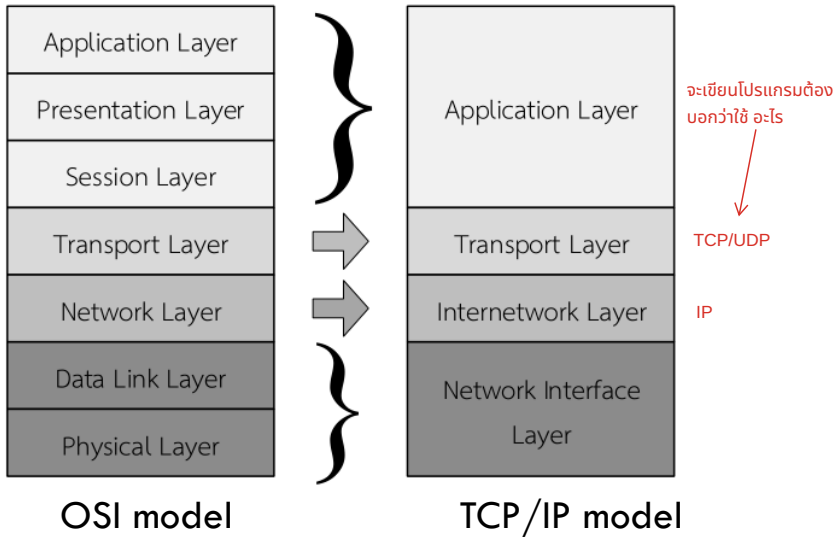
OSI Model

OSI (Open Source Interconnection) 7 Layer Model

Layer	Application/Example		Central Device/ Protocols		
Application (7) Serves as the window for users and application processes to access the network services.	**End User layer** Program that opens what was sent or creates what is to be sent Resource sharing • Remote file access • Remote printer access • Directory services • Network management		**User Applications** SMTP		GATEWAY
Presentation (6) Formats the data to be presented to the Application layer. It can be viewed as the "Translator" for the network.	**Syntax layer** encrypt & decrypt (if needed) Character code translation • Data conversion • Data compression • Data encryption • **Character Set Translation**		JPEG/ASCII EBDIC/TIFF/GIF PICT		
Session (5) Allows session establishment between processes running on different stations.	**Synch & send to ports** (logical ports) Session establishment, maintenance and termination • Session support - perform security, name recognition, logging, etc.		**Logical Ports** RPC/SQL/NFS NetBIOS names		
Transport (4) Ensures that messages are delivered error-free, in sequence, and with no losses or duplications.	**TCP** Host to Host, Flow Control Message segmentation • Message acknowledgement • Message traffic control • Session multiplexing	PACKET FILTERING	TCP/SPX/UDP		
Network (3) Controls the operations of the subnet, deciding which physical path the data takes.	**Packets** ("letter", contains IP address) Routing • Subnet traffic control • Frame fragmentation • Logical-physical address mapping • Subnet usage accounting		**Routers** IP/IPX/ICMP		Can be used on all layers
Data Link (2) Provides error-free transfer of data frames from one node to another over the Physical layer.	**Frames** ("envelopes", contains MAC address) [NIC card — Switch — NIC card] (end to end) Establishes & terminates the logical link between nodes • Frame traffic control • Frame sequencing • Frame acknowledgment • Frame delimiting • Frame error checking • Media access control		**Switch** **Bridge** **WAP** PPP/SLIP	Land Based Layers	
Physical (1) Concerned with the transmission and reception of the unstructured raw bit stream over the physical medium.	**Physical structure** Cables, hubs, etc. Data Encoding • Physical medium attachment • Transmission technique - Baseband or Broadband • Physical medium transmission Bits & Volts		**Hub**		

TCP/IP Model



Arguments

```
1 public class TestJava
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("Number of argument : " + args.length);
6         for(int i = 0; i < args.length; i++)
7             System.out.println("Args[" + i + "] = " + args[i]);
8     }
9 }
```

```
C:\WINDOWS\system32\cmd.exe
C:\>javac TestJava.java

C:\>java TestJava
Number of argument : 0

C:\>java TestJava Hello World 123
Number of argument : 3
Args[0] = Hello
Args[1] = World
Args[2] = 123
```

0 1 "2 3 AB C"

Type Conversion

- Arguments received from a command line are in the String format.
- So, if we want to use them as numbers, we need to convert them. We can use the static class below:
  - **Integer.parseInt**(String intValue)
  - **Float.parseFloat**(String floatValue)
  - **Double.parseDouble**(String doubleValue)

Example: Type Conversion

```java
import java.io.*;

public class TypeConversion {
        public static void main(String[] args) {
                String num1 = "1";
                int num2 = 2;
                System.out.println("Result1 = " + (num1 + num2));  12
                System.out.println("Result2 = " + (Integer.parseInt(num1)+num2));  3

        }
}
```

Quiz

```java
import java.io.*;

public class Exo1 {
        public static void main(String[] args) {
                if(args.length != 2) {
                        System.out.println("Please enter 2 arguments");
                        System.exit(1);
                }
                int num1 = Integer.parseInt(args[0]);
                int num2 = Integer.parseInt(args[1]);
                System.out.println("Result = " + (num1 + num2));

        }
}
```

- Find the output of this program when user runs it with the following commands:
  - java Exo1 please enter.....
  - java Exo1 125 please enter.....
  - java Exo1 25 15 Result = 40
  - java Exo1 25 a Exeption

Example: Class IOException

- java.lang.Exception คลาสใหญ่สุด
  - java.io.IOException
    - java.io.CharConversionException
    - java.io.EOFException
    - java.io.FileNotFoundException
    - java.io.InterruptedIOException
    - java.io.ObjectStreamException
      - java.io.InvalidClassException
      - java.io.InvalidObjectException
      - java.io.NotActiveException
      - java.io.NotSerializableException
      - java.io.OptionalDataException
      - java.io.StreamCorruptedException
      - java.io.WriteAbortedException
    - java.io.SyncFailedException
    - java.io.UnsupportedEncodingException
    - java.io.UTFDataFormatException

Example: Class Exception

- class java.lang.Exception
  - class java.lang.ClassNotFoundException
  - class java.lang.CloneNotSupportedException
  - class java.lang.IllegalAccessException
  - class java.lang.InstantiationException
  - class java.lang.InterruptedException
  - class java.lang.NoSuchFieldException
  - class java.lang.NoSuchMethodException
  - class java.lang.RuntimeException
    - class java.lang.ArithmeticException
    - class java.lang.ArrayStoreException
    - class java.lang.ClassCastException
    - class java.lang.IllegalArgumentException
      - class java.lang.IllegalThreadStateException
      - class java.lang.NumberFormatException
    - class java.lang.IllegalMonitorStateException
    - class java.lang.IllegalStateException
    - class java.lang.IndexOutOfBoundsException
      - class java.lang.ArrayIndexOutOfBoundsException
      - class java.lang.StringIndexOutOfBoundsException
    - class java.lang.NegativeArraySizeException
    - class java.lang.NullPointerException
    - class java.lang.SecurityException
    - class java.lang.UnsupportedOperationException

Fixed the problem of "ArrayIndexOutofBoundsException"

```
import java.io.*;

public class Exo1 {
        public static void main(String[] args) {
                try {
                        int num = Integer.parseInt(args[0]);
                        System.out.println("Result = " + num);
                } catch (ArrayIndexOutOfBoundsException e) {
                        System.out.println("Usage : java Exo1 <number>");
                }
        }
}
```

- In your opinion, what would be the output of the following command:
  - java Exo1 Hello

Fixed the problem of "NumberFormatException"

```
import java.io.*;

public class Exo1 {
        public static void main(String[] args) {
                try {
                        int num = Integer.parseInt(args[0]);
                        System.out.println("Result = " + num);
                } catch (ArrayIndexOutOfBoundsException e) {
                        System.out.println("Usage : java Exo1 <number>");
                } catch (NumberFormatException ee) {
                        System.out.println("Usage : java Exo1 <number>");
                }
        }
}
```

Make it easier to catch "Exception"

- The Exception class is the parent class of:
  - Class NumberFormatException
  - Class ArrayIndexOutOfBoundsException
- So, we can catch all exceptions by **catching the Exception class.**

```
import java.io.*;

public class Exo1 {
        public static void main(String[] args) {
                try {
                        int num = Integer.parseInt(args[0]);
                        System.out.println("Result = " + num);
                } catch (Exception e) {
                        System.out.println("Usage : java Exo1 <number>");
                }
        }
}
```

OutputStream

- The basic class for data transmission is

  java.io.OutputStream

- It has important methods: a[] = a b c d e
  - public abstract void **write(int b)** throws IOException
  - write(a) -> a b c d e
    public void **write(byte[ ] data)** throws IOException
  - write(a,1,2) -> b c จุดเริ่มต้น
    public void **write(byte[ ] data, int offset, int length)** throws IOException
  - ส่งทีเดียว
    public void **flush( )** throws IOException
  - flush อัตโนมัติ
    public void **close( )** throws IOException

InputStream

- The basic class for data reception is

  java.io.InputStream

- It has important methods:

  b[] = _ _ _ _ _ abcdefg

  อ่านตามขนาดของ array
  - public abstract int **read( )** throws IOException

  read(b) => a b c d e
  - public int **read(byte[ ] input)** throws IOException

  read(b,1,2) => _ a b _ _
  - public int **read(byte[ ] input, int offset, int length)** throws IOException
  - public long **skip(long n)** throws IOException
  - public void **available( )** throws IOException
  - public void **close( )** throws IOException

Java and data file

- Managing files in Java can be done in various ways.
- The simplest method is to use the **File class.**
- **Example:**
  - File f = new File(String *filename*);
  - File f = new File(String *pathname*, String *filename*);
  - File f = new File(File *pathname*, String *filename*);

Java File Methods

- Details about each method of java **File** class can be found in the JavaDoc manual.
- The important methods are:
  - boolean **delete**(); delete a file/directory.
  - boolean **exists**(); check if a file/directory exists
  - boolean **isDirectory**(); check if it is a directory
  - boolean **isFile**(); check if it is a file
  - long **length**(); get the size of a file/directory
  - File[] **listFiles**(); list file/directory name in that directory in an array of File type.
  - String[] **list**(); list file/directory name in that directory in an array of String type.
  - String **getName**(); get only file/directory name (removed path)

Example 1

```java
import java.io.*;

public class Example1 {
    public static void main(String[] args) {
        File f  = new File("myFile.txt");
        if(!f.exists()) {
            System.out.println("File does not exist");
            System.exit(1);
        }

        if(f.isFile()) {
            System.out.println("myFile.txt is a File");
            System.out.println("File size = " + f.length());
        } else if(f.isDirectory()) {
            System.out.println("myFile.txt is a directory");
        } else {
            System.out.println("....");
        }
    }
}
```

Example 2

```java
import java.io.*;

public class Example2 {
    public static void main(String[] args) {
        File f = new File("C:\\AppServ");
        File list[] = f.listFiles();

        for(int i = 0; i < list.length; i++) {
            if(list[i].isFile()) {
                System.out.println(list[i].getName() + " is a file");
            } else if (list[i].isDirectory()) {
                System.out.println(list[i].getName() + " is a directory");
            } else {
                System.out.println(list[i].getName() + " is unknown");
            }
        }
    }
}
```

Reading data from a file.

```java
import java.io.*;

public class ReadFile {
        public static void main(String[] args) {
                try {
                        int n;
                        byte[] b = new byte[16];

                        File f = new File("myfile.txt");
                        FileInputStream fin = new FileInputStream(f);
                        while((n = fin.read(b)) > 0) {
                                String data = new String(b, 0, n);
                                System.out.print(data);
                        }

                        fin.close();
                } catch(Exception e) { e.printStackTrace(); }

        }
}
```

Writing data to a file.

```java
import java.io.*;

public class WriteFile {
        public static void main(String[] args) {
                try {
                        String msg = "Hello World";
                        File f = new File("myfile.txt");
                        FileOutputStream fout = new FileOutputStream(f);
                        byte[] b = msg.getBytes();
                        fout.write(b);
                        fout.close();
                } catch(Exception e) { e.printStackTrace(); }

        }
}
```

Example: Usage of PrintWriter

```java
import java.io.*;

public class PwTest {
        public static void main(String[] args) {
                try {
                        String msg = "Hello World";
                        File f = new File("myfile.txt");
                        FileOutputStream fout = new FileOutputStream(f);
                        PrintWriter pout = new PrintWriter(fout);
                        pout.print(msg);
                        pout.flush();
                        fout.close();
                } catch(Exception e) { e.printStackTrace(); }

        }
}
```

Example: Usage of BufferedReader (1)

```java
import java.io.*;

public class BrTest {
        public static void main(String[] args) {
                try {
                        String msg;
                        File f = new File("myfile.txt");
                        FileInputStream fin  = new FileInputStream(f);
                        InputStreamReader ir = new InputStreamReader(fin);
                        BufferedReader br    = new BufferedReader(ir);

                        while((msg = br.readLine()) != null)
                                System.out.println(msg);

                        fin.close();
                } catch(Exception e) { e.printStackTrace(); }

        }
}
```

Example: Usage of BufferedReader (2)

```java
import java.io.*;

public class BrTest2 {
        public static void main(String[] args) {
                try {
                        String msg;
                        File f = new File("myfile.txt");
                        BufferedReader br    = new BufferedReader(
                                                new InputStreamReader(
                                                new FileInputStream(f)));

                        while((msg = br.readLine()) != null)
                                System.out.println(msg);

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

        }
}
```

Example: Thread

```java
import java.io.*;

public class TwoThread extends Thread {
        public void run( ) {
                for(int i = 0; i < 10; i++) {
                        System.out.println("New Thread");
                }
        }

        public static void main(String[] args) {
                TwoThread tt = new TwoThread();
                tt.start();

                for(int i = 0; i < 10; i++) {
                        System.out.println("Main Thread");
                }
        }
}
```

ต้องมี

สรุปคือทำงานพร้อมกัน

ทำงานเลย ไม่รอให้ run เสร็จ

What do you think is the output of the program ?

Example: Sleep

```java
import java.io.*;

public class TestSleep {
        public static void main(String[] args) {
                System.out.println("Hello");

                try {
                        Thread.sleep(3000);
                } catch (Exception e) {}

                System.out.println("Bye Bye");
        }
}
```

Example: multiple threads with sleep

```java
import java.io.*;

public class MultiThread extends Thread {
        String myName;
        long sleepTime;

        public MultiThread(String myName, long sleepTime) {
                this.myName    = myName;
                this.sleepTime = sleepTime;
        }

        public void run() {
                for(int i = 0; i < 5; i++) {
                        System.out.println(myName);
                        try {
                                Thread.sleep(sleepTime);
                        } catch(Exception e) {}
                }
        }

        public static void main(String[] args) {
                MultiThread t1 = new MultiThread("-1-", 1000);
                MultiThread t2 = new MultiThread("-2-", 2000);
                MultiThread t3 = new MultiThread("-3-", 3000);

                t1.start();
                t2.start();
                t3.start();
        }
}
```

4 Thread
t1 t2 t3
main

t1	t2	t3
-1-	-2-	-3-
5s	10s	15s

โปรแกรมนี้ทำงาน 15 วิ

ถ้ามี 1 thread ทำงานอยู่โปรแกรมจะยังไม่จบ

ผลการรัน

```
-1-
-3-
-2-
-1-
-2-
-1-
-3-
-1-
-2-
-1-
-3-
-2-
-2-
-3-
-3-
```

Example: Thread

```java
import java.io.*;

public class TwoThread implements Runnable {
        public void run( ) {
                for(int i = 0; i < 10; i++) {
                        System.out.println("New Thread");
                }
        }

        public static void main(String[] args) {
                TwoThread tt = new TwoThread();
                Thread t = new Thread(tt);
                t.start();

                for(int i = 0; i < 10; i++) {
                        System.out.println("Main Thread");
                }
        }
}
```

* The differences of 2 methods

(1) Method: extends Thread

```java
import java.io.*;

public class TwoThread extends Thread {
        public void run( ) {
                for(int i = 0; i < 10; i++) {
                        System.out.println("New Thread");
                }
        }

        public static void main(String[] args) {
                TwoThread tt = new TwoThread();
                tt.start();

                for(int i = ; i < 10; i++) {
                        Sys m.out.println("Main Thr
                }
        }
}
```

Class modifier
(1) extends Thread
(2) implements Runnable

Creating and Invoking a Thread Object.

```java
import java.io.*;

public class TwoThread implements Runnable {
        public void run( ) {
                for(int i = 0; i < 10; i++) {
                        System.out.println("New Thread");
                }
        }

        public static void main(String[] args) {
                TwoThread tt = new TwoThread();
                Thread t = new Thread(tt);
                t.start();

                for(int i = 0; i < 10; i++) {
                        System.out.println("Main Thread");
                }
        }
}
```

(2) Method: implements Runnable

Example: summation program

```java
import java.io.*;

public class Sum {
        int from;
        int where;
        int result = 0;

        public Sum(int from, int where) {
                this.from  = from;
                this.where = where;
        }

        public void run() {
                for(int i = from; i <= where; i++) {
                        result += i;
                }
        }

        public int getResult() {
                return result;
        }


        public static void main(String[] args) {
                Sum s = new Sum(0, 1000000);
                s.run();
                System.out.println("Result = " + s.getResult());
        }
}
```

Thread issues

```java
import java.io.*;

public class SumThreadWrong implements Runnable {
        int from;
        int where;
        int result = 0;

        public SumThreadWrong(int from, int where) {
                this.from  = from;
                this.where = where;
        }

        public void run() {
                for(int i = from; i <= where; i++) {
                        result += i;
                }
        }

        public int getResult() {
                return result;
        }

        public static void main(String[] args) {
                int s = 0;
                SumThreadWrong s1 = new SumThreadWrong(0, 499999);
                SumThreadWrong s2 = new SumThreadWrong(500000, 1000000);
                Thread t1 = new Thread(s1);
                Thread t2 = new Thread(s2);
                try {
                        t1.start();  t2.start();
                        s = s1.getResult() + s2.getResult();
                } catch(Exception e){}
                System.out.println("Result = " + s);
        }

}
```

Problem about the addition operation between the output of Thread 1 and Thread 2

Usage: join( )

```java
import java.io.*;

public class SumThread implements Runnable {
        int from;
        int where;
        int result = 0;

        public SumThread(int from, int where) {
                this.from  = from;
                this.where = where;
        }

        public void run() {
                for(int i = from; i <= where; i++) {
                        result += i;
                }
        }

        public int getResult() {
                return result;
        }

        public static void main(String[] args) {
                int s = 0;
                SumThread s1 = new SumThread(0, 499999);
                SumThread s2 = new SumThread(500000, 1000000);
                Thread t1 = new Thread(s1);
                Thread t2 = new Thread(s2);

                try {
                        t1.start();  t2.start();
                        t1.join();  t2.join();
                        s = s1.getResult() + s2.getResult();
                } catch(Exception e){}
                System.out.println("Result = " + s);
        }

}
```

DeadLock (1)

- Deadlock is a situation when more than 2 threads interlocks.

```java
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)

```java
    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();
    }
}
```

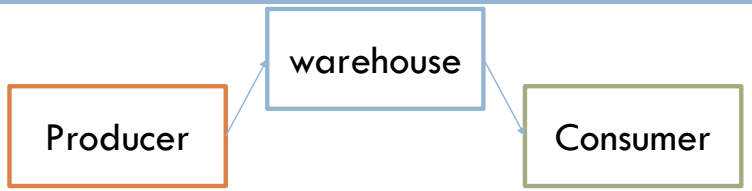
Command Prompt - java ...

```
C:\tmp>java Friend
Tom: Bob has bowed to me.
Bob: Tomhas bowed back to me.
Bob: Tom has bowed to me.
Tom: Bobhas bowed back to me.

C:\tmp>java Friend
Tom: Bob has bowed to me.
Bob: Tomhas bowed back to me.
Bob: Tom has bowed to me.
Tom: Bobhas bowed back to me.

C:\tmp>java Friend
Tom: Bob has bowed to me.
Bob: Tom has bowed to me.
```

Producer-Consumer Problem



- producer-consumer problem
  - Producer produces a product and stores it in a warehouse.
  - Consumer takes a product out of the warehouse.
  - Warehouse can store only 1 product.
  - Producer has to wait producing products if the warehouse is full.
  - Consumer has to wait taking products if the warehouse is empty.
- Time usage in producing or taking a product is a random number between 0 – 999 ms

Class : Warehouse (v.1)

```java
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;
    }
}
```

ไม่ควรอยู่ใน synchronized

Class: Producer (v.1)

```java
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) {}
        }
    }
}
```

แรนดอม 0-99

Class: Consumer (v.1)

```java
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)

```java
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();
    }
}
```

```
Command Prompt - java ProducerConsumer

C:\tmp\warehouse1>java ProducerConsumer
Consumer : try to take product
Producer: try to put product with id = 20
```

Deadlock…??!! Where is my wrong code ??!

Fixed: Warehouse (v.2)

```java
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;
    }
}
```

Fixed: Producer (v.2)

```java
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) {}
        }
    }
}
```

Fixed: Consumer (v.2)

```java
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){}
        }
    }
}
```

What do you think about this program??

Wait/Notify/NotifyAll

หยุดรอ เหมือนหรือ ไม่ได้ใช้พิษ

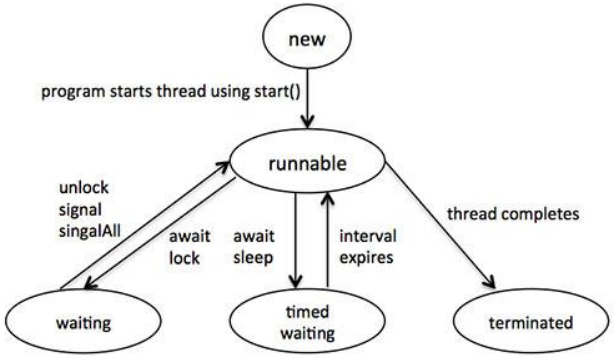
- When a thread needs to wait some data from another thread, it can invoke the method **wait()** to wait the notification from another thread.

เหมือนส่งไลน์ แต่สุ่ม ปลุกตัวเดียว

- **notify()** is a method to send the notication to 1 thread (random) to wake it up from **wait()**.

ปลุกหมดเลย

- **notifyAll()** is a method to send the notification to wake all of waiting threads up.



Fixed: Warehouse (v.3)

```java
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();  มีเพื่อ ปลุก wait()
    }

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

Use Class Producer and Consumer v.1

Example: LinkedList Class

```java
import java.util.LinkedList;

public class LinkedListTest {
    public static void main(String[] args) {
        LinkedList<String> myList = new LinkedList();
        String m;

        myList.offer("1"); myList.offer("2"); myList.offer("3");
        myList.offer("4"); myList.offer("5"); myList.offer("6");

        System.out.println(myList);

        m = myList.poll();
        System.out.println("Output = " + m);
        m = myList.poll();
        System.out.println("Output = " + m);

        System.out.println(myList);

    }
}
```

```
run:
[1, 2, 3, 4, 5, 6]
Output = 1
Output = 2
[3, 4, 5, 6]
```

Thread issue – 2 (1)

```java
public class Spawn implements Runnable {
    public void run() { }

    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        long startTime = System.currentTimeMillis();
        for(int i = 0; i < n; i++) {
            Spawn s = new Spawn();
            Thread t = new Thread(s);
            t.start();
            try {
                t.join();
            } catch(Exception e) { }
        }
        long stopTime = System.currentTimeMillis();
        System.out.println("Time usage : " + (stopTime - startTime) + " ms");
    }
}
```

จะได้เวลาว่าใช้ไปกี่millisec

Thread issue – 2 (2)

```java
public class NoSpawn  {

    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        long startTime = System.currentTimeMillis();
        for(int i = 0; i < n; i++) { }
        long stopTime = System.currentTimeMillis();
        System.out.println("Time usage : " + (stopTime - startTime) + " ms");
    }
}
```

Command Prompt

```
C:\tmp>java NoSpawn 1
Time usage : 0 ms

C:\tmp>java NoSpawn 10
Time usage : 0 ms

C:\tmp>java NoSpawn 100
Time usage : 0 ms

C:\tmp>java NoSpawn 1000
Time usage : 0 ms

C:\tmp>java NoSpawn 10000
Time usage : 1 ms

C:\tmp>java NoSpawn 100000
Time usage : 1 ms
```

Command Prompt

```
C:\tmp>java Spawn 1
Time usage : 0 ms

C:\tmp>java Spawn 10
Time usage : 2 ms

C:\tmp>java Spawn 100
Time usage : 17 ms

C:\tmp>java Spawn 1000
Time usage : 135 ms

C:\tmp>java Spawn 10000
Time usage : 1257 ms

C:\tmp>java Spawn 100000
Time usage : 12425 ms
```