Sinkronisasi

Mengapa menggunakan sinkronisasi threads?

- Bagaimanapun juga sebuah thread yang berjalan bersama-sama kadang-kadang membutuhkan resource atau method dari luar
- Butuh untuk berkomunikasi satu dengan yang lain sehingga dapat mengetahui status dan aktifitas mereka
- Contoh: Permasalahan Produser-Konsumer

```
1   class TwoStrings {
2     static void print(String str1, String str2) {
3       System.out.print(str1);
4       try {
5         Thread.sleep(500);
6       } catch (InterruptedException ie) {
7       }
8       System.out.println(str2);
9     }
10  }
11  class PrintStringsThread implements Runnable {
12    Thread thread;
13    String str1, str2;
14    PrintStringsThread(String str1, String str2) {
15      this.str1 = str1;
16      this.str2 = str2;
17      thread = new Thread(this);
18      thread.start();
19    }
20    public void run() {
21      TwoStrings.print(str1, str2);
22    }
23  }
24  class TestThread {
25    public static void main(String args[]) {
26      new PrintStringsThread("Hello ", "there.");
27      new PrintStringsThread("How are ", "you?");
28      new PrintStringsThread("Thank you ",
29                             "very much!");
30    }
31  }
```

- Contoh hasil:

```
Hello How are Thank you there.
you?
very much!
```

Mengunci sebuah object:

- Untuk memastikan bahwa hanya satu thread yang mendapatkan hak akses kedalam method tertentu

- Java memperbolehkan penguncian terhadap sebuah object termasuk method-method-nya dengan menggunakan monitor
  - Object tersebut akan menjalankan sebuah monitor implicit pada saat object dari method sinkronisasi dipanggil
  - Sekali object tersebut dimonitor, monitor tersebut akan memastikan bahwa tidak ada thread yang akan mengakses object yang sama

- Sinkronisasi sebuah method:
  - Menggunakan keyword *synchronized*
    - Dapat menjadi header dari pendefinisian method
    - Dapat mensinkronisasi object dimana method tersebut menjadi anggota dari synchronized (<object>) {
      //statements yang akan disinkronisasikan
      }

Contoh Synchronized Pertama

```
1   class TwoStrings {
2     synchronized static void print(String str1,
3                                    String str2) {
4       System.out.print(str1);
5       try {
6         Thread.sleep(500);
7       } catch (InterruptedException ie) {
8       }
9       System.out.println(str2);
10    }
11  }
12  class PrintStringsThread implements Runnable {
13    Thread thread;
14    String str1, str2;
15    PrintStringsThread(String str1, String str2) {
16      this.str1 = str1;
17      this.str2 = str2;
18      thread = new Thread(this);
19      thread.start();
20    }
21    public void run() {
22      TwoStrings.print(str1, str2);
23    }
24  }
25  class TestThread {
26    public static void main(String args[]) {
27      new PrintStringsThread("Hello ", "there.");
28      new PrintStringsThread("How are ", "you?");
29      new PrintStringsThread("Thank you ",
30                             "very much!");
31    }
32  }
```

Contoh Hasil:
Hello there.
How are you?
Thank you very much!

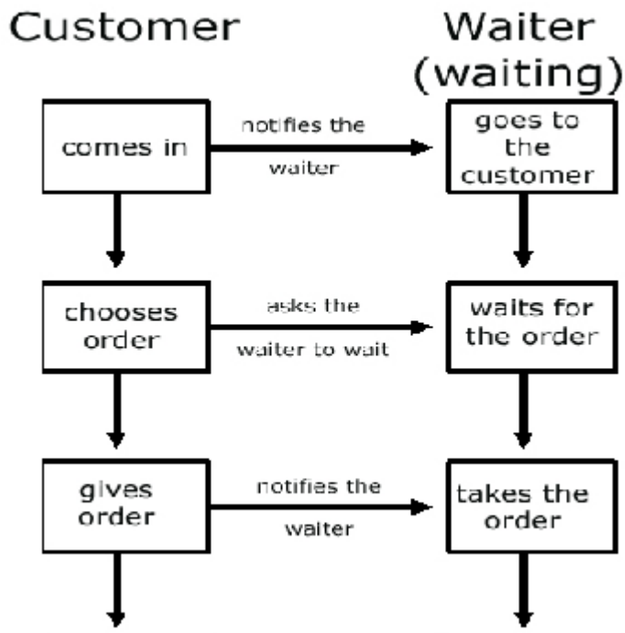
Contoh Synchronized Kedua

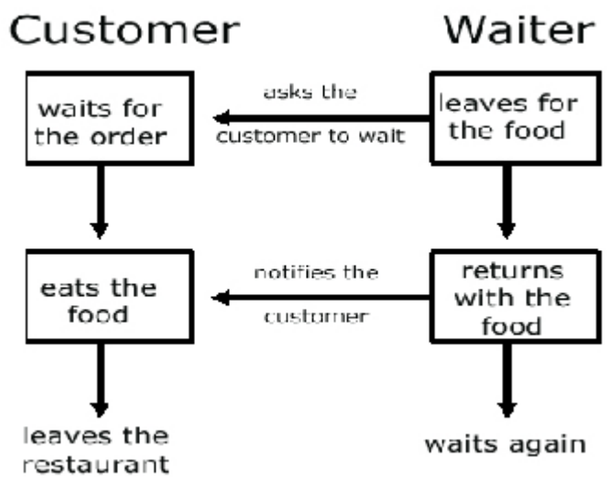
```
1   class TwoStrings {
2     static void print(String str1, String str2) {
3       System.out.print(str1);
4       try {
5         Thread.sleep(500);
6       } catch (InterruptedException ie) {
7       }
8       System.out.println(str2);
9     }
10  }
11  class PrintStringsThread implements Runnable {
12    Thread thread;
13    String str1, str2;
14    TwoStrings ts;
15    PrintStringsThread(String str1, String str2,
16                       TwoStrings ts) {
17      this.str1 = str1;
18      this.str2 = str2;
19      this.ts = ts;
20      thread = new Thread(this);
21      thread.start();
22    }
23    public void run() {
24      synchronized (ts) {
25        ts.print(str1, str2);
26      }
27    }
28  }
29  class TestThread {
30    public static void main(String args[]) {
31      TwoStrings ts = new TwoStrings();
32      new PrintStringsThread("Hello ", "there.", ts);
33      new PrintStringsThread("How are ", "you?", ts);
34      new PrintStringsThread("Thank you ",
35                             "very much!", ts);
36  }}
```

Komunikasi Antar Thread: Methods

Methods untuk komunikasi Interthread
public final void wait()
Menyebabkan thread ini menunggu sampai thread yang lain memanggil *notify* atau *notifyAll* method dari object ini. Hal ini dapat menyebabkan *InterruptedException*.
public final void notify()
Membangunkan thread yang telah memanggil method *wait* dari object yang sama.
public final void notifyAll()
Membangunkan semua thread yang telah memanggil method *wait* dari object yang sama.

Komunikasi Antar Thread





Contoh Produsen-Konsumen

```
1   class SharedData {
2     int data;
3     synchronized void set(int value) {
4       System.out.println("Generate " + value);
5       data = value;
6     }
7     synchronized int get() {
8       System.out.println("Get " + data);
9       return data;
10    }
11  }
12  class Producer implements Runnable {
13    SharedData sd;
14    Producer(SharedData sd) {
15      this.sd = sd;
16      new Thread(this, "Producer").start();
17    }
18    public void run() {
19      for (int i = 0; i < 10; i++) {
20        sd.set((int)(Math.random()*100));
21      }
22    }
23  }
```

```
24  class Consumer implements Runnable {
25    SharedData sd;
26    Consumer(SharedData sd) {
27      this.sd = sd;
28      new Thread(this, "Consumer").start();
29    }
30    public void run() {
31      for (int i = 0; i < 10 ; i++) {
32        sd.get();
33      }
34    }
35  }
36  class TestProducerConsumer {
37    public static void main(String args[])
38   throws Exception {
39      SharedData sd = new SharedData();
40      new Producer(sd);
41      new Consumer(sd);
42    }
43  }
```

- Contoh hasil:

```
Generate 8          Generate 49         Get 85
Generate 45         Get 49              Get 85
Generate 52         Generate 35         Get 85
Generate 65         Get 35              Generate 35
Get 65              Generate 39         Get 35
Generate 23         Get 39              Get 35
Get 23              Generate 85
```

Contoh Produsen-Konsumen yang telah diperbaiki

```
1   class SharedData {
2     int data;
3     boolean valueSet = false;
4     synchronized void set(int value) {
5       if (valueSet) {       //hanya dihasilkan jika mempumyai sebuah nilai
6         try {
7           wait();
8         } catch (InterruptedException ie) {
9         }
10      }
11      System.out.println("Generate " + value);
12      data = value;
13      valueSet = true;
14      notify();
15    }
16    synchronized int get() {
17      if (!valueSet) {
18    //producer belum memiliki suatu nilai
19        try {
20          wait();
21        } catch (InterruptedException ie) {
22        }
23      }
24      System.out.println("Get " + data);
25      valueSet = false;
26      notify();
27      return data;
28    }
29  }
30
31  /* Bagian kode tertentu tidak berubah. */
```

- Contoh hasil:

Generate 76	Generate 48	Generate 65
Get 76	Get 48	Get 65
Generate 25	Generate 29	Generate 38
Get 25	Get 29	Get 38
Generate 34	Generate 26	Generate 46
Get 34	Get 26	Get 46
Generate 84	Generate 86	
Get 84	Get 86	

Ringkasan

- Sinkronisasi
  - Mengunci sebuah Object
  - Keyword *synchronized*
    - Method header
    - Object

- Komunikasi Antar Thread (Interthread)
  - Methods
    - wait
    - notify
    - notifyAll