Nama : Restu Bumi Ryan Ramadhan
Kelas : TI21A
NIM : 20210040006

Tugas Sesi 11

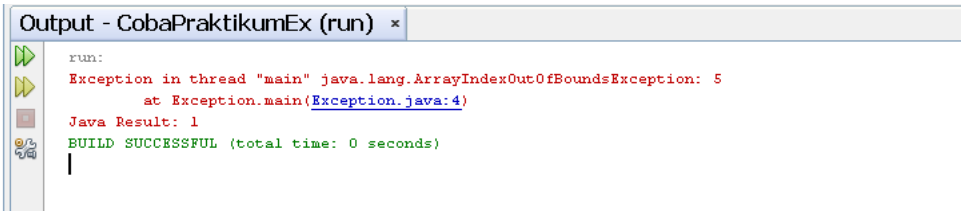
Pemrograman Berorientasi Objek

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;
 }
}
```

Output :



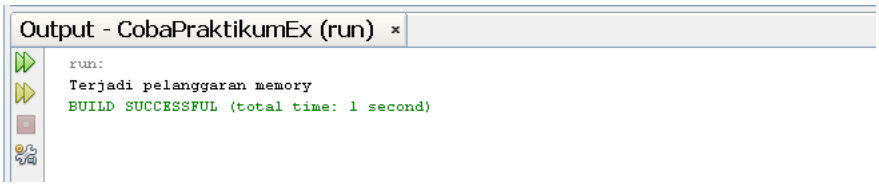
Hasil Analisa :

Program yang ada diatas tidak dapat dijalankan karena terdapat error pada baris ketiga. Error tersebut terjadi karena Kita mencoba mengakses indeks array di luar batas yang diizinkan. Array di Java memiliki indeks yang dimulai dari 0 sampai ukuran array - 1. Jadi, jika Kita memiliki array dengan ukuran 5, maka indeks yang valid adalah 0 sampai 4.

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");
 }
 }
}
```

Output :



Analisa :

Program yang Kita berikan masih tidak dapat dijalankan karena masih terdapat error pada baris keempat.

Error tersebut terjadi karena Kita mencoba mengakses indeks array di luar batas yang diizinkan. Array di Java memiliki indeks yang dimulai dari 0 sampai ukuran array - 1. Jadi, jika Kita memiliki array dengan ukuran 5, maka indeks yang valid adalah 0 sampai 4.

Untuk menangani error tersebut, Kita dapat menggunakan konstruksi try-catch. Di dalam blok try, Kita dapat menuliskan kode yang mungkin menyebabkan error. Jika error terjadi, maka kode di dalam blok catch akan dijalankan.

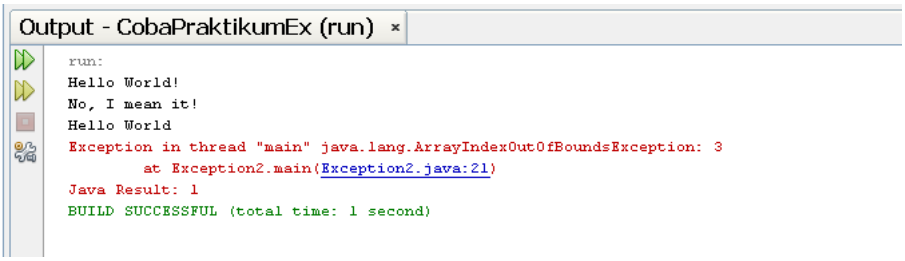
Dengan mengganti indeks 5 menjadi 4, program akan berjalan dengan benar dan tidak terjadi error. Jika Kita menjalankan program tersebut, maka tidak akan ada output dari program karena tidak terjadi error. Namun, jika Kita mengubah indeks yang Kita akses menjadi indeks yang tidak valid, misalnya 5 atau 6, maka akan terjadi error dan kode di dalam blok catch akan dijalankan.

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++;
 }
 }
}
```

Output :



Analisa :

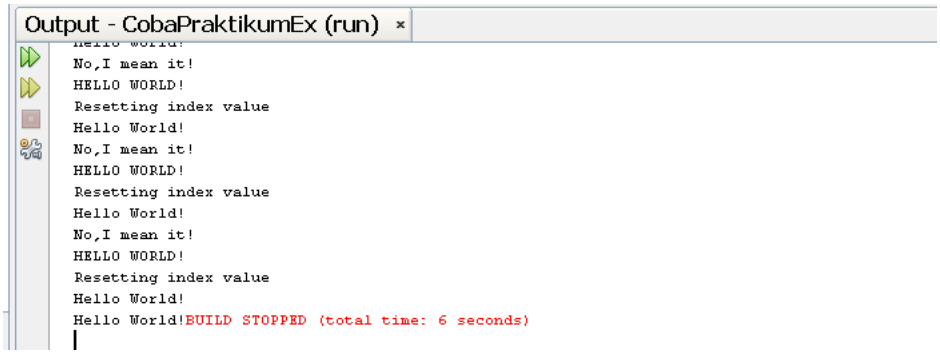
Jika dijalankan, program diatas akan menghasilkan error "ArrayIndexOutOfBoundsException", yang terjadi karena indeks array greeting yang diakses melebihi panjang dari array tersebut (yang hanya sebanyak 3 elemen).

Untuk memperbaiki error tersebut, kita dapat mengubah perulangan while menjadi while(i < 3), sehingga indeks yang diakses tidak melebihi panjang dari array greeting.

Perbaikan 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;
 }
 }
 }
}
```

Output :



Analisa :

Program diatas akan menjalankan perulangan while dengan indeks i yang dimulai dari 0. Di dalam perulangan tersebut, program akan mencoba mencetak elemen dari array greetings dengan indeks yang sama dengan i. Jika tidak terjadi error, maka indeks i akan ditambah 1. Namun, jika terjadi error ArrayIndexOutOfBoundsException, maka kode di dalam blok catch akan dijalankan. Di dalam blok catch, program akan mencetak pesan "Resetting index value" dan mengubah nilai indeks i menjadi 0.

Program tersebut akan mencetak pesan-pesan yang ada di dalam array greetings secara berulang-ulang sampai indeks i mencapai 4. Namun, setelah indeks i mencapai 4, perulangan akan berakhir karena kondisi di dalam perulangan while tidak lagi terpenuhi.

Untuk memperbaiki error yang terjadi, Kita dapat mengubah kondisi di dalam perulangan while menjadi "i<=3" agar perulangan tersebut berakhir setelah semua elemen dari array greetings dicetak. Berikut adalah program yang telah diperbaiki:

```java
public class Exception2 {
    Run | Debug
    public static void main(String[] args) {
    int i=0;
    String greetings[]={
    "Hello World!",
    "No,I mean it!",
    "HELLO WORLD!"
    };
    while(i<3)
    {
    try
    {
    System.out.println(greetings[i]);
    i++;
    }
    catch(ArrayIndexOutOfBoundsException e)
    {
    System.out.println(x: "Resetting index value");
    i=0;
    }
    }
    }
}
```

Dengan perubahan tersebut, program akan mencetak semua elemen dari array greetings dan tidak terjadi error ArrayIndexOutOfBoundsException. Output yang akan dihasilkan oleh program tersebut adalah:

Output :

```
Hello World!
No,I mean it!
HELLO WORLD!
```

Percobaan 3 :

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

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

Output :

```
Output - CobaPraktikumEx (run) ×
run:
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Exception3.main(Exception3.java:14)
Java Result: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

Analisa :

Program yang ada diatas tidak dapat dijalankan karena terdapat error pada baris keempat. Error tersebut terjadi karena Kita mencoba melakukan operasi pembagian dengan bilangan nol (0). Pembagian dengan nol merupakan operasi yang tidak valid dalam matematika, sehingga akan terjadi error pada saat program dijalankan.

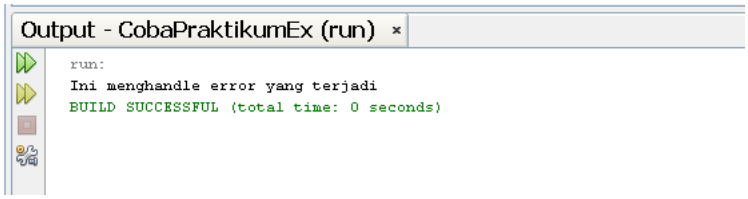
Untuk menangani error tersebut, Kita dapat menggunakan konstruksi try-catch. Di dalam blok try, Kita dapat menuliskan kode yang mungkin menyebabkan error. Jika error terjadi, maka kode di dalam blok catch akan dijalankan.

Untuk memperbaiki error tersebut, Kita dapat menambahkan blok catch yang akan menangani error pembagian dengan nol.

Perbaikan 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");
 }
 }
}
```

Output :



Analisa :

Program diatas akan mencoba melakukan operasi pembagian dengan bilangan nol (0). Pembagian dengan nol merupakan operasi yang tidak valid dalam matematika, sehingga akan terjadi error pada saat program dijalankan.

Untuk menangani error tersebut, kita telah menambahkan blok catch yang akan menangani error yang terjadi di dalam blok try. Di dalam blok catch, program akan mencetak pesan "Ini menghandle error yang terjadi".

Namun, jenis exception yang kita handle di dalam blok catch adalah Exception, yang merupakan superclass dari semua exception di Java. Jenis exception yang terjadi pada saat pembagian dengan nol adalah ArithmeticException, yang merupakan subclass dari Exception.

Oleh karena itu, kita perlu mengubah jenis exception yang kita handle di dalam blok catch menjadi ArithmeticException agar blok catch tersebut benar-benar dapat menangani error yang terjadi.

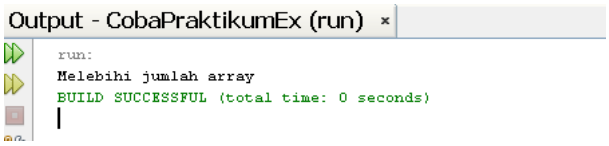
Dengan mengganti jenis exception yang dihandle di dalam blok catch menjadi ArithmeticException, program akan berjalan dengan benar dan tidak terjadi error. Jika kita menjalankan program tersebut, maka akan tercetak pesan "Ini menghandle error yang terjadi". Namun, jika Kita mengubah nilai yang akan dibagi dengan nol menjadi bilangan yang lain, misalnya 5 atau 7, maka program akan berjalan dengan benar dan mencetak hasil pembagian tersebut.

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");
 }
 }
}
```

Output :



Analisa :

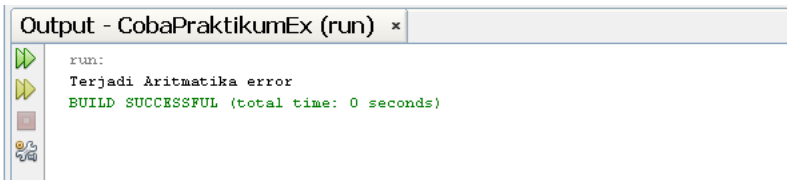
Program diatas akan mencoba mencetak elemen array dengan indeks 3 dan melakukan operasi pembagian dengan bilangan nol. Karena indeks array yang Kita akses melebihi batas yang diizinkan, maka akan terjadi error ArrayIndexOutOfBoundsException. Untuk menangani error tersebut, Kita telah menambahkan blok catch yang akan menangani jenis exception tersebut. Di dalam blok catch tersebut, program akan mencetak pesan "Melebihi jumlah array". Setelah error ArrayIndexOutOfBoundsException terjadi, program akan mencoba melakukan operasi pembagian dengan bilangan nol. Operasi pembagian dengan nol merupakan operasi yang tidak valid dalam matematika, sehingga akan terjadi error lagi pada saat operasi tersebut dijalankan. Untuk menangani error tersebut, Kita juga telah menambahkan blok catch yang akan menangani jenis exception ArithmeticException. Di dalam blok catch tersebut, program akan mencetak pesan "Terjadi Aritmatika error".

Setelah error tersebut dihandle, program akan berjalan dengan benar. Jika Kita menjalankan program tersebut, maka akan tercetak pesan "Melebihi jumlah array" kemudian "Terjadi Aritmatika error". Namun, jika Kita mengubah nilai yang akan dibagi dengan nol menjadi bilangan yang lain, misalnya 5 atau 7, dan juga mengubah indeks array yang akan dicetak menjadi indeks yang valid, maka program akan berjalan dengan benar dan mencetak hasil pembagian dan elemen array sesuai dengan yang diinginkan.

Perbaikan 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 menghandle error yang terjadi");
 }
 }
}
```

Output :



Analisa :

Program diatas akan mencoba melakukan operasi pembagian dengan bilangan nol (0) terlebih dahulu, kemudian mencetak elemen array dengan indeks 3. Operasi pembagian dengan nol merupakan operasi yang tidak valid dalam matematika, sehingga akan terjadi error pada saat operasi tersebut dijalankan.

Untuk menangani error tersebut, kita telah menambahkan blok catch yang akan menangani jenis exception ArithmeticException. Di dalam blok catch tersebut, program akan mencetak pesan "Terjadi Aritmatika error".

Setelah error tersebut dihandle, program akan melanjutkan eksekusi dengan mencoba mencetak elemen array dengan indeks 3. Karena indeks array yang kita akses melebihi batas yang diizinkan, maka akan terjadi error ArrayIndexOutOfBoundsException. Untuk menangani error tersebut, kita juga telah menambahkan blok catch yang akan menangani jenis exception

ArrayIndexOutOfBoundsException. Di dalam blok catch tersebut, program akan mencetak pesan "Melebihi jumlah array".

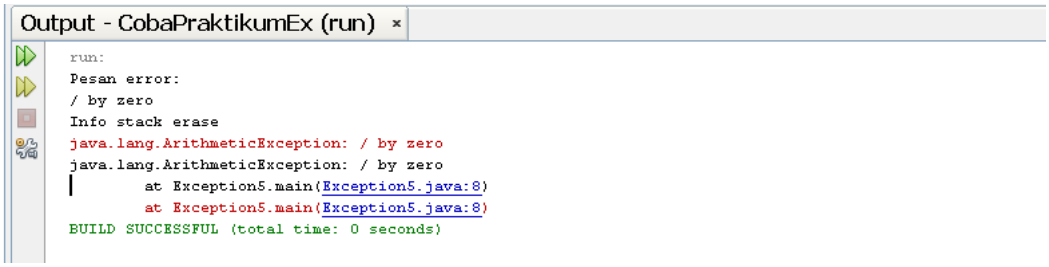
Setelah error tersebut dihandle, program akan berjalan dengan benar. Jika kita menjalankan program tersebut, maka akan tercetak pesan "Terjadi Aritmatika error" kemudian "Melebihi jumlah array". Namun, jika kita mengubah nilai yang akan dibagi dengan nol menjadi bilangan yang lain, misalnya 5 atau 7, dan juga mengubah indeks array yang akan dicetak menjadi indeks yang valid, maka program akan berjalan dengan benar dan mencetak hasil pembagian dan elemen array sesuai dengan yang diinginkan.

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");
 }
 }
}
```

Output :



Analisa :

Program diatas akan mencoba melakukan operasi pembagian dengan bilangan nol (0). Operasi pembagian dengan nol merupakan operasi yang tidak valid dalam matematika, sehingga akan terjadi error pada saat operasi tersebut dijalankan.

Untuk menangani error tersebut, kita telah menambahkan blok try-catch pada program tersebut. Di dalam blok try, kita menuliskan kode yang berpotensi menyebabkan error. Di dalam blok catch, kita menuliskan kode yang akan dijalankan jika error terjadi. Pada program ini, kita telah menambahkan dua blok catch.

Blok catch pertama akan menangani jenis exception ArithmeticException, yang merupakan exception yang terjadi saat terjadi kesalahan aritmatika seperti pembagian dengan nol. Di dalam blok catch tersebut, program akan mencetak pesan "Pesan error: " kemudian mencetak pesan error yang terjadi menggunakan method getMessage(). Selanjutnya, program juga akan mencetak pesan "Info stack erase" kemudian mencetak informasi mengenai stack trace menggunakan method printStackTrace() dan printStackTrace(System.out).

Blok catch kedua akan menangani jenis exception Exception, yang merupakan exception yang terjadi saat terjadi kesalahan umum yang tidak terprediksi. Di dalam blok catch tersebut, program akan mencetak pesan "Ini menghandle error yang terjadi".

Jika kita menjalankan program tersebut, maka akan tercetak pesan "Pesan error: " kemudian pesan error yang terjadi, "Info stack erase" kemudian informasi stack trace mengenai error yang terjadi. Hal ini terjadi karena blok catch pertama akan menangani jenis exception ArithmeticException yang terjadi saat terjadi pembagian dengan nol.

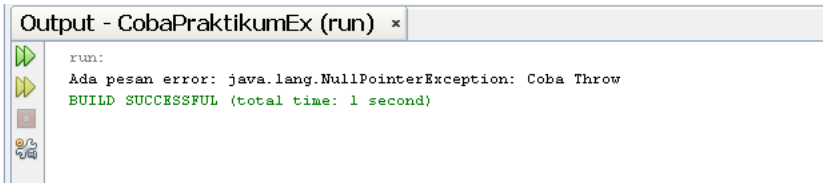
Namun, jika kita mengubah nilai yang akan dibagi dengan nol menjadi bilangan yang lain, misalnya 5 atau 7, maka program akan berjalan dengan benar dan mencetak hasil pembagian sesuai dengan yang diinginkan. Dalam kasus tersebut, blok catch pertama tidak akan dijalankan karena tidak ada exception ArithmeticException yang terjadi, sehingga blok catch kedua yang menangani exception Exception juga tidak akan dijalankan.

Percobaan 6 :

```
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);
 }
 }
}
```

Output :



Analisa :

Dalam program di atas, ada penggunaan try-catch untuk menangani exception yang mungkin terjadi. Exception adalah suatu kondisi yang tidak diharapkan atau tidak diinginkan yang terjadi saat program sedang berjalan. Exception dapat mengakibatkan program berhenti, jadi penting untuk menangani exception agar program dapat terus berjalan dengan baik.

Try-catch merupakan cara untuk menangani exception yang terjadi pada blok kode yang ditulis dalam blok try. Jika exception terjadi pada blok try, maka eksekusi akan dilanjutkan ke blok catch yang sesuai dengan tipe exception yang terjadi.

Dalam program di atas, terdapat blok try-catch dengan blok try yang mengeksekusi metode demo(). Jika exception NullPointerException terjadi pada blok try, maka eksekusi akan dilanjutkan ke blok catch yang menangani exception NullPointerException. Blok catch akan menampilkan pesan error yang terjadi.

Metode demo() menciptakan instance dari kelas NullPointerException dengan pesan "Coba Throw" dan menggunakan kata kunci throw untuk membuang exception tersebut. Ini akan mengakibatkan eksekusi metode demo() terhenti dan dilanjutkan ke blok catch pada blok try-catch di metode main().

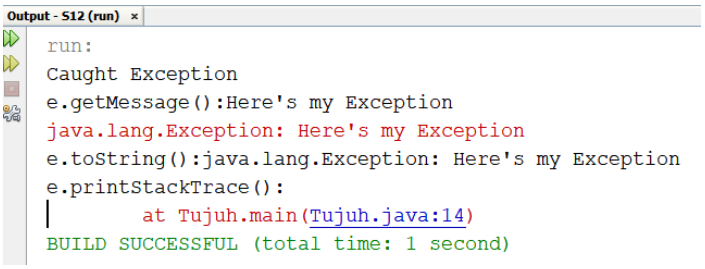
Jadi, penggunaan try-catch pada program di atas bertujuan untuk menangani exception NullPointerException yang mungkin terjadi pada blok try dan menampilkan pesan error yang terjadi.

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();
 }
 }
}
```

Output :



Analisa :

Dalam program di atas, blok try-catch digunakan untuk menangkap exception yang mungkin terjadi saat menjalankan perintah throw. Perintah throw menciptakan objek exception baru dengan pesan yang ditentukan ("Here's my Exception") dan melemparkannya. Jika exception tersebut terjadi, maka eksekusi akan langsung diteruskan ke blok catch yang sesuai dengan tipe exception yang ditangkap. Pada contoh ini, jika Exception terjadi, maka eksekusi akan langsung diteruskan ke blok catch yang menangani exception bertipe Exception.

Ketika perintah throw dijalankan, exception akan terjadi dan eksekusi akan langsung diteruskan ke blok catch. Pada blok catch tersebut, pesan error akan dicetak dengan menggunakan method getMessage(), toString(), dan printStackTrace(). Method getMessage() akan menampilkan pesan yang ditentukan dalam objek exception, method toString() akan menampilkan pesan error lengkap termasuk nama kelas exception, dan method printStackTrace() akan menampilkan trace stack yang menunjukkan urutan pemanggilan method yang menyebabkan exception terjadi.

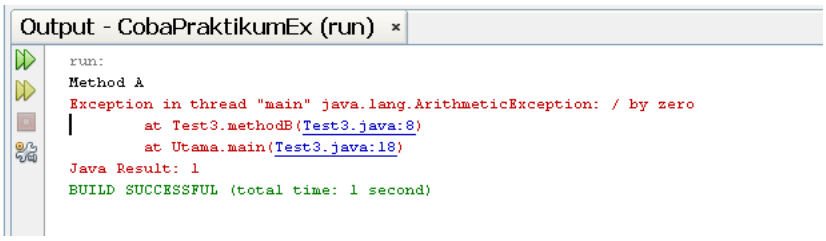
Jika tidak ada blok catch yang sesuai dengan tipe exception yang terjadi, maka exception akan diteruskan ke tingkatan yang lebih tinggi (seperti main method atau method lain yang memanggil method tersebut). Jika tidak ada blok catch yang menangani exception tersebut, maka program akan terhenti dan menampilkan pesan error.

Percobaan 8 :

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

```
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");
 }
}
10
class Utama
{
 public static void main(String[] args) throws IOException
 {
 Test3 c=new Test3();
 c.methodA();
 c.methodB();
 }
}
```

Output :



Pada baris pertama, akan dicetak "Method A" karena pemanggilan metode methodA() tidak menghasilkan exception. Namun, pada baris kedua, akan terjadi exception dengan tipe ArithmeticException karena terdapat pembagian dengan nol (20/0). Exception ini akan terjadi jika kode yang menyebabkan exception tersebut dijalankan, dan jika tidak ditangani dengan baik, akan menyebabkan program terhenti.

Jadi, output yang akan ditampilkan hanya akan sampai pada baris pertama, yaitu "Method A". Baris kedua tidak akan tercetak karena exception terjadi sebelumnya.

Analisa :

Pada program diatas, keyword "throws" digunakan dalam metode methodB() dan main(). Keyword "throws" digunakan untuk memberitahukan bahwa metode tersebut mungkin dapat menghasilkan exception (kesalahan) yang harus ditangani oleh pemanggil metode tersebut.

Exception yang mungkin terjadi dalam metode methodB() adalah IOException, yang merupakan exception yang terjadi saat terjadi masalah dalam operasi Input/Output. Exception ini mungkin terjadi karena adanya kesalahan dalam membaca atau menulis file, atau masalah jaringan yang terjadi saat mengirim atau menerima data.

Dalam contoh diatas, exception yang mungkin terjadi adalah ArithmeticException yang terjadi saat terjadi pembagian dengan nol (20/0). Exception ini akan terjadi jika kode yang menyebabkan exception tersebut dijalankan, dan jika tidak ditangani dengan baik, akan menyebabkan program terhenti.

Sementara itu, keyword "throws" juga digunakan dalam metode main(), yang memberitahukan bahwa metode tersebut mungkin dapat menghasilkan exception yang harus ditangani oleh pemanggil metode tersebut. Exception yang mungkin terjadi dalam metode main() adalah IOException yang terjadi saat terjadi masalah dalam operasi Input/Output.

Jadi, penggunaan keyword "throws" dalam program diatas adalah untuk memberitahukan bahwa metode methodB() dan main() mungkin dapat menghasilkan exception yang harus ditangani oleh pemanggil metode tersebut. Exception tersebut harus ditangani dengan cara menangkap exception tersebut dengan menggunakan pernyataan try-catch, atau dengan mengalihkan tanggung jawab penanganan exception ke pemanggil metode tersebut dengan menggunakan keyword "throws" pada deklarasi metode tersebut.

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

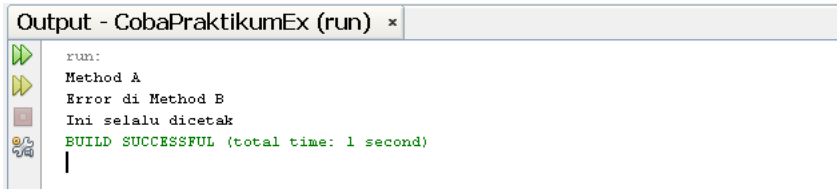
```
class Utama
{
 public static void main(String[] args)
 {
 Test3 o=new Test3();
 o.methodA();
 try
 {
 o.methodB();
 }
 catch(Exception e)
 {
 System.out.println("Error di Method B");
 }
 finally
 {
 System.out.println("Ini selalu dicetak");
 };
 }
}
```

Program yang sudah diubah :

Source History

```
1  import java.io.*;
2  public class Test3 {
3   public void methodA(){
4   System.out.println("Method A");
5   }
6   public void methodB() throws IOException
7   {
8   System.out.println(20/0);
9   System.out.println("Method B");
10  }
11 }
12 class Utama
13 {
14  public static void main(String[] args)
15  {
16  Test3 o=new Test3();
17  o.methodA();
18  try
19  {
20  o.methodB();
21  }
22  catch(Exception e)
23  {
24  System.out.println("Error di Method B");
25  }
26  finally
27  {
28  System.out.println("Ini selalu dicetak");
    };
30  }
31 }
```

Output :



Analisa :

Jika program diatas diubah dan dijalankan, maka akan terjadi perubahan dalam penanganan exception yang terjadi pada metode methodB(). Sebelumnya, exception yang terjadi pada metode methodB() tidak ditangani dan menyebabkan program terhenti. Namun setelah diubah, exception yang terjadi akan ditangani dengan menggunakan pernyataan try-catch.

Pada baris pertama, akan dicetak "Method A" karena pemanggilan metode methodA() tidak menghasilkan exception. Pada baris kedua, exception yang terjadi pada metode methodB() akan ditangani dengan mencetak "Error di Method B". Kemudian, pada baris ketiga akan dicetak "Ini selalu dicetak" karena ini merupakan bagian dari pernyataan finally yang selalu dijalankan setelah pernyataan try-catch selesai dijalankan. Jadi, dengan menggunakan pernyataan try-catch, exception yang terjadi pada metode methodB() akan ditangani dengan cara mencetak pesan error dan program tidak terhenti. Selain itu, pernyataan finally akan selalu dijalankan setelah pernyataan try-catch selesai dijalankan, baik exception terjadi atau tidak.

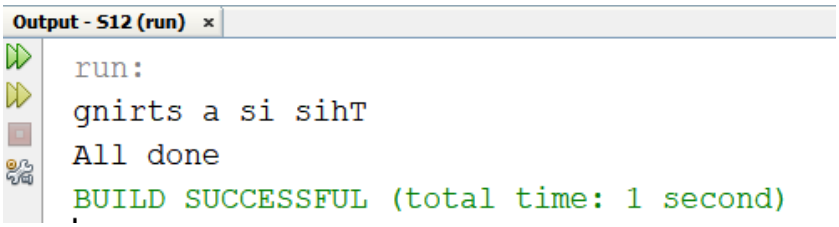
Percobaan 9 :

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

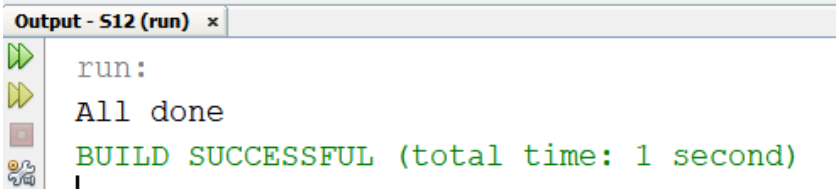
```
class Propagate {

 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)
11
 {
 throw new Exception();
 }
```

```
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;
 }
}
```

Output Pertama :



Output Kedua (Soal) :



Analisa :

Program ini akan menampilkan string yang diberikan dalam keadaan terbalik (reverse). Jika panjang string s adalah 0, maka akan terjadi exception yang akan ditangkap oleh blok catch. Pada blok try tersebut, apabila di hapus maka akan diteruskan ke blok catch. Pada blok catch tersebut, akan dicetak pesan "The String was blank". Setelah blok catch dijalankan, maka eksekusi akan langsung diteruskan ke blok finally yang mencetak pesan "All done".

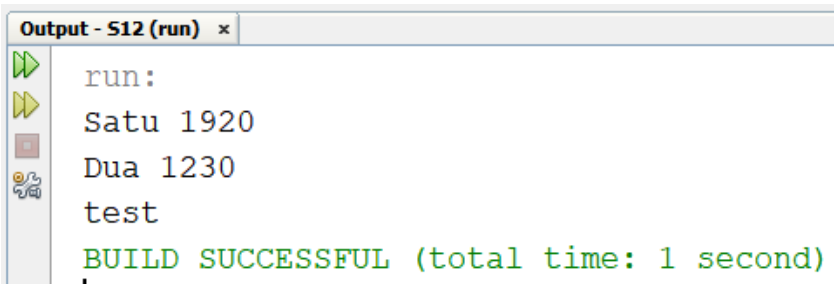
Jika tidak ada blok catch yang sesuai dengan tipe exception yang terjadi, maka exception akan diteruskan ke tingkatan yang lebih tinggi (seperti main method atau method lain yang memanggil method tersebut). Jika tidak ada blok catch yang menangani exception tersebut, maka program akan terhenti dan menampilkan pesan error.

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");
 }
} 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;
 }
}
```

Output :



Analisa :

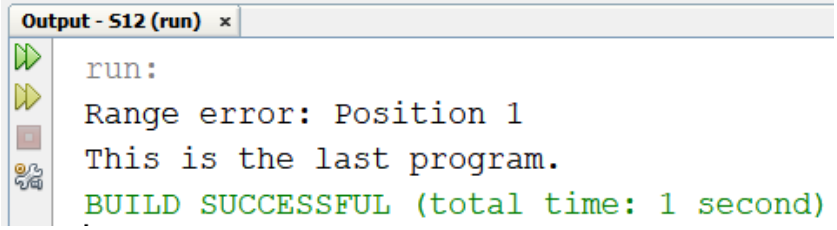
Dalam program di atas, blok try-catch digunakan untuk menangkap exception yang mungkin terjadi saat membuat objek RandomAccessFile. Exception yang mungkin terjadi adalah IOException, yang dapat terjadi jika terjadi masalah saat membaca atau menulis file.

Jika exception tersebut terjadi, maka eksekusi akan langsung diteruskan ke blok catch yang sesuai dengan tipe exception yang ditangkap. Pada contoh ini, jika IOException terjadi, maka eksekusi akan langsung diteruskan ke blok catch yang menangani exception bertipe IOException. Pada blok catch tersebut, akan dicetak pesan "Indeks melebihi batas".

Jika tidak ada blok catch yang sesuai dengan tipe exception yang terjadi, maka exception akan diteruskan ke tingkatan yang lebih tinggi (seperti main method atau method lain yang memanggil method tersebut). Jika tidak ada blok catch yang menangani exception tersebut, maka program akan terhenti dan menampilkan pesan error.

```
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.");
 }
}
```

Output :



Analisa :

Dalam program di atas, kelas RangeErrorException merupakan kelas yang merupakan turunan dari kelas Throwable. Kelas ini memiliki constructor yang menerima parameter string yang akan disimpan sebagai pesan error dalam objek exception.

Dalam method main, blok try-catch digunakan untuk menangkap exception yang mungkin terjadi saat melakukan percabangan if. Jika posisi (position) lebih besar dari 0, maka akan terjadi exception dengan pesan error "Position n" (dimana n adalah nilai dari posisi). Exception tersebut akan ditangkap oleh blok catch yang menangani exception bertipe RangeErrorException. Pada blok catch tersebut, pesan error akan dicetak dengan menggunakan method getMessage() dari objek exception yang ditangkap.

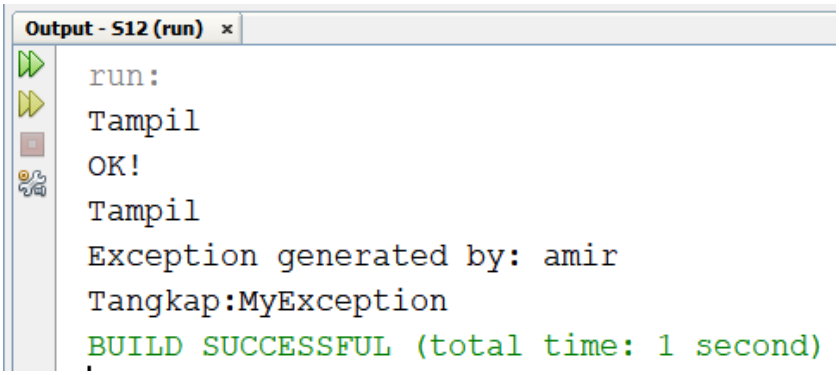
Jika tidak ada blok catch yang sesuai dengan tipe exception yang terjadi, maka exception akan diteruskan ke tingkatan yang lebih tinggi (seperti main method atau method lain yang memanggil method tersebut). Jika tidak ada blok catch yang menangani exception tersebut, maka program akan terhenti dan menampilkan pesan error.

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("ali");
 tampil("amir");
 }
 catch(MyException ex)
 {
 System.out.println("Tangkap:"+ex);
 }
 }
}
```

Output :



Analisa :

Dalam program di atas, kelas MyException merupakan kelas yang merupakan turunan dari kelas Exception. Kelas ini memiliki constructor yang menerima parameter string yang akan disimpan sebagai pesan error dalam objek exception. Saat objek exception dibuat, pesan error juga akan dicetak ke layar.

Method tampil() mengandung pernyataan throw yang menciptakan objek exception bertipe MyException jika string yang diberikan bernilai "amir". Jika tidak, maka akan dicetak pesan "OK!".

Dalam method main, blok try-catch digunakan untuk menangkap exception yang mungkin terjadi saat memanggil method tampil(). Jika exception terjadi, maka akan ditangkap oleh blok catch yang menangani exception bertipe MyException. Pada blok catch tersebut, pesan error akan dicetak dengan menggunakan toString() dari objek exception yang ditangkap. Jika tidak ada blok catch yang sesuai dengan tipe exception yang terjadi, maka exception akan diteruskan ke tingkatan yang lebih tinggi (seperti main method atau method lain yang memanggil method tersebut). Jika tidak ada blok catch yang menangani exception tersebut, maka program akan terhenti dan menampilkan pesan error.