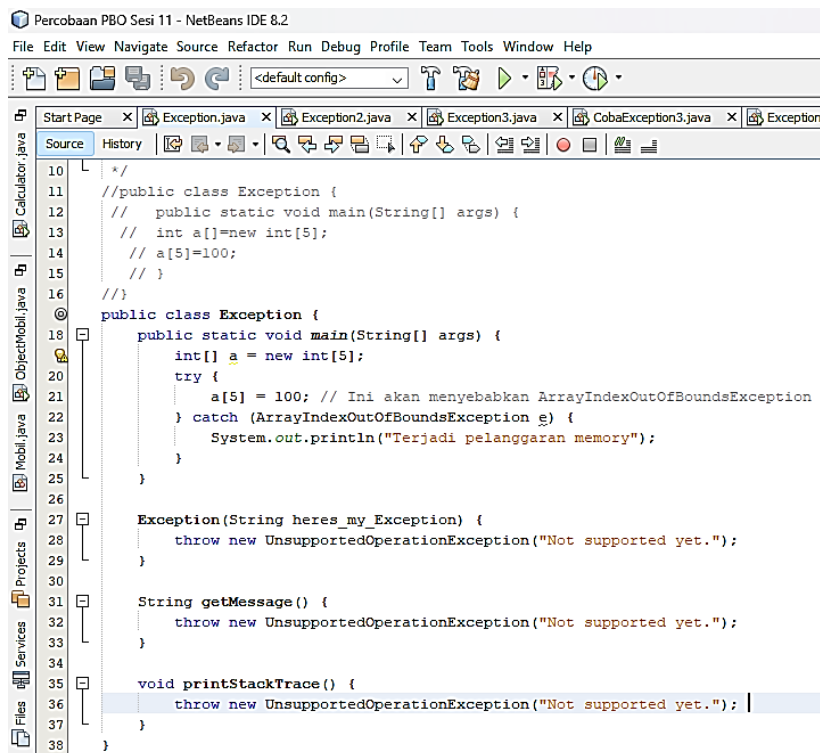


Nama : Rahayu Amaliyah Kamis
NIM : 20220040248
Kelas : TI22A
Mata Kuliah : Pemrograman Berorientasi Object
Tugas Sesi : Alun Sujjada, S.Kom, M.T

Rincian Jawaban :

1. Percobaan 1



```
10  /*
11  //public class Exception {
12  //  public static void main(String[] args) {
13  //    int a[]=new int[5];
14  //    a[5]=100;
15  //  }
16  //}
17
18  public class Exception {
19      public static void main(String[] args) {
20          int[] a = new int[5];
21          try {
22              a[5] = 100; // Ini akan menyebabkan ArrayIndexOutOfBoundsException
23          } catch (ArrayIndexOutOfBoundsException e) {
24              System.out.println("Terjadi pelanggaran memory");
25          }
26      }
27
28      Exception(String heres_my_Exception) {
29          throw new UnsupportedOperationException("Not supported yet.");
30      }
31
32      String getMessage() {
33          throw new UnsupportedOperationException("Not supported yet.");
34      }
35
36      void printStackTrace() {
37          throw new UnsupportedOperationException("Not supported yet.");
38      }
39  }
```

Setelah diamati, pada program awal, terdapat kesalahan dalam mengakses elemen array. Baris `a[5] = 100;` mencoba mengakses indeks ke-5 dari array `a`, padahal indeks valid hanya dari 0 hingga 4. Oleh karena itu, program menghasilkan `ArrayIndexOutOfBoundsException`. Pada program yang diperbaiki, kita menggunakan blok `try-catch` untuk menangani kesalahan tersebut dan memberikan pesan yang sesuai.

2. Percobaan 2

Percobaan PBO Sesi 11 - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

<default config>

Start Page X Exception.java X Exception2.java X Exception3.java X CobaException3.jav

Source History

```
19 //};
20 //while(i<4)
21 // {
22 //     System.out.println(greeting[i]);
23 //     i++;
24 // }
25 // }
26 //}
27
28 //perbaiki codingan
29 public class Exception2 {
30     public static void main(String[] args) {
31         int i=0;
32         String greetings[]={
33             "Hello World!",
34             "No,I mean it!",
35             "HELLO WORLD!"
36         };
37         while(i<4)
38         {
39             try{
40                 System.out.println(greetings[i]);
41                 i++;
42             }
43             catch(ArrayIndexOutOfBoundsException e)
44             {
45                 System.out.println("Resetting index value");
46                 i=0;
47             }
48         }
49     }
```

3. Percobaan 3

Percobaan PBO Sesi 11 - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

<default config>

Start Page X Exception.java X Exception2.java X Exception3.java X CobaException3.java X Exception4.java X

Source History

```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6
7 /**
8  *
9  * @author ASUS
10  */
11 //public class Exception3 {
12 //    public static void main(String[] args) {
13 //        int bil=10;
14 //        System.out.println(bil/0);
15 //    }
16 //}
17 public class Exception3 {
18     public static void main(String[] args) {
19         int bil = 10;
20         try {
21             System.out.println(bil / 0);
22         } catch (ArithmeticException e) {
23             System.out.println("Terjadi kesalahan aritmatika: pembagian dengan nol.");
24         }
25     }
26 }
```

```

1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6
7  /**
8   *
9   * @author ASUS
10  */
11  public class CobaException3 {
12      public static void main(String[] args) {
13          int bil = 10;
14          try {
15              System.out.println(bil / 0);
16          } catch (ArithmeticException e) {
17              System.out.println("Terjadi Aritmatika error: pembagian dengan nol.");
18          }
19      }
20  }

```

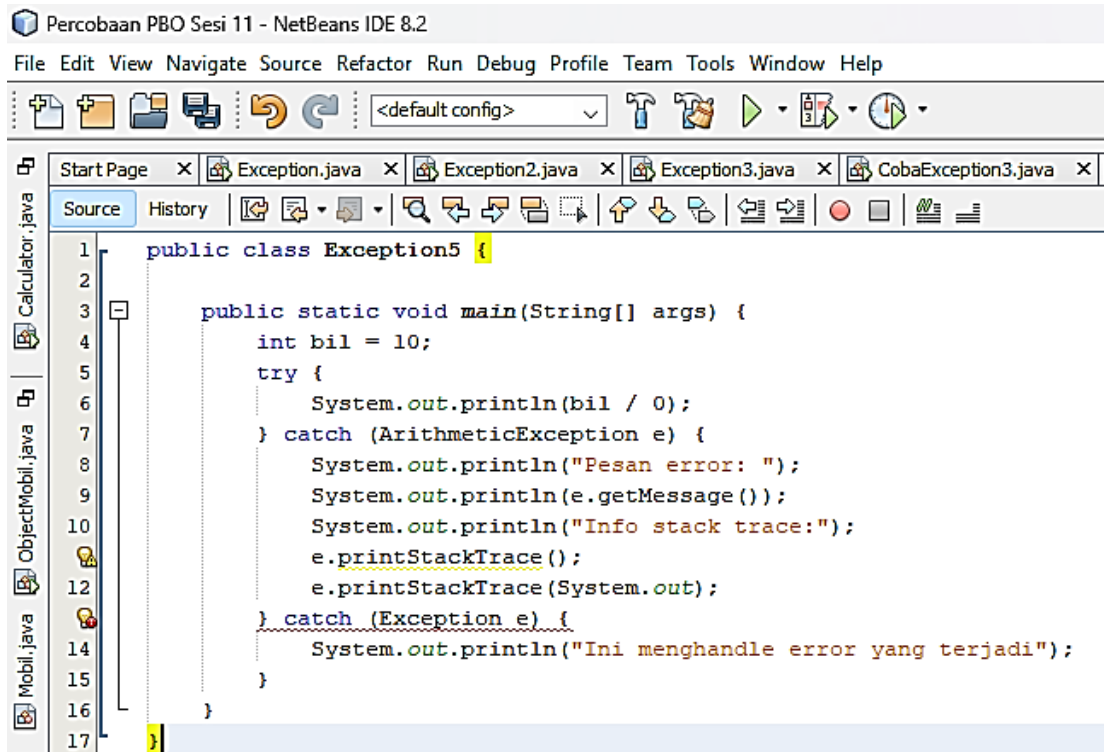
4. Percobaan 4

```

19  // }
20  //catch(ArithmeticException e)
21  //{
22  //    System.out.println("Terjadi Aritmatika error");
23  //}
24  //catch(ArrayIndexOutOfBoundsException e)
25  //{
26  //    System.out.println("Melebihi jumlah array");
27  //}
28  //catch(Exception e)
29  //{
30  //    System.out.println("Ini menghandle error yang terjadi");
31  //}
32  //}
33  //}
34
35  public class Exception4 {
36      public static void main(String[] args) {
37          int bil = 10;
38          String b[] = {"a", "b", "c"};
39
40          try {
41              System.out.println(bil / 0);
42              System.out.println(b[3]);
43          } catch (ArithmeticException e) {
44              System.out.println("Terjadi Aritmatika error");
45          } catch (ArrayIndexOutOfBoundsException e) {
46              System.out.println("Melebihi jumlah array");
47          }
48      }
49  }

```

5. Percobaan 5



```
1 public class Exception5 {
2
3     public static void main(String[] args) {
4         int bil = 10;
5         try {
6             System.out.println(bil / 0);
7         } catch (ArithmeticException e) {
8             System.out.println("Pesan error: ");
9             System.out.println(e.getMessage());
10            System.out.println("Info stack trace:");
11            e.printStackTrace();
12            e.printStackTrace(System.out);
13        } catch (Exception e) {
14            System.out.println("Ini menhandle error yang terjadi");
15        }
16    }
17 }
```

Analisis Penggunaan try dan catch:

1. Blok try:

- try berisi kode yang mungkin menyebabkan pengecualian. Dalam kasus ini, `bil / 0` menyebabkan kesalahan pembagian dengan nol (`ArithmeticException`).

2. Blok catch untuk `ArithmeticException`:

- Menangkap kesalahan pembagian dengan nol.
- Mencetak pesan error dan informasi detail tentang pengecualian.
- `e.printStackTrace()` mencetak detail pengecualian, membantu dalam debugging.

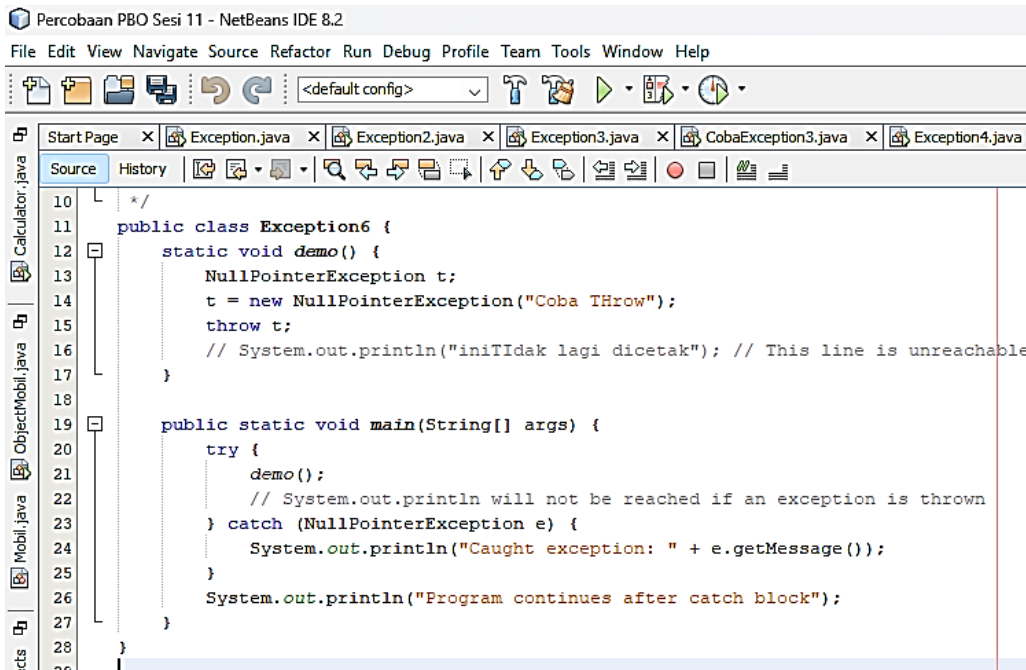
3. Blok catch untuk `Exception`:

- Menangkap semua jenis pengecualian lainnya yang mungkin terjadi.
- Dalam program ini, blok ini tidak akan dijalankan karena pengecualian `ArithmeticException` sudah ditangani sebelumnya.

Kesimpulan:

- `try` digunakan untuk menjalankan kode yang mungkin menyebabkan pengecualian.
- `catch` menangani pengecualian yang terjadi, memberikan informasi tentang kesalahan tersebut.

6. Percobaan 6



```
10  */
11  public class Exception6 {
12      static void demo() {
13          NullPointerException t;
14          t = new NullPointerException("Coba THrow");
15          throw t;
16          // System.out.println("iniTidak lagi dicetak"); // This line is unreachable
17      }
18
19      public static void main(String[] args) {
20          try {
21              demo();
22              // System.out.println will not be reached if an exception is thrown
23          } catch (NullPointerException e) {
24              System.out.println("Caught exception: " + e.getMessage());
25          }
26          System.out.println("Program continues after catch block");
27      }
28  }
```

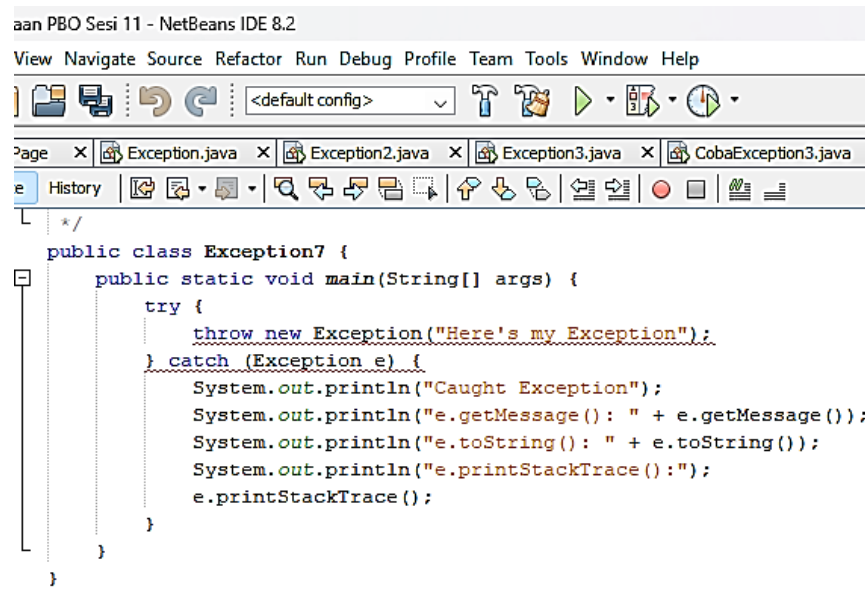
Analisis Program:

1. **Metode demo ():**
 - o Membuat dan melempar `NullPointerException` dengan pesan "Coba THrow".
 - o Baris setelah `throw t;` tidak akan dieksekusi.
2. **Metode main (String[] args):**
 - o **Blok try:**
 - Memanggil `demo ()` yang melempar pengecualian.
 - Kode setelah `demo ()` dalam blok `try` tidak akan dijalankan jika pengecualian dilempar.
 - o **Blok catch untuk NullPointerException:**
 - Menangkap `NullPointerException`.
 - Mencetak pesan "Caught exception: Coba THrow".
 - o **Setelah Blok catch:**
 - Mencetak "Program continues after catch block".

Kesimpulan:

- `try` digunakan untuk menjalankan kode yang mungkin melempar pengecualian.
- `catch` menangani pengecualian yang dilempar.
- Program berlanjut setelah pengecualian ditangani, memastikan program tidak berhenti tiba-tiba.

7. Percobaan 7



The screenshot shows the NetBeans IDE 8.2 interface. The title bar reads "aan PBO Sesi 11 - NetBeans IDE 8.2". The menu bar includes View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The toolbar contains icons for file operations, running, and debugging. The project explorer on the left shows a package structure with a file named "Exception7.java". The main editor window displays the following Java code:

```
public class Exception7 {  
    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();  
        }  
    }  
}
```

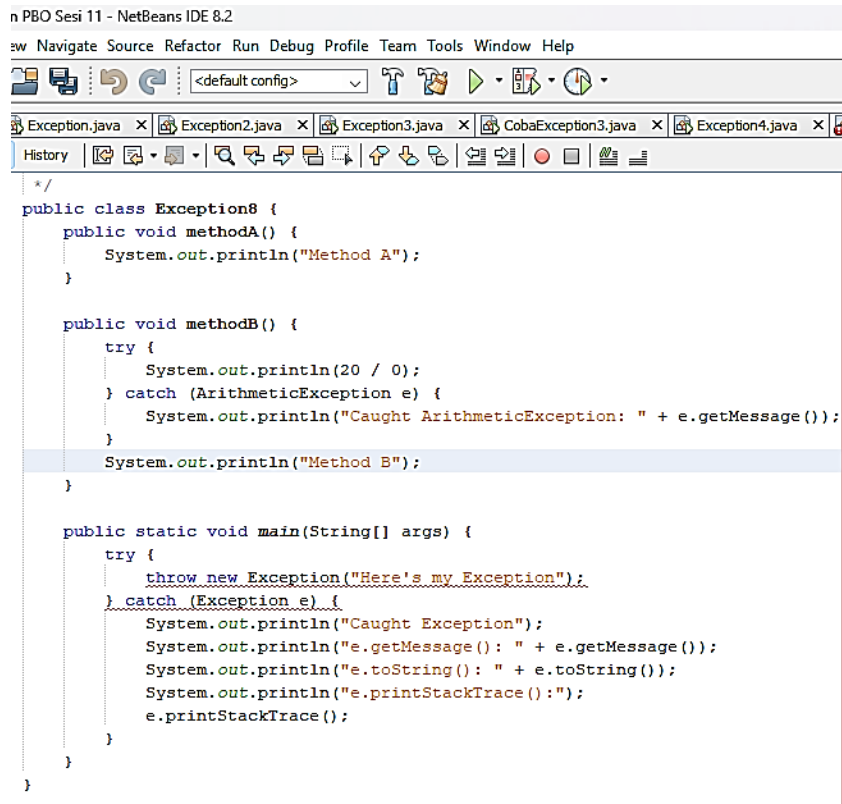
Analisis Program:

1. **Blok try:**
 - o Melempar `Exception` dengan pesan "Here's my Exception".
2. **Blok catch untuk Exception:**
 - o Menangkap pengecualian yang dilempar.
 - o Mencetak "Caught Exception".
 - o Mencetak pesan dari pengecualian menggunakan `e.getMessage()`.
 - o Mencetak representasi string dari pengecualian menggunakan `e.toString()`.
 - o Mencetak stack trace dari pengecualian menggunakan `e.printStackTrace()`.

Kesimpulan:

- `try` digunakan untuk menjalankan kode yang mungkin melempar pengecualian.
- `catch` menangani pengecualian dan mencetak informasi tentang pengecualian tersebut.
- Program menangkap dan menampilkan detail pengecualian, lalu melanjutkan eksekusi.

8. Percobaan 8



```

n PBO Sesi 11 - NetBeans IDE 8.2
aw Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Exception.java x Exception2.java x Exception3.java x CobaException3.java x Exception4.java x
History
*/
public class Exception8 {
    public void methodA() {
        System.out.println("Method A");
    }

    public void methodB() {
        try {
            System.out.println(20 / 0);
        } catch (ArithmeticException e) {
            System.out.println("Caught ArithmeticException: " + e.getMessage());
        }
        System.out.println("Method B");
    }

    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();
        }
    }
}

```

Analisis Program:

1. **Metode methodA () :**
 - o Menampilkan pesan "Method A".
2. **Metode methodB () :**
 - o **Blok try:**
 - Mencoba melakukan operasi `20 / 0`, yang menyebabkan `ArithmeticException` karena pembagian dengan nol.
 - o **Blok catch untuk ArithmeticException:**
 - Menangkap `ArithmeticException` dan mencetak pesan "Caught ArithmeticException: / by zero".
 - o Setelah blok `catch`, mencetak "Method B".
3. **Metode main (String[] args) :**
 - o **Blok try:**
 - Melempar `Exception` dengan pesan "Here's my Exception".
 - o **Blok catch untuk Exception:**
 - Menangkap pengecualian dan mencetak beberapa informasi:
 - "Caught Exception".
 - Pesan dari pengecualian menggunakan `e.getMessage()`.
 - Representasi string dari pengecualian menggunakan `e.toString()`.

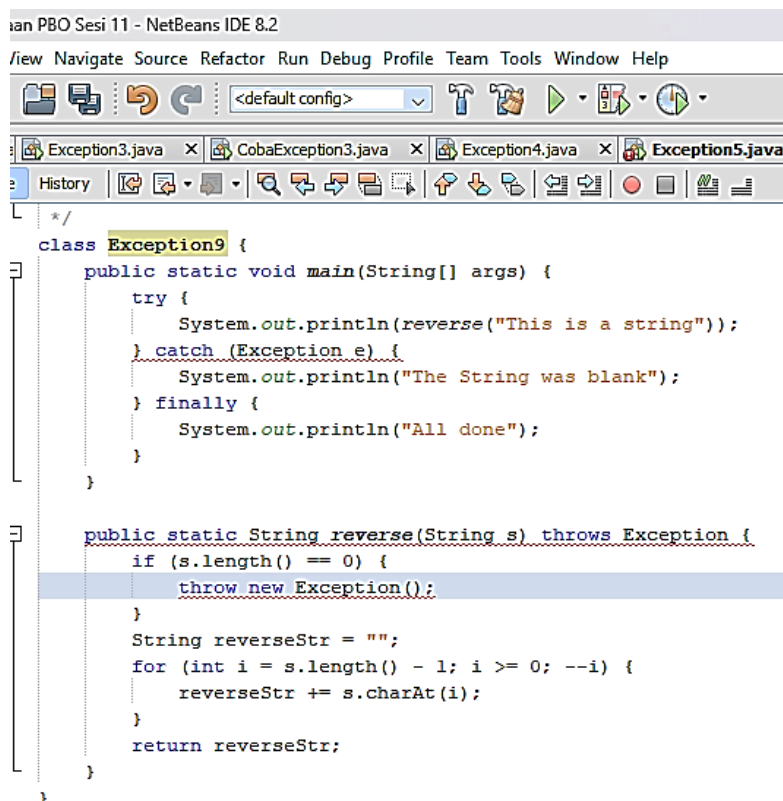
- Stack trace dari pengecualian menggunakan `e.printStackTrace()`.

Kesimpulan Penggunaan throws:

- **Blok try dan catch di methodB():**
 - Menangani `ArithmeticException` yang terjadi saat pembagian dengan nol.
 - Menangkap dan menangani pengecualian sehingga program tidak berhenti tiba-tiba.
 - Setelah menangkap pengecualian, program melanjutkan eksekusi dan mencetak "Method B".
- **Blok try dan catch di main(String[] args):**
 - Melempar dan menangkap `Exception`.
 - Menangkap pengecualian umum, mencetak informasi tentang pengecualian, dan melanjutkan eksekusi program.

Dalam program ini, pengecualian yang terjadi di dalam `methodB()` dan di dalam `main(String[] args)` ditangani dengan baik oleh blok `catch`, sehingga program tetap dapat melanjutkan eksekusi setelah menangkap pengecualian. Tidak ada penggunaan kata kunci `throws` dalam definisi metode atau konstruktor, sehingga analisis penggunaan `throws` tidak diperlukan untuk program ini.

9. Percobaan 9



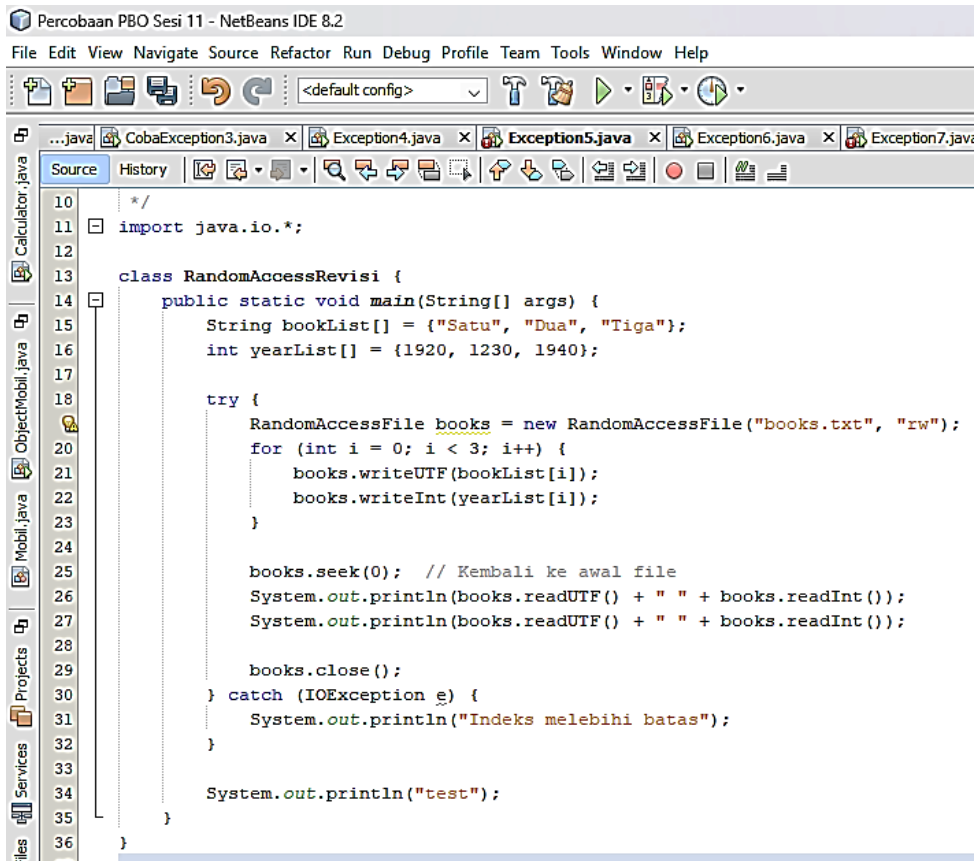
```

/*
class Exception9 {
    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;
    }
}

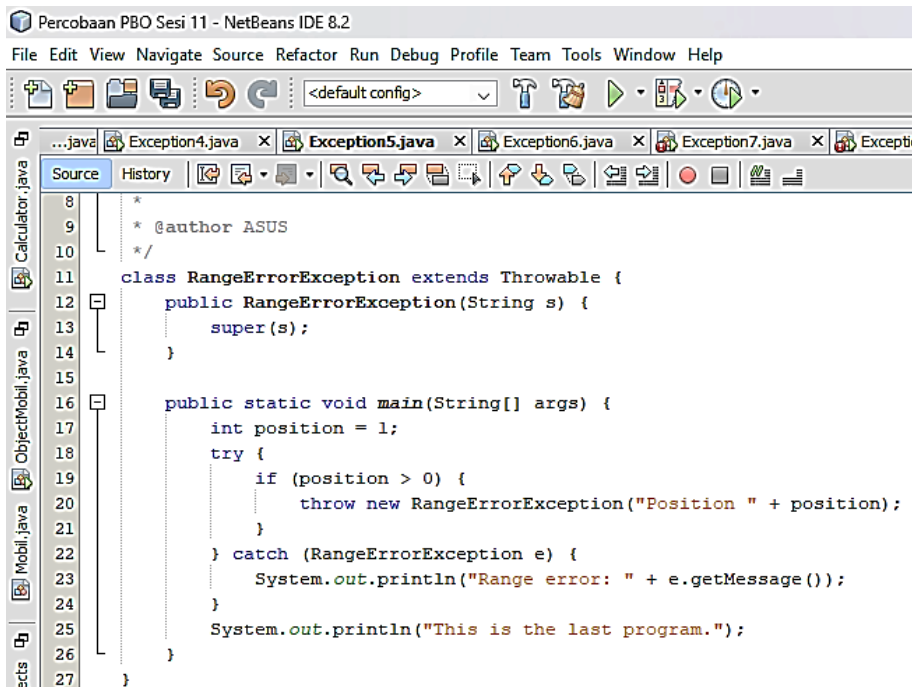
```


10. Percobaan 10



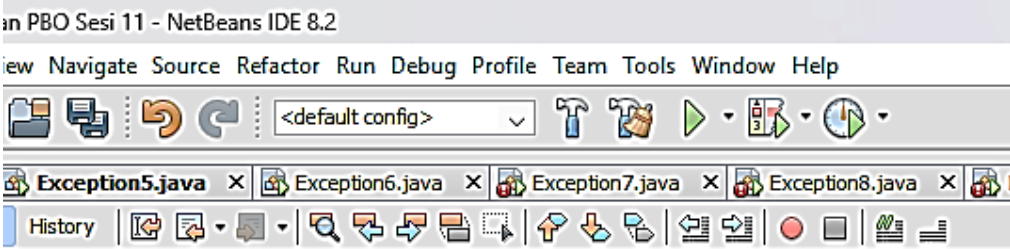
```
10  /*
11  import java.io.*;
12
13  class RandomAccessRevisi {
14      public static void main(String[] args) {
15          String bookList[] = {"Satu", "Dua", "Tiga"};
16          int yearList[] = {1920, 1230, 1940};
17
18          try {
19              RandomAccessFile books = new RandomAccessFile("books.txt", "rw");
20              for (int i = 0; i < 3; i++) {
21                  books.writeUTF(bookList[i]);
22                  books.writeInt(yearList[i]);
23              }
24
25              books.seek(0); // Kembali ke awal file
26              System.out.println(books.readUTF() + " " + books.readInt());
27              System.out.println(books.readUTF() + " " + books.readInt());
28
29              books.close();
30          } catch (IOException e) {
31              System.out.println("Indeks melebihi batas");
32          }
33
34          System.out.println("test");
35      }
36  }
```

11. Percobaan 11



```
8  /*
9  * @author ASUS
10  */
11  class RangeErrorException extends Throwable {
12      public RangeErrorException(String s) {
13          super(s);
14      }
15
16      public static void main(String[] args) {
17          int position = 1;
18          try {
19              if (position > 0) {
20                  throw new RangeErrorException("Position " + position);
21              }
22          } catch (RangeErrorException e) {
23              System.out.println("Range error: " + e.getMessage());
24          }
25          System.out.println("This is the last program.");
26      }
27  }
```

12. Percobaan 12



The screenshot shows the NetBeans IDE 8.2 interface. The title bar reads "an PBO Sesi 11 - NetBeans IDE 8.2". The menu bar includes "File", "Edit", "View", "Navigate", "Source", "Refactor", "Run", "Debug", "Profile", "Team", "Tools", "Window", and "Help". The toolbar contains icons for file operations, running, and debugging. The tabs bar shows four open files: "Exception5.java", "Exception6.java", "Exception7.java", and "Exception8.java". The "Exception5.java" tab is active, displaying the following code:

```
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);
        }
    }
}
```