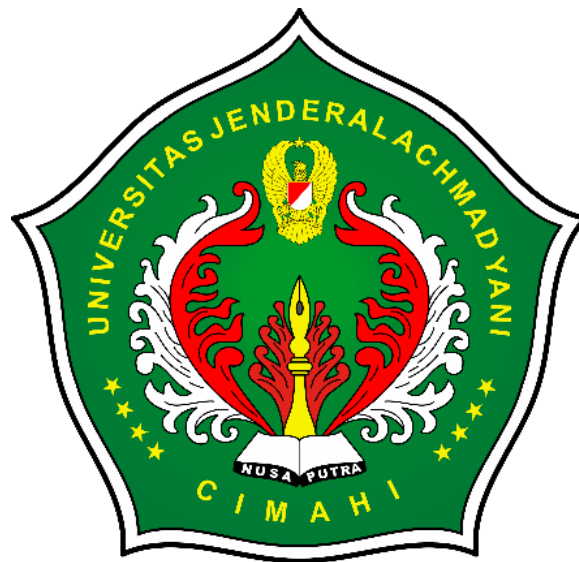


**LAPORAN PRAKTIKUM  
PEMROGRAMAN OBJEK 2**

**MODUL 2  
Exception Handling**

**DISUSUN OLEH :  
Fathir Ahmad Nurpadli - 2250081132**



**PROGRAM STUDI INFORMATIKA  
FAKULTAS SAINS DAN INFORMATIKA  
UNIVERSITAS JENDERAL ACHMAD YANI  
TAHUN 2024**



# DAFTAR ISI

DAFTAR ISI.....	i
DAFTAR GAMBAR.....	iv
BAB I. HASIL PRAKTIKUM.....	1
I.1 Program 2-1 MemoryException.java .....	1
I.1.A. Source Code .....	1
I.1.B. Hasil .....	1
I.1.C. Analisa .....	1
I.2 Program 2-2 MemoryException1.java .....	2
I.2.A. Source Code .....	2
I.2.B. Hasil .....	2
I.2.C. Analisa .....	2
I.3 Program 2-3 DivideByZeroNoExceptionHandling.java .....	3
I.3.A. Source Code .....	3
I.3.B. Hasil .....	4
I.3.C. Analisa .....	4
I.4 Program 2-4 DivideByZeroWithExceptionHandling.java .....	4
I.4.A. Source Code .....	4
I.4.B. Hasil .....	6
I.4.C. Analisa .....	6
I.5 Program 2-5 NullReference.java .....	6
I.5.A. Source Code .....	6
I.5.B. Hasil .....	8
I.5.C. Analisa .....	8
I.6 Program 2-6 Try-catch-finally .....	8
I.6.A. Source Code .....	8

I.6.B.	Hasil .....	11
I.6.C.	Analisa .....	12
I.7	Program 2-7 WrongCasting.java.....	12
I.7.A.	Source Code .....	12
I.7.B.	Hasil .....	13
I.7.C.	Analisa .....	13
I.8	Program 2-8 Account.java.....	14
I.8.A.	Source Code .....	14
I.8.B.	Hasil .....	15
I.8.C.	Analisa .....	15
I.9	Program 2-9 FundTransfer .....	15
I.9.A.	Source Code .....	15
I.9.B.	Hasil .....	16
I.9.C.	Analisa .....	16
I.10	Program 2-10 InternetBanking.java .....	16
I.10.A.	Source Code .....	16
I.10.B.	Hasil .....	17
I.10.C.	Analisa.....	17
BAB II.	TUGAS PRAKTIKUM .....	18
II.1	Tugas 2-1 .....	18
II.2	Tugas 2-2.....	18
II.3	Tugas 2-3.....	18
II.4	Tugas 2-4.....	18
II.5	Tugas 2-5.....	19
II.6	Tugas 2-6.....	19
II.7	Tugas 2-7.....	20
II.8	Tugas 2-8.....	20

II.9	Tugas 2-9.....	20
BAB III.	KESIMPULAN .....	21

## DAFTAR GAMBAR

Gambar 1 Hasil Output dari Program MemoryException .....	1
Gambar 2 Hasil Output dari Program MemoryException1 .....	2
Gambar 3 Hasil Output dari Program DivideByZeroNoExceptionHandling .....	4
Gambar 4 Hasil Output dari Program DivideByZeroWithExceptionHandling .....	6
Gambar 5 Hasil Output dari program NullReference .....	8
Gambar 6 Hasil Output dari Program Try-catch-finally .....	12
Gambar 7 Hasil Output sebelum di perbaiki pada Program WrongCasting .....	13
Gambar 8 Hasil Output sesudah di perbaiki pada Program WrongCasting .....	13
Gambar 9 Hasil Output dari Program Account.java .....	15
Gambar 10 Hasil Output dari Program FundTransfer .....	16
Gambar 11 Hasil Output dari Program InternetBanking .....	17

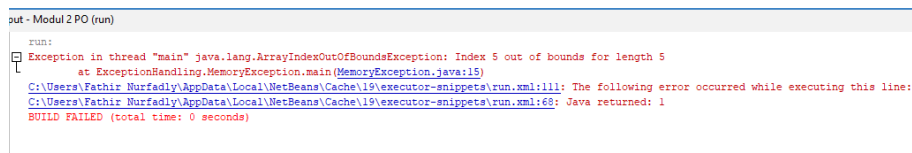
# BAB I. HASIL PRAKTIKUM

## I.1 Program 2-1 MemoryException.java

### I.1.A. Source Code

```
public class MemoryException {  
    public static void main(String []args) {  
        int[] myArray = new int[5];  
        for(int i = 0; i <= 5; i++) {  
            myArray[i] = i;  
        }  
        System.out.println("Hello World");  
    }  
}
```

### I.1.B. Hasil



*Gambar 1 Hasil Output dari Program MemoryException*

### I.1.C. Analisa

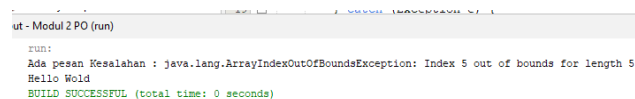
Program tersebut mencoba untuk membuat array `myArray` dengan panjang 5 dan kemudian menginisialisasi setiap elemen array tersebut dengan nilai indeks. Namun, terdapat kesalahan pada loop `for`, dimana iterasi dilakukan dari 0 hingga 5 (inklusif), padahal indeks array dimulai dari 0 hingga 4. Akibatnya, pada iterasi terakhir ( $i = 5$ ), akan terjadi pembacaan atau penulisan di luar batas array yang menyebabkan `ArrayIndexOutOfBoundsException`. Selain itu, program tidak memiliki output yang berkaitan dengan array tersebut, sehingga cetakan "Hello World" yang ditampilkan tidak ada hubungannya dengan operasi yang dilakukan dalam loop.

## I.2 Program 2-2 MemoryException1.java

### I.2.A. Source Code

```
public class MemoryException1 {  
    public static void main(String []args){  
        int[] myArray = new int[5];  
  
        try {  
            for(int i = 0; i <=5; i++) {  
                myArray[i] = i;  
            }  
        } catch (Exception e) {  
            System.out.println("Ada pesan Kesalahan : " + e);  
        }  
  
        System.out.println("Hello Wold");  
    }  
}
```

### I.2.B. Hasil



```
run:
Ada pesan Kesalahan : java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5
Hello Wold
BUILD SUCCESSFUL (total time: 0 seconds)
```

*Gambar 2 Hasil Output dari Program MemoryException1*

### I.2.C. Analisa

Program tersebut hampir sama dengan program sebelumnya, dengan penambahan blok `try-catch` yang mencoba menangkap exception yang mungkin terjadi saat mengakses array di luar batas indeksny. Meskipun demikian, ada beberapa kesalahan dalam program ini. Loop



`for` masih melakukan iterasi dari 0 hingga 5 (inklusif), sehingga tetap terjadi `ArrayIndexOutOfBoundsException`. Selain itu, blok `catch` menangkap semua jenis `Exception` tanpa spesifikasi, yang sebaiknya dihindari karena dapat menyembunyikan masalah yang sebenarnya. Juga, pesan kesalahan yang ditampilkan hanya menampilkan pesan umum `e`, yang kurang memberikan informasi yang berguna dalam men-debug program.

### I.3 Program 2-3 DivideByZeroNoExceptionHandling.java

#### I.3.A. Source Code

```
public class DivideByZeroNoExceptionHandling {  
    public static int pembagian(int bil, int pbg) {  
        return bil / pbg;  
    }  
    public static void main(String []args){  
        Scanner scanner = new Scanner(System.in);  
  
        System.out.println("Masukkan Nilai Pembilang: ");  
        int numerator = scanner.nextInt();  
        System.out.println("Masukkan Nilai Penyebut(Pembagi):  
");  
        int denominator = scanner.nextInt();  
  
        int result = pembagian(numerator, denominator);  
        System.out.printf("%nResult %d / %d = %d%n", numerator,  
denominator, result);  
    }  
}
```

}

### I.3.B. Hasil

```
at - Modul 2 PO (run)
run:
Masukkan Nilai Pembilang:
5
Masukkan Nilai Penyebut (Pembagi):
0
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at ExceptionHandling.DivideByZeroNoExceptionHandling.pembagian(DivideByZeroNoExceptionHandling.java:15)
    at ExceptionHandling.DivideByZeroNoExceptionHandling.main(DivideByZeroNoExceptionHandling.java:25)
C:\Users\Fathir Murtadly\AppData\Local\JetBrains\Cache\19\executor-snippets\run.xml:111: The following error occurred while executing this line:
C:\Users\Fathir Murtadly\AppData\Local\JetBrains\Cache\19\executor-snippets\run.xml:68: Java returned: 1
BUILD FAILED (total time: 25 seconds)
```

Gambar 3 Hasil Output dari Program DivideByZeroNoExceptionHandling

### I.3.C. Analisa

Program tersebut merupakan sebuah program sederhana yang meminta pengguna untuk memasukkan dua bilangan bulat (pembilang dan penyebut) melalui input dari keyboard menggunakan objek `Scanner`. Kemudian, program akan memanggil fungsi `pembagian` untuk melakukan operasi pembagian antara pembilang dan penyebut yang telah dimasukkan oleh pengguna. Hasil pembagian tersebut kemudian dicetak ke layar dengan format yang sesuai menggunakan `printf`. Program ini tidak memperhitungkan penanganan exception jika penyebut (pembagi) bernilai 0, yang dapat menyebabkan terjadinya `ArithmeticException` saat pembagian dengan 0.

## I.4 Program 2-4 DivideByZeroWithExceptionHandling.java

### I.4.A. Source Code

```
public class DivideByZeroWithExceptionHandling {

    public static int pembagian(int bil, int pbgi) throws
ArithmeticException {

        return bil / pbgi;

    }

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

```

Scanner scanner = new Scanner(System.in);

boolean continueLoop = true;

do {
    try {
        System.out.print("Please enter an integer
numerator: ");

        int numerator = scanner.nextInt();

        System.out.print("Please enter an integer
denominator: ");

        int denominator = scanner.nextInt();

        int result = pembagian(numerator, denominator);
        System.out.printf("\nResult: %d / %d = %d\n",
numerator, denominator, result);

        continueLoop = false;
    } catch (InputMismatchException e){
        System.err.printf("\nException: %s\n", e);
        scanner.nextLine();

        System.out.printf("You Must enter Integers.
Please try Again.\n\n");
    } catch (ArithmeticException arithmeticException) {
        System.err.printf("\nException: %s\n",
arithmeticException);

        System.out.printf("Zero is an nvalid
denominator. Please try again.\n\n");
    }
}

```

```

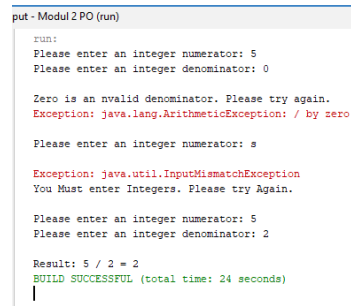
        } while(continueLoop);

    }

}

```

#### I.4.B. Hasil



```

put - Modul 2 PO (run)
run:
Please enter an integer numerator: 5
Please enter an integer denominator: 0

Zero is an nvalid denominator. Please try again.
Exception: java.lang.ArithmeticException: / by zero

Please enter an integer numerator: s
Exception: java.util.InputMismatchException
You Must enter Integers. Please try Again.

Please enter an integer numerator: 5
Please enter an integer denominator: 2

Result: 5 / 2 = 2
BUILD SUCCESSFUL (total time: 24 seconds)

```

*Gambar 4 Hasil Output dari Program DivideByZeroWithExceptionHandling*

#### I.4.C. Analisa

Program ini merupakan sebuah program yang meminta pengguna untuk memasukkan dua bilangan bulat (pembilang dan penyebut) melalui input dari keyboard menggunakan objek `Scanner`. Program kemudian mencoba melakukan operasi pembagian dengan memanggil fungsi `pembagian`. Jika pengguna memasukkan nilai yang tidak valid, seperti bukan bilangan bulat atau mencoba membagi dengan 0, program akan menangani exception yang mungkin terjadi dengan menggunakan blok `try-catch`. Jika terjadi `InputMismatchException`, program akan mencetak pesan kesalahan dan meminta pengguna untuk memasukkan bilangan bulat. Jika terjadi `ArithmeticException` karena pembagian dengan 0, program akan memberikan pesan yang sesuai. Pengguna diberikan kesempatan untuk memasukkan kembali nilai hingga nilai yang valid dimasukkan. Hal ini dilakukan dengan menggunakan loop `do-while` dan variabel boolean `continueLoop`.

### I.5 Program 2-5 NullReference.java

#### I.5.A. Source Code

```

public class NullReference {

```

```

private static Rectangle rectangle; // Tanpa inisialisasi

public static void main(String []args) {
    rectangle = new Rectangle(); // Inisialisasi objek
Rectangle di sini

    if(rectangle == null) {
        System.out.println("Rectangle variable doesn't
refer to a Rectangle object");
    } else {
        int area = rectangle.area();
        System.out.println("Area: " + area);
    }
}

class Rectangle {
    private int length;
    private int width;

    public Rectangle() {
        this.length = 5; // Contoh nilai panjang
        this.width = 3;  // Contoh nilai lebar
    }

    public int area() {

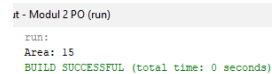
```

```

        return length * width;
    }
}

```

### I.5.B. Hasil



```

rt - Modul 2 PO (run)
run:
Area: 15
BUILD SUCCESSFUL (total time: 0 seconds)

```

*Gambar 5 Hasil Output dari program NullReference*

### I.5.C. Analisa

Program ini adalah contoh penerapan inisialisasi objek dan penanganan referensi null dalam bahasa pemrograman Java. Kelas `NullReference` memiliki variabel statis `rectangle` yang dideklarasikan tanpa inisialisasi, kemudian diinisialisasi dengan objek dari kelas `Rectangle` dalam metode `main`. Program memeriksa apakah objek `rectangle` telah diinisialisasi; jika belum (nilainya null), akan dicetak pesan bahwa variabel `rectangle` tidak merujuk pada objek `Rectangle`. Jika sudah diinisialisasi, program memanggil metode `area()` pada objek `rectangle` untuk menghitung dan mencetak luasnya. Kelas `Rectangle` memiliki atribut panjang dan lebar, serta metode konstruktor untuk menginisialisasi nilai panjang dan lebar dengan nilai contoh 5 dan 3, serta metode `area()` untuk menghitung luasnya.

## I.6 Program 2-6 Try-catch-finally

### I.6.A. Source Code

Sebelum

```

public class UsingExceptions {
    public static void main(String []args) {
        try {
            throwException;

```

```

        } catch (Exception exception) {
            System.err.println("Exception handled in main");
        } doesNotThrowException();
    }

    public static void throwException() throws Exception {
        try {
            System.out.println("Method throwException");
            throws new Exception();
        } catch (Exception exception) {
            System.err.println("Exception handled in method
throwException");
            throw exception;
        } finally {
            System.err.println("Finaaly Executed in
throwException");
        }
    }

    public static void doesNotThrowException() {
        try {
            System.out.println("Method doesNotThrowException");
        } catch (Exception exception) {
            System.err.println(exception);
        } finally {

```

```

        System.err.println("Finally executed in
doesNotThrowException");

    }

    System.out.println("End of method
doesNotThrowException");

}

private static void doesnotThrowException() {

    throw new UnsupportedOperationException("Not supported
yet."); // Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMetho
dBody

}

}

```

Sesudah

```

public class UsingExceptions {

    public static void main(String []args) {

        try {

            throwException();

        } catch (Exception exception) {

            System.err.println("Exception handled in main");

        }

        doesNotThrowException();

    }

    public static void throwException() throws Exception {

        try {

```



```

        System.out.println("Method throwException");
        throw new Exception(); // Perbaiki kesalahan sintaks
    } catch (Exception exception) {
        System.err.println("Exception handled in method
throwException");
        throw exception;
    } finally {
        System.err.println("Finally Executed in
throwException");
    }
}

public static void doesNotThrowException() {
    try {
        System.out.println("Method doesNotThrowException");
    } catch (Exception exception) {
        System.err.println(exception);
    } finally {
        System.err.println("Finally executed in
doesNotThrowException");
    }
    System.out.println("End of method
doesNotThrowException");
}
}

```

```

at - Modul 2 PO (run)
run:
Method throwException
Exception handled in method throwException
Method doesNotThrowException
Finally Executed in throwException
Exception handled in main
Finally executed in doesNotThrowException
End of method doesNotThrowException
BUILD SUCCESSFUL (total time: 0 seconds)

```

*Gambar 6 Hasil Output dari Program Try-catch-finally*

### **I.6.C. Analisa**

Program ini adalah contoh penggunaan blok `try-catch-finally` dalam bahasa pemrograman Java untuk menangani pengecualian. Metode `main` mencoba memanggil metode `throwException`, yang didalamnya terdapat blok `try-catch-finally` untuk menangani pengecualian yang dilempar. Jika terjadi pengecualian di dalam `throwException`, pesan kesalahan akan dicetak dan pengecualian akan dilempar kembali ke metode pemanggil. Metode `doesNotThrowException` digunakan untuk menunjukkan sebuah metode yang tidak melempar pengecualian, sehingga tidak perlu menggunakan blok `try-catch-finally`.

## **I.7 Program 2-7 WrongCasting.java**

### **I.7.A. Source Code**

Sebelum

```

public class WrongCasting {
    public static void main(String []args) {
        Object rectangle = new Rectangle(10, 10);
        String str = (String) rectangle;
        System.out.println("String str : " + str);
    }
}

```

Sesudah

```

public class WrongCasting {

```

```

        public static void main(String []args) {

            Rectangle rectangle = new Rectangle(10, 10);

            String str = rectangle.toString(); // Menggunakan metode
toString() dari kelas Rectangle

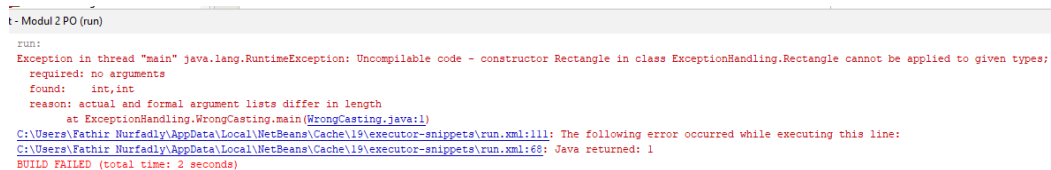
            System.out.println("String str : " + str);

        }

    }

```

### I.7.B. Hasil



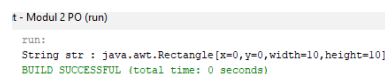
```

t - Modul 2 PO (run)

run:
Exception in thread "main" java.lang.RuntimeException: Uncompilable code - constructor Rectangle in class ExceptionHandling.Rectangle cannot be applied to given types;
required: no arguments
found:    int,int
reason: actual and formal argument lists differ in length
    at ExceptionHandling.WrongCasting.main(WrongCasting.java:1)
C:\Users\Fathir Nurfadly\AppData\Local\NetBeans\Cache\19\executor-snippets\run.xml:111: The following error occurred while executing this line:
C:\Users\Fathir Nurfadly\AppData\Local\NetBeans\Cache\19\executor-snippets\run.xml:68: Java returned: 1
BUILD FAILED (total time: 2 seconds)

```

*Gambar 7 Hasil Output sebelum di perbaiki pada Program WrongCasting*



```

t - Modul 2 PO (run)

run:
String str : java.awt.Rectangle[x=0,y=0,width=10,height=10]
BUILD SUCCESSFUL (total time: 0 seconds)

```

*Gambar 8 Hasil Output sesudah di perbaiki pada Program WrongCasting*

### I.7.C. Analisa

Program tersebut mencoba melakukan penugasan objek dari kelas `Rectangle` ke variabel `rectangle` dengan tipe `Object`, yang memungkinkan objek dari kelas apa pun untuk disimpan di dalamnya. Namun, kemudian program mencoba melakukan operasi casting yang tidak valid dari objek `Rectangle` ke tipe `String`. Ini merupakan kesalahan, karena tidak mungkin mengonversi objek `Rectangle` secara langsung menjadi objek `String`. Kesalahan ini dapat menyebabkan `ClassCastException` saat program dijalankan. Untuk memperbaiki program, kita bisa menggunakan metode `toString()` yang sudah ada pada objek `Rectangle` untuk menghasilkan representasi string dari objek tersebut.

## I.8 Program 2-8 Account.java

### I.8.A. Source Code

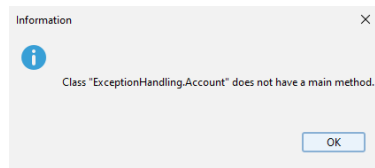
```
public class Account {  
    private int accountNumber;  
    private int currentBalance;  
  
    // Konstruktor untuk inisialisasi nomor akun dan saldo awal  
    public Account(int accountNumber, int initialBalance) {  
        this.accountNumber = accountNumber;  
        this.currentBalance = initialBalance;  
    }  
  
    public void credit(int amt) {  
        currentBalance = currentBalance + amt;  
    }  
  
    public void debit(int amt) {  
        int tempBalance = currentBalance - amt;  
        if(tempBalance < 0) {  
            int i = 1 / 0;  
            System.out.println(i);  
        }  
        currentBalance = tempBalance;  
    }  
  
    // Metode untuk mengembalikan saldo saat ini
```

```

        public int getCurrentBalance() {
            return currentBalance;
        }
    }
}

```

### I.8.B. Hasil



*Gambar 9 Hasil Output dari Program Account.java*

### I.8.C. Analisa

Kelas `Account` merupakan representasi dari sebuah akun dalam sistem perbankan, yang memiliki atribut nomor akun (`accountNumber`) dan saldo saat ini (`currentBalance`). Konstruktor kelas ini digunakan untuk menginisialisasi nomor akun dan saldo awal. Metode `credit` digunakan untuk menambahkan sejumlah uang ke saldo akun, sedangkan metode `debit` digunakan untuk mengurangi sejumlah uang dari saldo akun. Metode `debit` memeriksa apakah saldo yang tersisa setelah pengurangan akan menjadi negatif; jika ya, maka akan dilemparkan `ArithmeticException`. Metode `getCurrentBalance` digunakan untuk mengembalikan nilai saldo saat ini dari akun.

## I.9 Program 2-9 FundTransfer

### I.9.A. Source Code

```

public class FundTransfer {
    public static boolean transferFunds(Account xAcc, Account
yAcc, int amt) {
        xAcc.debit(amt);
    }
}

```

```

        yAcc.credit(amt);

        System.out.println("Completed fund transfer");

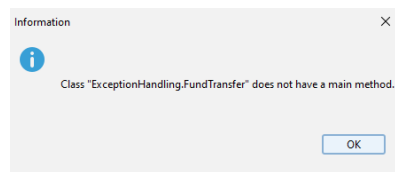
        return true;

    }

}

```

### I.9.B. Hasil



*Gambar 10 Hasil Output dari Program FundTransfer*

### I.9.C. Analisa

Program `FundTransfer` adalah sebuah kelas yang menyediakan sebuah metode statis bernama `transferFunds`. Metode ini menerima dua objek akun (`xAcc` dan `yAcc`) dan jumlah uang yang akan ditransfer (`amt`) sebagai argumen. Dalam metode ini, saldo dari `xAcc` dikurangi sejumlah `amt` menggunakan metode `debit`, dan saldo dari `yAcc` ditambahkan sejumlah `amt` menggunakan metode `credit`. Setelah proses transfer selesai, pesan "Completed fund transfer" dicetak. Metode ini mengembalikan nilai boolean `true` yang menunjukkan bahwa transfer berhasil dilakukan.

## I.10 Program 2-10 InternetBanking.java

### I.10.A. Source Code

```

public class InternetBanking {

    public static void main(String []args) {

        Account xAcc = new Account(1, 1000);

        Account yAcc = new Account(2, 1200);
    }
}

```

```

        FundTransfer.transferFunds(xAcc, yAcc, 1400);

        System.out.println("xAcc's    current    balance    "    +
xAcc.getCurrentBalance());

        System.out.println("yAcc's    current    balance    "    +
yAcc.getCurrentBalance());

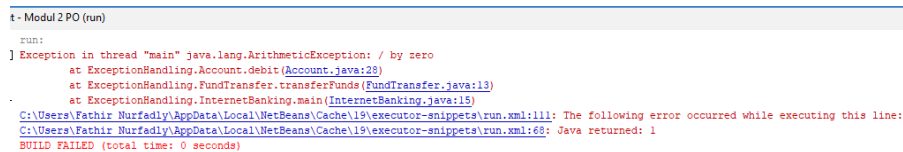
        System.out.println("Completed    execution    of    main
method");

    }

}

```

### I.10.B. Hasil



```

t- Modul 2 PO (run)
run:
] Exception in thread "main" java.lang.ArithmeticException: / by zero
  at ExceptionHandling.Account.debit (Account.java:28)
  at ExceptionHandling.FundTransfer.transferFunds (FundTransfer.java:13)
  at ExceptionHandling.InternetBanking.main (InternetBanking.java:15)
C:\Users\Fathir Nurfadly\AppData\Local\NetBeans\Cache\19\executor-snippets\run.xml:111: The following error occurred while executing this line:
C:\Users\Fathir Nurfadly\AppData\Local\NetBeans\Cache\19\executor-snippets\run.xml:68: Java returned: 1
BUILD FAILED (total time: 0 seconds)

```

*Gambar 11 Hasil Output dari Program InternetBanking*

### I.10.C. Analisa

Program `InternetBanking` merupakan program utama yang melakukan simulasi penggunaan sistem perbankan melalui internet. Dalam metode `main`, dua objek akun (`xAcc` dan `yAcc`) dibuat dengan saldo awal masing-masing sebesar 1000 dan 1200. Selanjutnya, metode `transferFunds` dari kelas `FundTransfer` dipanggil untuk mentransfer dana sebesar 1400 dari `xAcc` ke `yAcc`. Setelah transfer selesai, saldo terbaru dari kedua akun dicetak ke layar bersama dengan pesan yang menunjukkan bahwa eksekusi metode `main` telah selesai.

## **BAB II. TUGAS PRAKTIKUM**

### **II.1 Tugas 2-1**

1. Program sudah di Compile pada BAB1. Hasil Praktikum Program 2-1
2. Program sudah di Analisa pada BAB1. Hasil Praktikum Program 2-1

### **II.2 Tugas 2-2**

1. Program sudah di Compile pada BAB1. Hasil Praktikum Program 2-2
2. Program sudah di Analisa pada BAB1. Hasil Praktikum Program 2-2
3. Perbedaan penggunaan saat menggunakan try catch dan tidak. Jika tidak menggunakan try catch maka program akan error, sementara apabila menggunakan try catch program akan berjalan tanpa error walau ada error dikarenakan ada nya pengecualian untuk error.

### **II.3 Tugas 2-3**

1. Program sudah di perbaiki pada BAB1. Hasil Praktikum Program 2-3
2. Program Menampilkan error jika diberikan nilai pembagi 0
3. Program tidak memiliki exception handling didalam nya, karena itu akan menampilkan error. Jika program menggunakan try catch maka program akan berjalan dan menampilkan pesan jika apapun yang dibagi dengan 0 maka akan error.

### **II.4 Tugas 2-4**

1. Program sudah di perbaiki pada BAB1. Hasil Praktikum Program 2-4
2. Program menampilkan pesan error jika di bagi dengan 0, dan menampilkan pesan bahwa huruf bukanlah angka.
3. Program menampilkan pesan error tetapi program masih dapat berjalan sebagaimana mestinya.
4. Walaupun program diberikan nilai 0, program akan terus berjalan karena ada nya exception handling. Exception handling ini mengurus pengecualian pada program, misalnya jika ada angka yang ingin di bagi dengan 0, atau angka yang akan di bagi dengan huruf, maka exception handling akan menanganinya, dan program akan terus berjalan bila program belum memenuhi tugasnya.



## II.5 Tugas 2-5

Program sudah di perbaiki pada BAB1. Hasil Praktikum Program 2-5. Pada program ini akan terus error karena ada variabel area yang belum terdefinisi. Maka di buatlah kelas untuk mendefinisikan area tersebut.

## II.6 Tugas 2-6

1. Program sudah di perbaiki pada BAB1. Hasil Praktikum Program 206
2. Semua ditampilkan dalam console
3. `throw` dan `throws` adalah dua konsep penting dalam penanganan pengecualian (exception handling) dalam bahasa pemrograman Java. Kata kunci `throw` digunakan untuk melempar atau menghasilkan pengecualian secara eksplisit di dalam blok kode. Ketika kondisi tertentu terpenuhi, misalnya kesalahan atau kondisi yang tidak diinginkan terjadi, program dapat menggunakan `throw` untuk menghentikan alur normal eksekusi dan melemparkan pengecualian yang sesuai. Pengecualian tersebut kemudian akan ditangani di tempat lain dalam kode program, biasanya dengan menggunakan blok `try-catch`. Di sisi lain, kata kunci `throws` digunakan dalam deklarasi metode untuk menunjukkan bahwa metode tersebut mungkin melempar pengecualian tertentu. Dengan menggunakan `throws`, kita memberi tahu pemanggil metode bahwa mereka perlu menangani pengecualian yang mungkin dilemparkan oleh metode tersebut. Ketika metode yang dideklarasikan dengan `throws` dipanggil, pemanggil metode harus menangani pengecualian tersebut dengan menggunakan blok `try-catch` atau meneruskan kembali (`throws`) pengecualian ke metode yang memanggilmnya.
4. Blok `finally` adalah bagian dari konstruksi penanganan pengecualian (exception handling) dalam bahasa pemrograman Java yang digunakan untuk menentukan bagian dari kode yang akan dieksekusi terlepas dari apakah sebuah pengecualian terjadi atau tidak. Blok `finally` akan selalu dieksekusi, bahkan jika terjadi pengecualian di dalam blok `try` yang terkait atau jika kode di dalam blok `try` selesai dijalankan tanpa pengecualian. Ini membuat blok `finally` berguna untuk menempatkan kode yang harus dijalankan untuk membersihkan sumber daya yang mungkin dibuka di dalam blok `try`, seperti file atau koneksi database, sehingga sumber daya tersebut akan dilepaskan atau ditutup dengan benar tanpa tergantung pada kejadian pengecualian.

5. Program sudah di Analisa pada BAB1. Hasil Praktikum Program 2-6

## **II.7 Tugas 2-7**

1. Program sudah di Analisa pada BAB1. Hasil Praktikum Program 2-7
2. Program sudah di perbaiki pada BAB1. Hasil Praktikum Program 2-7

## **II.8 Tugas 2-8**

Program sudah di lengkapi pada BAB1. Hasil Praktikum Program 2-8

## **II.9 Tugas 2-9**

Program sudah di perbaiki dan di lengkapi pada BAB1. Hasil Praktikum Program 2-8 hingga Program 2-10. Program juga sudah diberikan Analisa

### **BAB III. KESIMPULAN**

Dalam penanganan pengecualian (exception handling) di Java, terdapat beberapa konsep penting yang digunakan untuk mengelola dan menangani pengecualian dengan baik. Pertama, kata kunci `throw` digunakan untuk melemparkan pengecualian secara eksplisit di dalam kode, sedangkan kata kunci `throws` digunakan dalam deklarasi metode untuk menunjukkan bahwa metode tersebut mungkin melemparkan pengecualian tertentu. Selain itu, blok `finally` adalah bagian dari konstruksi penanganan pengecualian yang selalu dieksekusi terlepas dari terjadinya pengecualian atau tidak, dan sering digunakan untuk membersihkan sumber daya yang dibuka di dalam blok `try`.