

## 1. Pengertian Serial Console

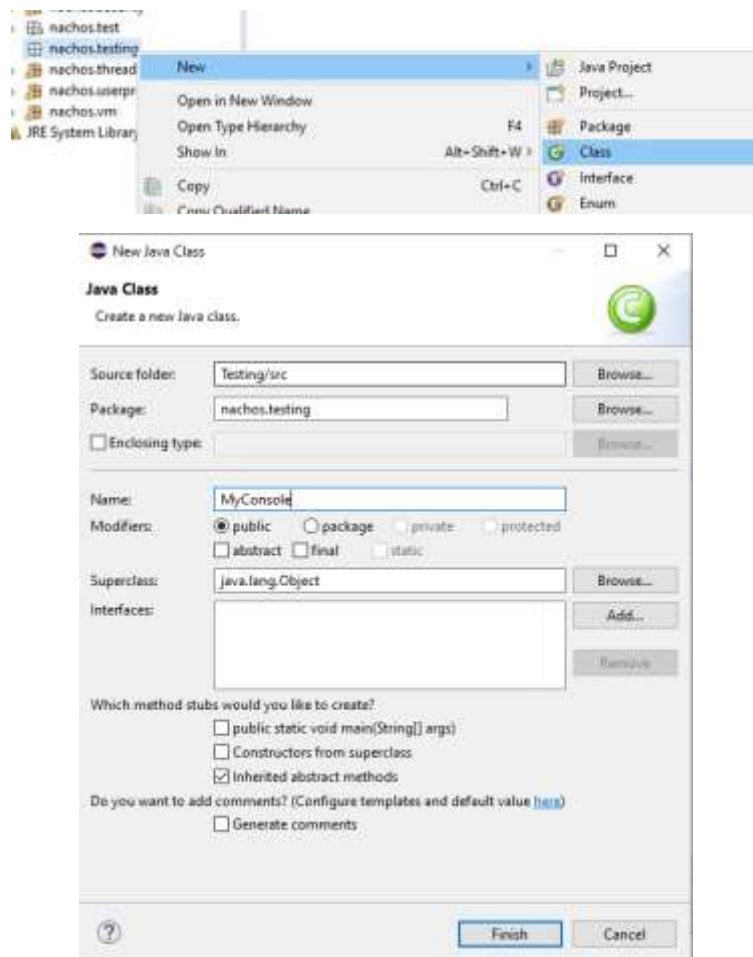
Serial Console merupakan salah satu fitur yang dapat digunakan untuk mensimulasikan cara kerja process I/O di NachOS. Ibaratnya, jika terbiasa menggunakan class Scanner untuk melakukan input output, maka kita dapat membuatnya sendiri dengan bantuan SerialConsole. Begitu juga method dengan System.out.println(), kalau biasanya menggunakan method ini untuk melakukan proses output, maka kita dapat membuatnya sendiri dengan library SerialConsole. Begitulah kira-kira pemanfaatan serial console di NachOS ini. Tujuannya supaya kita tahu alur dan cara kerja sistem operasi menerima/mengeluarkan data.

## 2. Dokumentasi Serial Console NachOS

Link : [http://www.cas.mcmaster.ca/~rzheng/course/Nachos\\_Tutorial/nachossu11.html](http://www.cas.mcmaster.ca/~rzheng/course/Nachos_Tutorial/nachossu11.html)

## 3. Penjelasan Detail

- a) Sesuai penjelasan diatas, kita dapat membuat I/O kita sendiri. Pertama Buatlah Class baru dengan nama “MyConsole”.



- b) Setelah itu, sesuai dokumentasi, untuk dapat membuat I/O, ada library Bernama “SerialConsole” yang berada satu class dengan package class Machine. Maka buatlah objek nya terlebih dahulu seperti di bawah ini:

```
MyConsole.java
1 package nachos.testing;
2
3 import nachos.machine.Machine;
4 import nachos.machine.SerialConsole;
5
6 public class MyConsole {
7     private SerialConsole sc = Machine.console();
8 }
```

- c) Di dokumentasi dijelaskan bahwa sebenarnya di belakang layar, mesin melakukan Input dan Output karakter dalam bentuk tipe data “byte”. Pertama, Mari kita buat Fungsi untuk membaca String sebagai berikut :

```
public String bacaString() {
    String str= "";

    return str;
}
```

- d) Kita tahu bahwa String merupakan kumpulan karakter. Maka dari itu, kita perlu membuat variable dengan tipe data char sebagai penampung. Variabel tersebut akan ditambahkan ke variable “str” di fungsi “bacaString()”. Logikanya akan terus menerima input sampai user menekan tombol enter, maka perlu dibuat looping dengan “do-while” menjadi seperti berikut :

```
MyConsole.java
1 package nachos.testing;
2
3 import nachos.machine.Machine;
4 import nachos.machine.SerialConsole;
5
6 public class MyConsole {
7     private SerialConsole sc = Machine.console();
8     char c;
9
10    public String bacaString() {
11        String str= "";
12        do {
13            if(c=='\n') {
14                break;
15            }else{
16                str+=c;
17            }
18        }while(true);
19        return str;
20    }
21 }
```

- e) Kita sebagai manusia tentu bisa melihat suatu input dengan mudah atau mengeluarkan suatu output, tapi tidak dengan mesin, dimana dia perlu membaca satu per satu sedangkan user terus menginput suatu hal (Sesuai dokumentasi yang mengatakan **“Only one byte can be queued at a time”**). Maka perlu mengatasi masalah ini dengan **“InterruptHandler”** yang berkaitan dengan yang Namanya **“Semaphore”** dan hal itu berkaitan dengan threading. Buatlah Variabel Objek Semaphore dan buatlah constructor MyConsole untuk Interrupt nya seperti berikut :

```
private SerialConsole sc = Machine.console();
char c;
private Semaphore semaphore = new Semaphore(0);

public MyConsole() {
    Runnable receiveInterruptHandler = new Runnable() {
        @Override
        public void run() {
            c=(char) sc.readByte();
            semaphore.V();
        }
    };
    Runnable sendInterruptHandler = new Runnable() {
        @Override
        public void run() {

        }
    };
    sc.setInterruptHandlers(receiveInterruptHandler, sendInterruptHandler);
}
```

**Dan tambahkan sedikit validasi di fungsi **“bacaString()”****

```
public String bacaString() {
    String str= "";
    do {
        semaphore.P();
        if(c=='\n') {
            break;
        }else{
            str+=c;
        }
    }while(true);
    return str;
}
```

**\*Catatan:**

**Semaphore memiliki 2 method yaitu ‘P’ dan ‘V’ dimana:**

‘P’ istilahnya Pause, karena hanya bisa baca 1 char setiap saat.

‘V’ istilahnya Resume, setelah selesai baca 1 char, tentu mesin bisa lanjut.

- f) Kita sudah membuat Fungsi membaca input sendiri. Bagaimana dengan membaca tipe data integer/bilangan bulat yaa?. Kita tidak perlu mengulang banyak, hanya tinggal memanfaatkan fungsi yang sudah kita buat menjadi :

```
public int bacaInteger() {
    int value=-1;
    try {
        value=Integer.parseInt(bacaString());
    } catch (Exception e) {
        // TODO: handle exception
    }
    return value;
}
```

Kenapa memerlukan “try-catch”? Jawabannya dikarenakan bisa saja ketika kita input dengan tipe data integer, namun terjadi human error dan ke input selain itu kan?. Makannya dibuat ada “try-catch” disana.

- g) Ketika sudah puas dengan Input, bagaimana cara membuat Output nya ya? Tentu sama saja logikanya. Ketika Input, mesin membaca satu-satu dan disimpan. Output pun sama hanya dibalik saja logikanya. Berarti dia akan “Write” secara satu satu seperti berikut :

```
public void cetak(String str)
{
    for(int i=0;i<str.length();i++)
    {
        sc.writeByte(str.charAt(i));
        semaphore.P();
    }
}
```

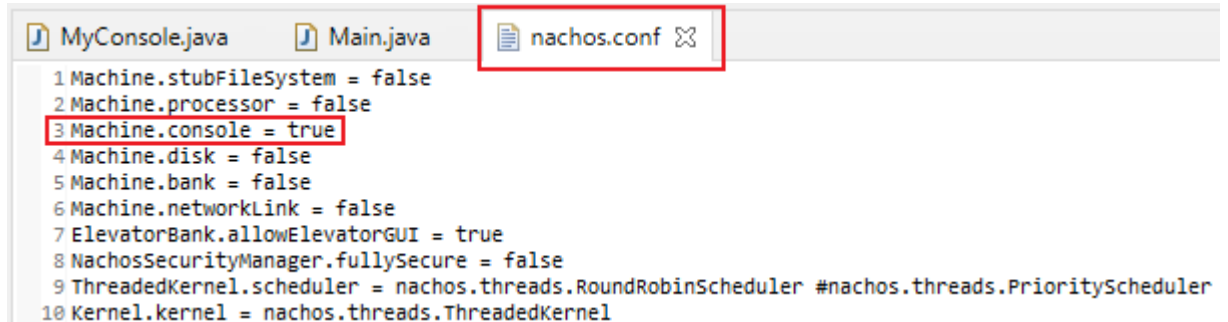
Dan Interrupt nya akan resume tiap mesin udh selesai cetak satu char seperti berikut :

```
Runnable sendInterruptHandler = new Runnable() {
    @Override
    public void run() {
        semaphore.V();
    }
};
```

- h) Sama halnya dengan input, bagaimana jika mau mengeluarkan output dengan enter? Kita tidak perlu membuat ulang semua, tinggal memanfaatkan fungsi yang sudah ada dan kita buat seperti berikut :

```
public void cetakEnter(String str)
{
    cetak(str);
    System.out.print('\n');
}
```

- i) Setelah Selesai, bukalah file “nachos.conf” dan ubah “Machine.console” menjadi True dikarenakan kita membuat I/O versi kita sendiri dan itu menggunakan SerialConsole.



```
MyConsole.java Main.java nachos.conf
1 Machine.stubFileSystem = false
2 Machine.processor = false
3 Machine.console = true
4 Machine.disk = false
5 Machine.bank = false
6 Machine.networkLink = false
7 ElevatorBank.allowElevatorGUI = true
8 NachosSecurityManager.fullySecure = false
9 ThreadedKernel.scheduler = nachos.threads.RoundRobinScheduler #nachos.threads.PriorityScheduler
10 Kernel.kernel = nachos.threads.ThreadedKernel
```

- j) Persiapan Kita sudah beres semua, saatnya melakukan uji coba apakah bisa dipakai hasil I/O buatan kita di Class Main sebagai berikut :

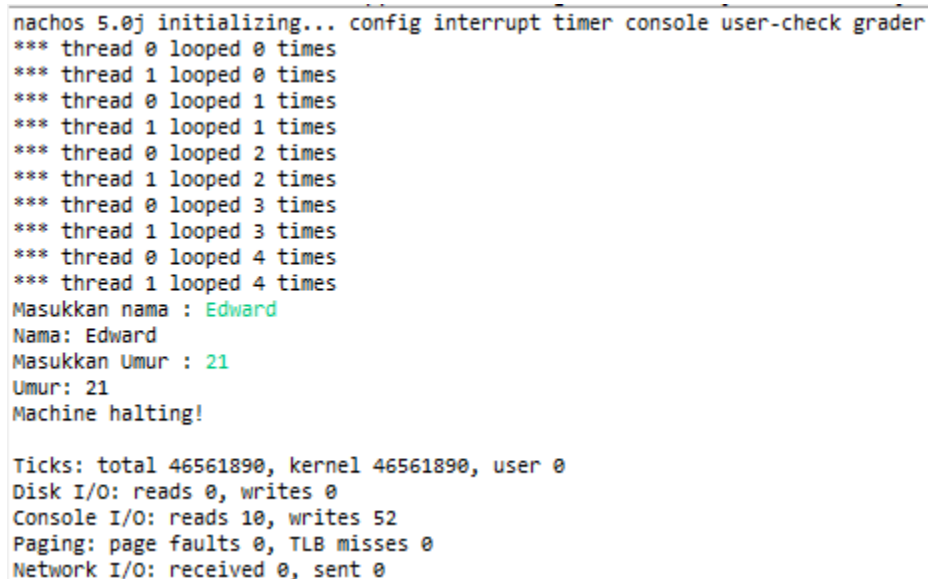
```
package nachos.testing;

public class Main {
    private MyConsole console = new MyConsole();

    public Main() {
        String nama;
        console.cetak("Masukkan nama : ");
        nama = console.bacaString();
        console.cetakEnter("Nama: " + nama);

        int umur;
        console.cetak("Masukkan Umur : ");
        umur = console.bacaInteger();
        console.cetakEnter("Umur: " + umur);
    }
}
```

### Hasil Output:



```
nachos 5.0j initializing... config interrupt timer console user-check grader
*** thread 0 looped 0 times
*** thread 1 looped 0 times
*** thread 0 looped 1 times
*** thread 1 looped 1 times
*** thread 0 looped 2 times
*** thread 1 looped 2 times
*** thread 0 looped 3 times
*** thread 1 looped 3 times
*** thread 0 looped 4 times
*** thread 1 looped 4 times
Masukkan nama : Edward
Nama: Edward
Masukkan Umur : 21
Umur: 21
Machine halting!

Ticks: total 46561890, kernel 46561890, user 0
Disk I/O: reads 0, writes 0
Console I/O: reads 10, writes 52
Paging: page faults 0, TLB misses 0
Network I/O: received 0, sent 0
```

## k) Kodingan Final (MyConsole.java)

```
package nachos.testing;

import nachos.machine.Machine;
import nachos.machine.SerialConsole;
import nachos.threads.Semaphore;

public class MyConsole {
    private SerialConsole sc = Machine.console();
    char c;
    private Semaphore semaphore = new Semaphore(0);

    public MyConsole() {
        Runnable receiveInterruptHandler = new Runnable() {
            @Override
            public void run() {
                c=(char) sc.readByte();
                semaphore.V();
            }
        };

        Runnable sendInterruptHandler = new Runnable() {
            @Override
            public void run() {
                semaphore.V();
            }
        };

        sc.setInterruptHandlers(receiveInterruptHandler, sendInterruptHandler);
    }

    public String bacaString() {
        String str= "";
        do {
            semaphore.P();
            if(c=='\n') {
                break;
            }else{
                str+=c;
            }
        }while(true);
        return str;
    }

    public int bacaInteger() {
        int value=-1;
        try {
            value=Integer.parseInt(bacaString());
        } catch (Exception e) {
            // TODO: handle exception
        }
        return value;
    }

    public void cetak(String str){
        for(int i=0;i<str.length();i++){
            sc.writeByte(str.charAt(i));
            semaphore.P();
        }
    }

    public void cetakEnter(String str){
        cetak(str);
        System.out.print('\n');
    }
}
```