PERTEMUAN KESEMBILAN

Java Packages dan Enumerasi

Prepared by Adi Wahyu Pribadi

Pendahuluan

Java Package adalah mekanisme untuk mengelompokkan kelas (class), antarmuka (interface), enumerasi (enum), dan anotasi (annotation) yang terkait dalam sebuah unit namespace yang terstruktur.

Package di Java terbagi menjadi dua macam:

- *Built-in Package* (Paket dari Java API)
- *User-defined Package* atau buatan sendiri

Kegunaan Package

- Mengorganisir kode
  Memudahkan pengelolaan dan pencarian kelas yang terkait.
- Menghindari konflik dalam penamaan
  Kelas dengan nama yang sama dapat dibedakan berdasarkan package-nya.
- Kontrol akses
  Menentukan kelas atau anggota kelas mana yang dapat diakses dari package lain.

Built-in Package

Java API adalah library kumpulan Class yang sudah ditulis dan dapat langsung digunakan secara bebas. Library berisi komponen untuk mengelola input, pemrograman basis data, dan banyak lagi lainnya. Dengan built-in packages, pengembang dapat fokus pada logika bisnis aplikasi tanpa harus mengimplementasikan fungsi dasar dari awal.

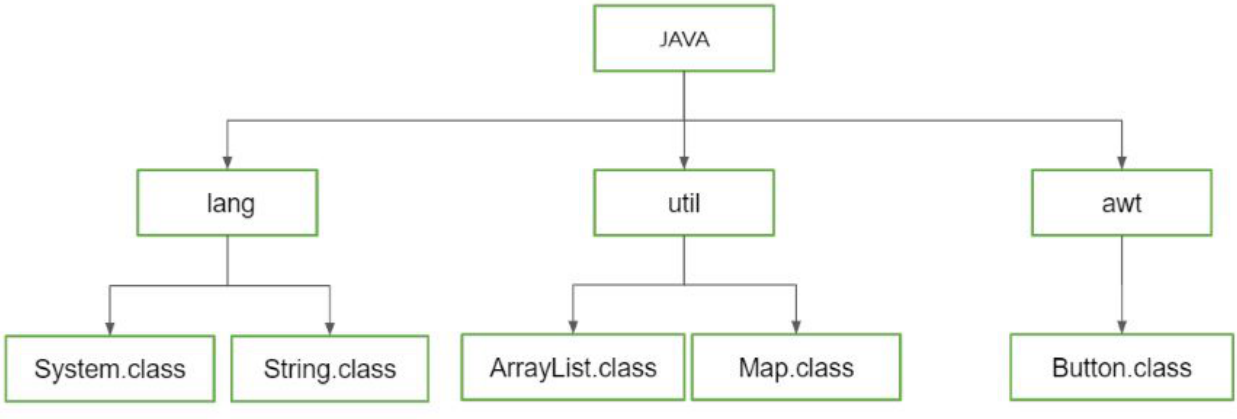
Daftar lengkapnya dapat ditemukan di situs web Oracles:

https://docs.oracle.com/javase/8/docs/api/

https://docs.oracle.com/en/java/javase/11/docs/api/index.html

Built-in Package Java 19 (Oktober 2023)

https://www.geeksforgeeks.org/built-in-packages-in-java/



Beberapa built-in package Oracle Java 19:

java.lang	java.math
java.util	java.io
java.net	java.sql
java.security	java.util.concurrent
java.util.stream	java.util.regex
java.util.zip	java.awt
java.swing	

Built-in package menyediakan fungsi dasar yang dibutuhkan sebagian besar aplikasi Java. Beberapa kegunaan built-in package adalah:

- memanipulasi string dan angka
- mengelola file dan jaringan
- mengakses database
- melakukan enkripsi dan dekripsi
- mengelola multithreading
- mengembangkan aplikasi GUI

Library terbagi atas packages dan class. Berarti kita dapat mengimpor satu class (beserta metode dan atributnya), atau seluruh package yang berisi semua class yang ada di dalam package yang kita pilih.

Untuk menggunakan sebuah class atau package dari library, kita gunakan keyword **import**

Syntax

```
import package.name.Class;   // Import sebuah class
import package.name.*;       // Import seluruh package
```

Import sebuah class

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

Pada baris di atas, nama package-nya adalah java.util dan nama class-nya adalah Scanner. Untuk menggunakan buat objek class dan gunakan salah satu metode yang tersedia yang ditemukan dalam dokumentasi class Scanner. Kita akan menggunakan method nextLine(), yang digunakan untuk membaca baris lengkap:

```
import java.util.Scanner;
class MyClass {
  public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);
    System.out.println("Masukkan nama Anda");

    String userName = myObj.nextLine();
    System.out.println("Nama Anda adalah " + userName);
  }
}
```

Import Sebuah Package

Ada banyak paket untuk dipilih. Pada contoh sebelumnya, kita menggunakan class Scanner dari package java.util. Package ini juga berisi fungsi utilitas seperti tanggal dan waktu, pembuat nomor acak, dan class utilitas lainnya.

Untuk mengimpor seluruh package, akhiri kalimat dengan tanda bintang (*). Contoh berikut akan mengimpor semua class di package java.util:

```
import java.util.*;
```

User-Defined Package atau Paket Buatan Sendiri

Untuk membuat package sendiri, Java menggunakan sistem folder file untuk menyimpan, sama seperti folder di komputer. Misalkan:

```
|___ src
   |___ App
      |___ ClassSaya.java
```

Untuk menggunakan sebuah package gunakan keyword package;

```
package App;
class ClassSaya {
  public static void main(String[] args) {
    System.out.println("Ini sebuah package!");
  }
}
```

Simpan file di atas dengan nama ClassSaya.java di dalam folder src

Compile File ClassSaya.java dengan perintah

```
C:\Users\Adiwa\src> javac ClassSaya.java
```

Menjalankan file ClassSaya.class

```
C:\Users\Adiwa\src>java ClassSaya
Error: Could not find or load main class ClassSaya
Caused by: java.lang.NoClassDefFoundError: App/ClassSaya (wrong name:
ClassSaya)
```

Muncul Error dikarenakan ClassSaya terdapat package App

Maka gunakan perintah javac -d . ClassSaya.java

```
C:\Users\Adiwa\src>javac -d . ClassSaya.java
```

Dan untuk menjalankan ClassSaya adalah dengan perintah `java NamaPackage.NamaClass`

```
C:\Users\Adiwa\src>java App.ClassSaya
Ini sebuah package!
```

Java Enums

Enum adalah Class spesial yang merepresentasikan grup KONSTAN/CONSTANTS (variable yang tidak bisa diubah, seperti variabel final). Untuk membuath sebuah enum, gunakan keyword enum, dan pisahkan constant dengan tanda koma (,). Ingat, constant ditulis dengan HURUF BESAR.

```
enum Level {
   LOW, MEDIUM, HIGH
}
```

Untuk mengakses konstan di enum gunakan titik / dot syntax:

```
Level kemahiran = Level.MEDIUM;
Level rasaPedas = Level.MEDIUM;
Level rasaPedas = Level.HIGH;
```

Enum di dalam sebuah Class

```
Public class ClassKu {
   enum Level {
      LOW, MEDIUM, HIGH
   }
   Public static void main(String[] args) {
      Level kemahiran = Level.HIGH;
      System.out.println(kemahiran);
   }
}
```

Outputnya:

```
HIGH
```

Enum di dalam Statement Switch

```
enum Level {
  LOW, MEDIUM, HIGH
}

public class ClassKu {
  public static void main(String[] args) {
    Level kemahiran = Level.MEDIUM;

    switch(kemahiran) {
      case LOW:
        System.out.println("Anak bawang");
        break;
      case MEDIUM:
         System.out.println("Anggota");
        break;
      case HIGH:
        System.out.println("Suhu");
        break;
    }
  }
}
```

Outputnya:

```
Anggota
```

Looping di dalam Enum

Tipe enum memiliki sebuah method bernama values(), method tersebut akan mengembalikan array dari seluruh constant yang ada di dalam enum tersebut. Method ini bermanfaat ketika ingin menampilkan seluruh constant yang ada di enum.

```
for (Level kemahiran : Level.values()) {
  System.out.println(kemahiran);
}
```

Outputnya:

```
LOW
MEDIUM
HIGH
```

Java Enum juga dapat digunakan untuk membuat method dan properti. Berikut contoh penggunaannya:

```
enum Warna {
  MERAH(255, 0, 0),
  HIJAU(0, 255, 0),
  BIRU(0, 0, 255);

  private int r, g, b;

  Warna(int r, int g, int b) {
    this.r = r;
    this.g = g;
    this.b = b;
  }

  public int getRed() {
    return r;
  }
  public int getGreen() {
    return g;
  }
  public int getBlue() {
    return b;
  }
}
```

```
public class DemoWarna {
    public static void main(String[] args) {
        // Menampilkan semua warna dan komponen RGB mereka
        for (Warna warna : Warna.values()) {
            System.out.println("Warna: " + warna.name());
            System.out.println("  Merah  : " + warna.getRed());
            System.out.println("  Hijau  : " + warna.getGreen());
            System.out.println("  Biru   : " + warna.getBlue());
            System.out.println();
        }

        // Demonstrasi penggunaan enum dalam kondisi
        Warna baju = Warna.HIJAU;

        if (baju == Warna.HIJAU) {
            System.out.println("Baju berwarna hijau cocok untuk pergi ke
alam!");
        } else if (baju == Warna.MERAH) {
            System.out.println("Baju berwarna merah cocok untuk pesta!");
        } else if (baju == Warna.BIRU) {
            System.out.println("Baju berwarna biru cocok untuk ke kantor!");
        }
    }
}
```

Output:

```
Warna: MERAH
  Merah  : 255
  Hijau  : 0
  Biru   : 0

Warna: HIJAU
  Merah  : 0
  Hijau  : 255
  Biru   : 0

Warna: BIRU
  Merah  : 0
  Hijau  : 0
  Biru   : 255

Baju berwarna hijau cocok untuk pergi ke alam!
```

Fitur	Enum	Class
Nilai	Terbatas	Tidak terbatas
Metode	Tidak ada	Ada
Properti	Tidak ada	Ada
Kegunaan	Representasi nilai-nilai yang terbatas	Representasi kumpulan data dan perilaku