

LAPORAN PRAKTIKUM
JAVA: EXCEPTION HANDLING

Dosen Pengampu : Alun Sujjada, S.Kom, M.T



Nama : Feni Nurul Hafifah
NIM : 20230040053
Kelas : TI23F
Mata Kuliah : Pemrograman Berorientasi Objek

PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS KOMPUTER DAN DESAIN
NUSA PUTRA UNIVERSITY

2025

Tujuan Praktikum

Memahami dan mengimplementasikan penanganan kesalahan (exception handling) dalam bahasa pemrograman Java, baik menggunakan exception bawaan maupun membuat exception sendiri.

Percobaan dan Analisa

Percobaan 1

Jalankan program dibawah ini, amati error yang terjadi. Kemudian perbaiki seperti pada program dibawahnya.

```
public class Exception {  
  
    public static void main(String[] args) {  
        int a[]=new int[5];    a[5]=100;  
    }  
}
```

```
PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10> cd  
"d:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi  
10\src" ; if ($?) { javac Exception1.java } ; if  
($?) { java Exception1 }  
Terjadi pelanggaran memory  
PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\ses  
i10\src>
```

Analisa :

Pada saat program dijalankan ternyata output tidak sesuai dengan yang diharapkan ternyata ada beberapa kesalahan :

- Program ini membuat sebuah array a dengan panjang 5 elemen.
- Dalam Java, indeks array dimulai dari 0 sampai 4 untuk array berukuran 5.
- Baris `a[5] = 100;` mencoba mengakses indeks ke-5, yang tidak valid, sehingga akan terjadi `ArrayIndexOutOfBoundsException` saat program dijalankan.
- Karena tidak ada mekanisme penanganan error (seperti try-catch), maka program akan crash dengan menampilkan exception.

Pembetulan program:

```

public class Exception {

    public static void main(String[] args)
    {
        int a[]=new int[5];    try    {
            a[5]=100;
        }
        catch(Exception e)
        {
            System.out.println("Terjadi pelanggaran memory");
        }
    }
}

```

Analisa :

Pada saat diperbaiki ternyata output bisa menampilkan hasil :

- Saat Try-catch ditambahkan untuk menangkap exception yang mungkin terjadi.
- Saat a[5] = 100 dieksekusi dan error terjadi, program tidak crash, tetapi langsung masuk ke blok catch.
- Pesan "Terjadi pelanggaran memory" ditampilkan, memberikan indikasi bahwa terjadi kesalahan dalam mengakses array.
- Ini adalah praktik yang baik dalam pemrograman Java: menggunakan mekanisme exception handling untuk menghindari program berhenti secara mendadak.

Percobaan 2

Jalankan program dibawah ini, amati error yang terjadi. Kemudian perbaiki seperti pada program dibawahnya.

```

public class Exception2 {
    public static void main(String[] args) {
        int i=0;
        String greeting[]={
            "Hello World!",
            "No, I mean it!",
            "Hello World"
        };    while(i<4)
        {
            System.out.println(greeting[i]);    i++;
        }
    }
}

```

Pembetulan program:

```
public class Exception2 {
    public static void main(String[] args) {        int
i=0;
    String
greetings[]={        "Hello World!",
        "No,I mean it!",
        "HELLO WORLD!"
    };
    while(i<4)

{        try
        {
            System.out.println(greetings[i]);
i++;        }
        catch(ArrayIndexOutOfBoundsException e)
        {
            System.out.println("Resetting index value");        i=0;
        }
    }
}
```

```
) { javac Exception1.java } ; if ($?) { java Exception1 }
Terjadi pelanggaran memory
PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\src> cd "d:\feninurulhafifah\CODING\Tugas-PBO\se
si10\sesi10\src\" ; if ($?) { javac Exception2.java } ; if ($?) { java Exception2 }
Hello World !
No, I mean it !
HELLO WORLD !
PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\src>
```

Analisa :

Program pada percobaan kedua bertujuan untuk mencetak elemen-elemen dari array greeting yang berisi tiga buah string. Namun, terdapat kesalahan pada kondisi perulangan while($i < 4$) karena jumlah elemen array hanya 3 (indeks 0 sampai 2), sedangkan perulangan mencoba mengakses hingga indeks ke-3 yang tidak tersedia. Akibatnya, saat program dijalankan, akan terjadi `ArrayIndexOutOfBoundsException`.

Untuk memperbaikinya, digunakan blok try-catch yang membungkus proses akses elemen array. Jika terjadi exception karena indeks keluar dari batas, program akan

menangkapnya melalui blok catch, mencetak pesan “Resetting index value”, dan mengatur ulang nilai indeks i ke 0. Dengan cara ini, program tidak akan crash, dan akan terus mencetak elemen array secara berulang. Teknik ini menunjukkan penggunaan penanganan exception untuk menjaga agar program tetap berjalan meskipun terjadi kesalahan dalam pengaksesan array.

Percobaan 3

Jalankan program dibawah ini, amati error yang terjadi. Kemudian perbaiki seperti pada program dibawahnya.

```
public class Exception3 {  
  
    public static void main(String[] args) {        int bil=10;  
        System.out.println(bil/0);  
    }  
}
```

Pembetulan Program:

```
public class Exception3 {  
  
    public static void main(String[] args) {        int  
bil=10;        try        {  
        System.out.println(bil/0);  
        }  
        catch(Exception e)  
        {  
        System.out.println("Ini menghandle error yang terjadi");  
        }  
    }  
}
```

Pembetulan Program:

```
public class CobaException3 {  
  
    public static void main(String[] args)  
{  
    int bil=10;    try    {  
        System.out.println(bil/0);  
    }  
    catch(ArithmeticException e)  
    {  
        System.out.println("Terjadi Aritmatika error");  
    }  
    catch(Exception  
e)  
    {  
        System.out.println("Ini menghandle error yang terjadi");  
    }  
    }  
}
```

```
● PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src> cd "d:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src" ; if ($?) { javac Exception3.java } ; if ($?) { java Exception3 }  
Aritmatika Error  
○ PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src> |
```

Analisa yang bisa didapat pada program ke 3 :

Program pada percobaan kedua bertujuan untuk mencetak elemen-elemen dari array greeting yang berisi tiga buah string. Namun, terdapat kesalahan pada kondisi perulangan while($i < 4$) karena jumlah elemen array hanya 3 (indeks 0 sampai 2), sedangkan perulangan mencoba mengakses hingga indeks ke-3 yang tidak tersedia.

Akibatnya, saat program dijalankan, akan terjadi `ArrayIndexOutOfBoundsException`. Untuk memperbaikinya, digunakan blok try-catch yang membungkus proses akses elemen array. Jika terjadi exception karena indeks keluar dari batas, program akan menangkapnya melalui blok catch, mencetak pesan “Resetting index value”, dan mengatur ulang nilai indeks i ke 0. Dengan cara ini, program tidak akan crash, dan akan terus mencetak elemen array secara berulang. Teknik ini menunjukkan penggunaan penanganan exception untuk menjaga agar program tetap berjalan meskipun terjadi kesalahan dalam pengaksesan array.

Percobaan 4

Jalankan program dibawah ini, amati error yang terjadi. Kemudian perbaiki seperti pada program dibawahnya.

```
public class CobaException4 {
    public static void main(String[] args) {        int
bil=10;
    String b[]={ "a","b","c"};
    try    {
        System.out.println(b[3]);
        System.out.println(bil/0);
    }
    catch(ArithmeticException e)
    {
        System.out.println("Terjadi Aritmatika error");
    }
    catch(ArrayIndexOutOfBoundsException e)
    {
        System.out.println("Melebihi jumlah array");
    }
    catch(Exception e)
    {
        System.out.println("Ini menghandle error yang terjadi");
    }
    }
}
```

Pembetulan Program:

```
public class CobaException4 {
    public static void main(String[] args) {        int
bil=10;
    String b[]={ "a","b","c"};
    try    {
        System.out.println(bil/0);
        System.out.println(b[3]);
    }
    catch(ArithmeticException e)
    {
        System.out.println("Terjadi Aritmatika error");
    }
    catch(ArrayIndexOutOfBoundsException e)
    {
        System.out.println("Melebihi jumlah array");
    }
}
```

```

    }
    catch(Exception e)
    {
        System.out.println("Ini menhandle error yang terjadi");
    }
}
}

```

```

PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src> cd "d:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src\" ; if ($?) { javac Exception4.java } ; if ($?) { java Exception4 }
Terjadi Aritmatika error
PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src> 

```

Analisa yang didapat :

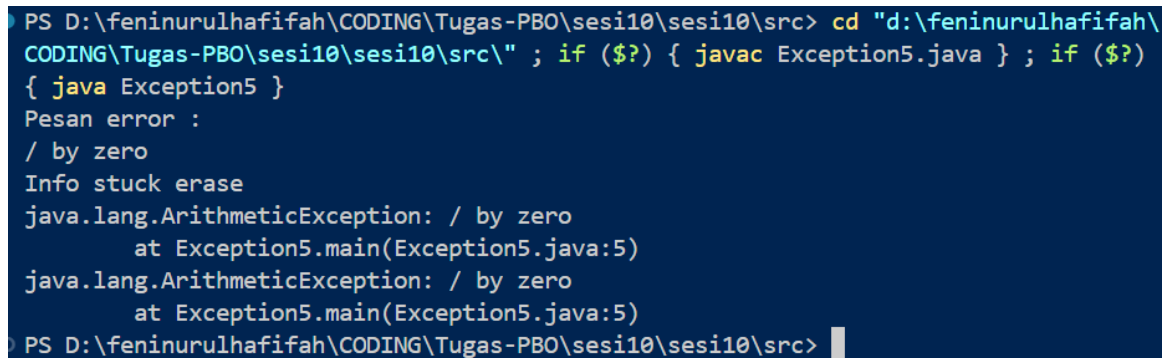
Program ini mencoba mencetak dua baris perintah: mengakses elemen array dengan indeks yang tidak tersedia (b[3]) dan melakukan pembagian dengan nol. Versi awal program mencoba mencetak b[3] terlebih dahulu, sehingga program langsung melempar `ArrayIndexOutOfBoundsException`. Dalam versi perbaikan, urutan perintah dalam blok try diubah agar pembagian (bil/0) dilakukan terlebih dahulu.

Dengan begitu, exception yang pertama kali ditangani adalah `ArithmeticException`. Hal ini menunjukkan bahwa urutan perintah mempengaruhi jenis exception yang ditangani terlebih dahulu. Program juga menggunakan beberapa catch untuk menangani berbagai jenis error, yang merupakan pendekatan umum dalam menangani beberapa kemungkinan kesalahan runtime.

Percobaan 5

Jalankan program dibawah ini, berikan analisa penggunaan try dan catch pada program dibawah ini.

```
public class Exception5 {  
  
    public static void main(String[] args)  
    {  
        int bil=10;    try    {  
            System.out.println(bil/0);  
        }  
        catch(ArithmeticException e)  
        {  
            System.out.println("Pesan error: ");  
            System.out.println(e.getMessage());        System.out.println("Info  
stack erase");  
            e.printStackTrace();  
            e.printStackTrace(System.out);        }  
        catch(Exception e)  
        {  
            System.out.println("Ini menghandle error yang terjadi");  
        }  
    }  
}
```



```
PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src> cd "d:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src\" ; if ($?) { javac Exception5.java } ; if ($?) { java Exception5 }  
Pesan error :  
/ by zero  
Info stuck erase  
java.lang.ArithmeticException: / by zero  
    at Exception5.main(Exception5.java:5)  
java.lang.ArithmeticException: / by zero  
    at Exception5.main(Exception5.java:5)  
PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src>
```

Analisa :

Program ini menunjukkan penggunaan blok try-catch yang menangani ArithmeticException. Saat pembagian dengan nol terjadi, program tidak hanya menangkap exception, tetapi juga menampilkan pesan error menggunakan

getMessage(), serta menampilkan jejak eksekusi program (printStackTrace()). Ini berguna untuk debugging karena menampilkan informasi lengkap mengenai lokasi dan jenis error.

Selain itu, tersedia catch(Exception e) sebagai penanganan umum jika terjadi jenis error lain. Percobaan ini menunjukkan bagaimana exception dapat digunakan tidak hanya untuk menangkap error, tetapi juga untuk memberikan informasi detail yang bermanfaat dalam proses pengembangan dan perbaikan program.

Percobaan 6

Jalankan program dibawah ini, berikan analisa penggunaan try dan catch pada program dibawah ini.

```
public class ThrowExample {  
  
    static void demo()  
    {  
        NullPointerException t;  
        t=new NullPointerException("Coba Throw");        throw t;  
  
        // Baris ini tidak lagi dikerjakan;  
        System.out.println("Ini tidak lagi dicetak");  
    }  
  
    public static void main(String[] args)  
    {  
        try {  
            demo();  
            System.out.println("Selesai");  
        }  
        catch(NullPointerException e)  
        {  
            System.out.println("Ada pesan error: "+e);  
        }  
    }  
}
```

```
CODING\Tugas-PBO\sesi10\sesi10\src\" ; if ($?) { javac ThrowExample.java } ; if ($?  
) { java ThrowExample }  
Ada pesan error: java.lang.NullPointerException: Coba Throw  
PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src> |
```

Analisa :

Pada percobaan ini, method `demo()` secara eksplisit melempar `NullPointerException` menggunakan pernyataan `throw`. Setelah exception dilempar, baris setelahnya tidak akan dijalankan. Di method `main()`, blok `try` memanggil `demo()`, dan exception ditangkap oleh blok `catch`, lalu menampilkan pesan error. Ini menunjukkan bagaimana exception dapat dilempar secara manual menggunakan `throw`, dan bagaimana program dapat diarahkan ke blok `catch` jika exception terjadi.

Percobaan 7

Jalankan program dibawah ini, berikan analisa penggunaan `try` dan `catch` pada program dibawah ini.

```
public class ThrowExample2 {  
  
    public static void main(String[] args)  
    {  
        try {  
            throw new Exception("Here's my Exception");  
        }  
        catch(Exception e)  
        {  
            System.out.println("Caught Exception");  
            System.out.println("e.getMessage():"+e.getMessage());  
            System.out.println("e.toString():"+e.toString());  
            System.out.println("e.printStackTrace():");  
            e.printStackTrace();  
        }  
    }  
}
```

```
PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src> cd "d:\feninurulha  
fifah\CODING\Tugas-PBO\sesi10\sesi10\src\" ; if ($?) { javac ThrowExample2.ja  
va } ; if ($?) { java ThrowExample2 }  
Caught Exception  
e.getMessage():Here's my Exception  
e.toString(): java.lang.Exception: Here's my Exception  
e.printStackTrace() :  
java.lang.Exception: Here's my Exception  
    at ThrowExample2.main(ThrowExample2.java:4)  
PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src> █
```

Analisa :

Program ini kembali menggunakan throw untuk secara eksplisit melempar objek Exception dengan pesan khusus. Setelah exception dilempar, blok catch menangkapnya dan menampilkan informasi lengkap tentang exception, seperti pesan (getMessage()), representasi string (toString()), dan jejak tumpukan eksekusi (printStackTrace()). Ini merupakan teknik umum dalam debugging, yang membantu pengembang memahami jenis dan lokasi error dalam program.

Percobaan 8

Jalankan program dibawah ini, berikan analisa penggunaan throws pada program dibawah ini.

<pre>import java.io.*; public class Test3 { public void methodA(){ System.out.println("Method A"); } public void methodB() throws IOException { System.out.println(20/0); System.out.println("Method B"); } }</pre>
<pre>class Utama { public static void main(String[] args) throws IOException { Test3 c=new Test3(); c.methodA(); c.methodB(); } }</pre>

Kemudian coba ubah class utama diatas dengan yang program baru di bawah ini:

```
class Utama
{
    public static void main(String[] args)
    {
        Test3 o=new Test3();
        o.methodA();
    try
        {
            o.methodB();        }
        catch(Exception e)
        {
            System.out.println("Error di Method B");
        }
        finally
        {
            System.out.println("Ini selalu dicetak");
        }
    };
}
```

```
PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src> cd "d:\feninurulha
fifah\CODING\Tugas-PBO\sesi10\sesi10\src\" ; if ($?) { javac Utama.java } ; i
f ($?) { java Utama }
Method A
Error di Method B
Ini selalu dicetak
PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src>
```

Analisa :

Dalam percobaan tersebut, method methodB() dideklarasikan untuk melempar IOException, meskipun error yang sebenarnya terjadi adalah ArithmeticException karena pembagian dengan nol. Versi pertama program menggunakan throws IOException di method main(), namun tidak menangkap error, sehingga program akan crash. Pada versi perbaikan, digunakan blok try-catch untuk menangkap exception dan menampilkan pesan “Error di Method B”. Ditambahkan juga blok finally yang berfungsi mencetak pesan “Ini selalu dicetak”, menunjukkan bahwa blok finally akan selalu dijalankan, terlepas dari apakah terjadi error atau tidak. Percobaan ini menyoroti perbedaan antara deklarasi throws dan penggunaan try-catch.

Percobaan 9

Jalankan program membalik string (reverse) berikut ini, coba hapus isi dari method reverse ("This is a string") kemudian jalankan kembali.

```
class Propagate {

    public static void main(String [] args)
    {
        try
        {
            System.out.println(reverse("This is a string"));
        }
        catch(Exception e)
        {
            System.out.println("The String was blank");
        }
        finally
        {
            System.out.println("All done");
        }
    }

    public static String reverse(String s) throws Exception
    {
        if(s.length()==0)
        {
            throw new Exception();
        }
        String reverseStr = "";
        for(int i=s.length()-1 ; i>=0 ; --i){
            reverseStr+=s.charAt(i);
        }
        return reverseStr;
    }
}
```

Analisa :

```
PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src> cd "d:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src" ; if ($?) { javac Propagate.java ; if ($?) { java Propagate }
gnirts a si sihT
All done
PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src> █
```

Program ini mendefinisikan method reverse() yang akan membalik string. Jika string yang diberikan kosong, maka exception akan dilempar. Saat dijalankan

dengan string kosong, method reverse() akan melempar exception, yang ditangkap oleh catch di method main(), dan menampilkan pesan bahwa string kosong. Selain itu, blok finally akan selalu dijalankan dan mencetak "All done". Percobaan ini menunjukkan penggunaan try-catch-finally dalam pengelolaan error dan menjaga eksekusi bagian program tertentu tetap berjalan.

Dan ternyata output yang dihasilkan tersebut kata This is a string di balikan karena reserve tersebut.

Percobaan 10

Program berikut ini menerapkan IOException ketika membuat objek dari class RandomAccessFile (java.io) yang menghasilkan file txt.

```
class RandomAccessRevisi
{
    public static void main(String[] args) {

        String bookList[]={"Satu","Dua","Tiga"};
        int yearList[]={ 1920,1230,1940};    try    {
            RandomAccessFile books = new RandomAccessFile
            ("books.txt","rw");

            for(int i=0;i<3;i++)
            {
                books.writeUTF(bookList[i]);          books.writeInt(yearList[i]);
            }
            books.seek(0);
            System.out.println(books.readUTF()+" "+books.readInt());
            System.out.println(books.readUTF()+" "+books.readInt());
            books.close();
        }
        catch(IOException e)
        {
            System.out.println("Indeks melebihi batas");
        }
        System.out.println("test");
    }
}
```

```

PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src> cd "d:\feninurulha
fifah\CODING\Tugas-PBO\sesi10\sesi10\src\" ; if ($?) { javac RandomAccessRevi
si.java } ; if ($?) { java RandomAccessRevisi }
Satu 1920
Dua 1230
Tiga 1940
Test selesai.
PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src>

```

Analisa :

Program ini menggunakan class `RandomAccessFile` untuk menulis dan membaca data ke/dari file teks. Program mencoba menulis string dan integer ke dalam file, lalu membaca kembali sebagian dari data tersebut. Jika terjadi kesalahan I/O seperti file tidak ditemukan atau kesalahan pembacaan data, program akan menangkap `IOException` dan menampilkan pesan error. Ini menunjukkan bagaimana file handling harus dilakukan dalam blok try-catch untuk mengantisipasi kemungkinan error saat berinteraksi dengan sistem file.

Percobaan 11

Program berikut ini menunjukkan proses throw and catch pada class yang extends `Throwable`.

```

class RangeErrorException extends Throwable
{
    public RangeErrorException(String s)
    {
        super(s);    }
}

```

```

    public static void main(String[] args)
    {
        int position=1;
    try    {
        if(position>0)
        {
            throw new RangeErrorException("Position " +position);
        }    }
    catch(RangeErrorException e)
    {
        System.out.println("Range error: " +e.getMessage());    }
    System.out.println("This is the last program.");
    }
}

```



```
PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src> cd "d:\feninurulha
fifah\CODING\Tugas-PBO\sesi10\sesi10\src\" ; if ($?) { javac RangeErrorExcept
ion.java } ; if ($?) { java RangeErrorException }
Range error: Position 1
This is the last program.
PS D:\feninurulhafifah\CODING\Tugas-PBO\sesi10\sesi10\src> █
```

Analisa :

Percobaan ini menunjukkan bagaimana membuat class exception sendiri (RangeErrorException) yang merupakan turunan dari Throwable. Jika kondisi tertentu terpenuhi, exception dilempar dan ditangkap dalam blok catch, lalu pesan kesalahan ditampilkan. Ini memperlihatkan bahwa kita dapat membuat exception kustom dengan nama dan logika sendiri, yang sangat berguna untuk validasi aplikasi yang lebih kompleks.

Percobaan 12

Program berikut ini menunjukkan proses throw and catch pada class yang extends Exception.

```
class MyException extends Exception{

    private String Teks;    MyException(String s)
    {
        Teks="Exception generated by: "+s;
        System.out.println(Teks);
    }
}
class Eksepsi
{
    static void tampil(String s)throws Exception
    {
        System.out.println("Tampil");        if(s.equals("amir"))
        {
            throw new MyException(s);
        }
        System.out.println("OK!");
    }

    public static void main(String[] args)throws Exception
    {
        try
        {
            tampil("ali");        tampil("amir");
        }
        catch(MyException ex)
        {
            System.out.println("Tangkap:"+ex);
        }
    }
}
```

```
Tampil  
OK!  
Tampil  
Exception generated by: amir  
Tangkap: MyException
```

Analisa :

Mirip dengan percobaan sebelumnya, pada percobaan ini dibuat class exception sendiri MyException yang merupakan turunan dari Exception. Dalam method tampil(), jika nama yang diterima adalah "amir", maka exception akan dilempar. Di main(), method tersebut dipanggil dua kali, dan hanya pemanggilan kedua yang menyebabkan exception. Exception yang dilempar ditangkap oleh blok catch dan ditampilkan. Percobaan ini menekankan pentingnya validasi input dan bagaimana exception kustom dapat digunakan untuk mengontrol alur program berdasarkan kondisi tertentu.