Modul 9
Exception

A) Pokok Bahasan

1. Exception
2. Exception Handling
3. Throws dan Throw

B) Tujuan

1. Mahasiswa Mampu Memahami Apa itu Exception

C) Dasar Teori

1. Exception

   Exception adalah Masalah-masalah yang terjadi pada saat program berjalan. Saat Exception terjadi, program secara otomatis akan menampilkan detail Exception yang terjadi dan menghentikan alur program seketika

   Ada 3 Jenis Exception :

   a) Checked Exception

   Exception yang terjadi saat source code dicompile

   ```
   Contoh :
   public static void main(String []args) {
      System.out.println( this );
      /* error: non-static variable this cannot be referenced from a
      static context */
   }
   ```

   b) Unchecked Exception

   Exception yang terjadi saat program sedang berjalan

   ```
   Contoh :
   public static void main(String []args) {
      System.out.println( args[9999] );
      /* Exception in thread "main"
      java.lang.ArrayIndexOutOfBoundsException: 9999 */
   }
   ```

   c) Error

Masalah yang terjadi dan tidak akan dapat ditangani oleh programmer

```
Contoh :
public static void re() {
   re();
}
public static void main(String []args) {
   re();
   /* Exception in thread "main" java.lang.StackOverflowError */
}
```

2. Exception Handling

Java memberikan cara untuk menangani Exception apabila Exception itu terjadi

a) try-catch

Kode yang kemungkinan akan menimbulkan Exception ditaruh di dalam block `try`. Apabila terjadi Exception, program akan menjalankan statement yang terdapat pada block `catch`

```
Syntax :
try {
   [statement]
} catch ([exception_class] [object_name]) {
   [handling]
}

Contoh :
try {
   int x = 2 / 0;
} catch (Exception e) {
   System.out.println("Exception ditemukan");
}
System.out.println("Program Berlanjut");
```

b) multiple catch

Kita juga bisa melakukan handling yang berbeda untuk setiap Exception

```
Contoh :
try {
   int []a = {10, 20, 30};
   System.out.println(a[3]);
} catch (ArithmeticException e) {
   System.out.println("Masalah Aritmatika");
} catch (ArrayIndexOutOfBoundsException e) {
   System.out.println("Masalah Index Kelewat Batas");
}
System.out.println("Program Berlanjut");
```

c) finally

Block `finally` berada setelah semua block `catch`. Statement pada block `finally` akan selalu dijalankan tidak peduli Exception itu terjadi atau tidak

```
Contoh :
try {
   System.out.println(2/0);
} catch(ArrayIndexOutOfBoundsException e) {
   System.out.println("Index Array Kelewat Batas");
} finally {
   System.out.println("Akan Selalu Dijalankan");
}
```

3. Throws dan Throw

Throw menandakan bahwa telah terjadi Exception pada saat itu. Untuk melakukan throw, method yang melakukannya harus ditambahkan keyword `throws` yang kemudian diikuti dengan jenis Exception apa saja yang akan di throw

```
Syntax :
[access_modifier] [return_type] [nama_method]([parameter]) throws
[exceptions] {
   throw [throwable_object];
}

Contoh :
public static void main(String []args) throws Exception, Error {
   String papa = "Baik";
   if (papa.equals("Jahat")) {
      throw new Exception("Papa Jahad");
   } else if (papa.equals("Bejat")) {
      throw new Error("Papa bejad");
   }
   System.out.println("Program Berjalan dengan baik");
}
```

D) Praktik

file : PembelianGagalException.java

```
// Custom Exception
public class PembelianGagalException extends Exception {
    private int uang;
    private int harga;
    private String alasan;

    public PembelianGagalException(int uang, int harga) {
        this.uang = uang;
        this.harga = harga;
```

```
        this.periksaAlasan();
    }

    private void periksaAlasan() {
        if (this.uang < 0) {
            this.alasan = "Uang bernilai negatif";
        } else if (this.harga < 0) {
            this.alasan = "Harga bernilai negatif";
        } else if (this.uang < this.harga) {
            this.alasan = "Uang tidak Cukup";
        } else {
            this.alasan = "Alasan tidak diketahui";
        }
    }

    public void tampilkanAlasan() {
        System.out.println("Pembelian Gagal karena "+this.alasan);
    }
}
```

file : Main.java

```
import java.util.Scanner;
import java.util.InputMismatchException;

public class Main {

    public static void beli(String barang, int uang, int harga) throws
PembelianGagalException {
        if (uang < 0 || harga < 0 || uang < harga) {
            throw new PembelianGagalException(uang, harga);
        } else {
            System.out.println("Pembelian Berhasil");
            System.out.println("Kembalian = "+ (harga-uang));
        }
    }

    public static void main(String []args) {
        Scanner input = new Scanner(System.in);
        int uangAnda, hargaBarang;
        String barangYangInginDibeli;

        while (true) {
            try {
                System.out.print("Masukkan uang Anda : ");
                uangAnda = input.nextInt();
                // Mengatasi masalah buffer input
                input.nextLine();

                System.out.print("Masukkan harga Barang : ");
                hargaBarang = input.nextInt();
                // Mengatasi masalah buffer input
```

```
                input.nextLine();

                System.out.print("Masukkan Barang Yang Ingin Dibeli : ");
                barangYangInginDibeli = input.nextLine();
                beli(barangYangInginDibeli, uangAnda, hargaBarang);

                break;
            } catch (InputMismatchException e) {
                System.out.println("Yang Anda Masukkan Tidak Cocok");
                // Mengatasi Masalah Buffer input
                input.nextLine();
                continue;
            } catch (PembelianGagalException e) {
                System.out.println("Pembelian Gagal");
                e.tampilkanAlasan();
                break;
            } finally {
                System.out.println("Transaksi Selesai");
            }
        }
    }
}
```