
    public static void main(String[] args) {
        final Friend tom = new Friend("Tom");
        final Friend bob = new Friend("Bob");
        new Thread(new Runnable() {
            public void run() { tom.bow(bob);}
        }).start();
        new Thread(new Runnable() {
            public void run() { bob.bow(tom);}
        }).start();
    }
}
```



Producer-Consumer Problem

- ปัญหา **producer-consumer problem** คือ
  - ผู้ผลิต (producer) ผลิตสินค้าแล้วนำไปเก็บในโกดังสินค้า (warehouse)
  - ผู้บริโภค (consumer) จะนำสินค้าออกมาจากโกดัง
  - Warehouse จะเก็บสินค้าได้แค่ 1 ชิ้น
  - Producer จะต้องรอ ไม่ผลิตของถ้าโกดังเต็ม
  - Consumer จะต้องรอ ไม่นำสินค้าออกจากโกดังถ้าโกดังว่าง
- เวลาในการผลิตหรือนำสินค้าออกจะสุ่มระหว่าง 0 – 999 ms

Class : Warehouse (v.1)

```
public class Warehouse {
    volatile int productID;
    volatile boolean empty = true;

    public synchronized void put(int productID) {
        while (!empty) { }
        empty = false;
        this.productID = productID;
    }

    public synchronized int take() {
        while (empty) { }
        int result = this.productID;
        empty = true;
        return result;
    }
}
```

Class: Producer (v.1)

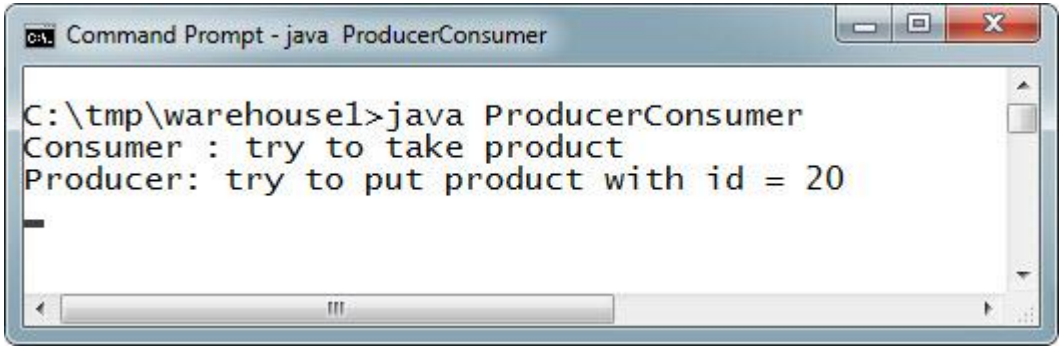
```
import java.util.*;

public class Producer extends Thread {
    Warehouse w;

    public Producer(Warehouse w) {
        this.w = w;
    }

    public void run() {
        Random r = new Random();
        for(int i = 0; i < 10; i++) {
            int id = r.nextInt(100);
            System.out.println("Producer: try to put product with id = " + id);
            w.put(id);
            System.out.println("Producer: put product with id = " + id);
            try {
                Thread.sleep(r.nextInt(1000));
            } catch(Exception e) {}
        }
    }
}
```

Class: Consumer (v.1)

```
import java.util.*;

public class Consumer extends Thread {
    Warehouse w;

    public Consumer(Warehouse w) {
        this.w = w;
    }

    public void run() {
        Random r = new Random();
        for(int i = 0; i < 10; i++) {
            System.out.println("Consumer : try to take product");
            int id = w.take();
            System.out.println("Consumer : take product with id = " + id);
            try {
                Thread.sleep(r.nextInt(1000));
            } catch(Exception e){}
        }
    }
}
```

Class : ProducerConsumer (main)

```
public class ProducerConsumer {
    public static void main(String[] args) {
        Warehouse w = new Warehouse();
        Producer p  = new Producer(w);
        Consumer c  = new Consumer(w);
        p.start(); c.start();
    }
}
```



Deadlock...??!!

เขียนผิดตรงไหนเนี่ย ??!

แก้: Warehouse (v.2)

```
public class Warehouse {
    volatile int productID;
    volatile boolean empty = true;

    public synchronized boolean put(int productID) {
        if (!empty) return false;
        empty = false;
        this.productID = productID;
        return true;
    }

    public synchronized int take() {
        if (empty) return -1;
        int result = this.productID;
        empty = true;
        return result;
    }
}
```

แก้: Producer (v.2)

```
import java.util.*;

public class Producer extends Thread {
    Warehouse w;

    public Producer(Warehouse w) {
        this.w = w;
    }

    public void run() {
        Random r = new Random();
        for(int i = 0; i < 10; i++) {
            int id = r.nextInt(100);
            System.out.println("Producer: try to put product with id = " + id);
            while(!w.put(id));
            System.out.println("Producer: put product with id = " + id);
            try {
                Thread.sleep(r.nextInt(1000));
            } catch(Exception e) {}
        }
    }
}
```

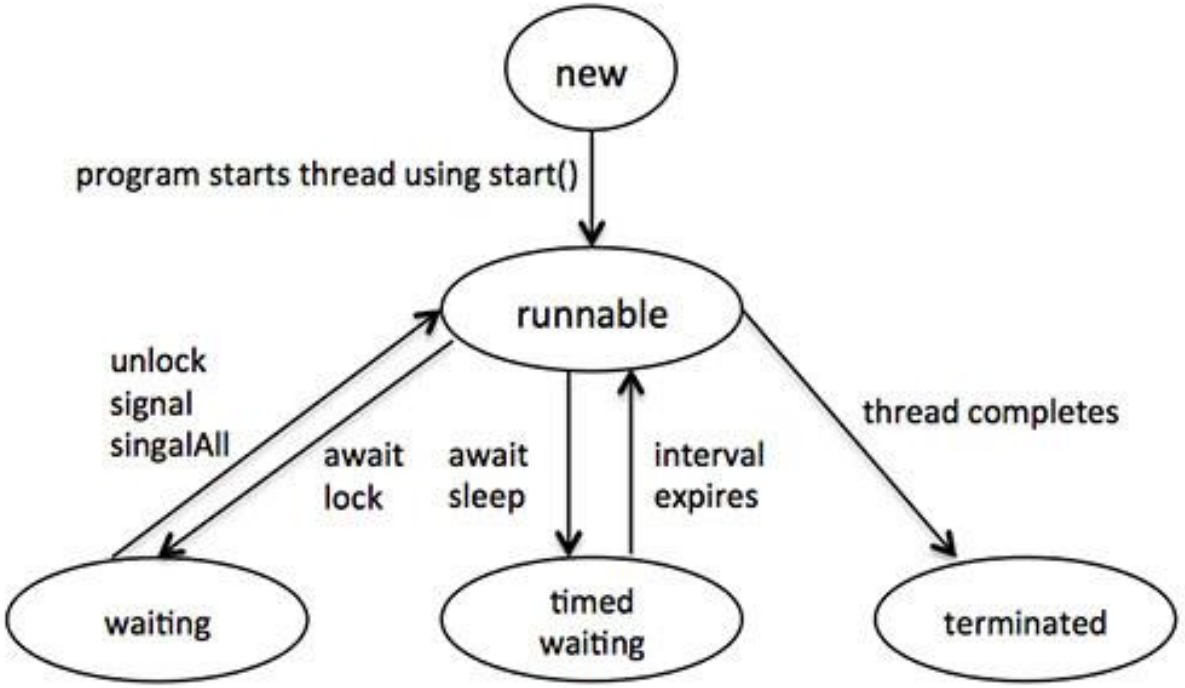
แก้: Consumer (v.2)

```
import java.util.*;

public class Consumer extends Thread {
    Warehouse w;

    public Consumer(Warehouse w) {
        this.w = w;
    }

    public void run() {
        int id;
        Random r = new Random();
        for(int i = 0; i < 10; i++) {
            System.out.println("Consumer : try to take product");
            while((id = w.take()) == -1);
            System.out.println("Consumer : take product with id = " + id);
            try {
                Thread.sleep(r.nextInt(1000));
            } catch(Exception e){}
        }
    }
}
```

คิดยังไงกับโปรแกรมนี้ ??

Wait/Notify/NotifyAll

- Thread เมื่อจำเป็นจะต้องรอข้อมูลอะไรที่อาจจะกินเวลา สามารถเรียกใช้งานเมธอด **wait()** เพื่อให้หยุดรอจนกว่าจะมีการแจ้งจาก Thread อื่น
- **notify()** เป็นเมธอดสำหรับส่งสัญญาณให้ Thread 1 ตัว เพื่อให้ตื่นจากการ wait()
- **notifyAll()** เป็นเมธอดสำหรับส่งสัญญาณให้ทุก Thread ที่อยู่ในสถานะ wait ให้ตื่นขึ้นเพื่อทำงาน



แก้: Warehouse (v.3)

```
public class Warehouse {
    volatile int productID;
    volatile boolean empty = true;

    public synchronized void put(int productID) {
        if(!empty) {
            try {
                wait();
            } catch(Exception e) {}
        }
        this.productID = productID;
        empty = false;
        notify();
    }

    public synchronized int take() {
        if(empty) {
            try {
                wait();
            } catch(Exception e) {}
        }
        int result = this.productID;
        empty = true;
        notify();
        return result;
    }
}
```

ใช้ Class Producer และ Consumer ของ v.1

ปัญหา Classic ☺

- Dining Philosophers
  - มีนักปรัชญาอยู่ 5 คน นั่งบนโต๊ะกลม
  - มีส้อมอยู่ทั้งหมด 5 อัน
  - นักปรัชญาจะมีวงการทำงานคือ
    - คิด (think)
    - หิว (hungry)
    - กิน (eat)
  - การที่นักปรัชญาจะกินได้ จะต้องหยิบส้อมด้านซ้าย และด้านขวาของตัวเองเท่านั้น และต้องหยิบได้ทั้ง 2 อันถึงจะสามารถกินได้
  - **ตัวช่วย :** ถ้าต้องการกินแล้วไม่ได้ส้อมทั้ง 2 อันให้กลับไปคิดต่อ