ArrayList

ArrayList

- **Java ArrayList** adalah bagian dari **kerangka koleksi** Java dan merupakan kelas paket java.util. Ini memberi kita array dinamis di Java.
- Meskipun demikian, ini mungkin lebih lambat dari array standar tetapi dapat membantu dalam program yang memerlukan banyak manipulasi dalam array.
- Kelebihan utama **ArrayList di Java** adalah jika kita mendeklarasikan sebuah array maka kita perlu menyebutkan ukurannya, namun pada ArrayList tidak perlu menyebutkan ukuran ArrayList. Jika ingin menyebutkan ukurannya bisa juga.

Generic Vs Non-Generic ArrayList

- Non-generic ArrayList diperkenalkan pertama kali sebelum Java 5 dan tidak mendukung generic
- Dapat menyimpan objek dari berbagai tipe karena secara default menerima tipe data Object. Ini berarti ArrayList tanpa generic dapat menyimpan elemen dengan tipe data apa pun.
- Type-Safety: Tidak ada. Karena dapat menyimpan objek dari berbagai tipe, tidak ada pengecekan tipe saat waktu kompilasi. Hal ini dapat menyebabkan ClassCastException jika objek diambil dari ArrayList dan dikonversi ke tipe yang salah pada waktu runtime.

Generic Vs Non-Generic ArrayList

- Generic diperkenalkan dalam Java 5 untuk memberikan type-safety.
- ArrayList generic memungkinkan penentuan tipe data spesifik yang dapat disimpan. Ini dilakukan melalui generic types. Saat sebuah ArrayList dideklarasikan dengan tipe tertentu, ia hanya dapat menyimpan elemen dari tipe tersebut.
- Menyediakan type-safety pada waktu kompilasi. Ini berarti jika Anda mencoba menambahkan objek dari tipe yang berbeda dari yang telah ditentukan, akan terjadi error pada saat kompilasi, bukan pada saat runtime.

Contoh Non-Generic ArrayList

```
ArrayList list = new ArrayList();
list.add(1); // Menambahkan integer
list.add("Java"); // Menambahkan String
// Tidak aman, karena membutuhkan casting saat pengambilan data
String data = (String) list.get(1); // Potensial ClassCastException
```

Contoh Generic ArrayList

```
ArrayList<String> list = new ArrayList<String>();
list.add("Java"); // Valid
list.add(1); // Error kompilasi
// Tidak membutuhkan casting karena tipe sudah diketahui
String data = list.get(0);
```

Contoh Generic ArrayList

```
// Membuat ArrayList untuk menyimpan String dengan kapasitas awal 10
ArrayList<String> names = new ArrayList<>(10);

// Menambahkan elemen ke dalam ArrayList
names.add("Alice");
names.add("Bob");
names.add("Charlie");
names.add("Diana");
```

Operasi dalam ArrayList (Class Hero)

```java
package com.tutorial;

public class Hero {
    private String name;
    private double health;

    Hero(String name, double health){
        this.name = name;
        this.health = health;
    }

    void setName(String name){
        this.name = name;
    }

    void display(){
        System.out.println("This is " + this.name + " with health " + this.health);
    }

    public String toString(){
        return "Hero: " + this.name;
    }
}
```

Operasi menambah (Class Main (1))

```java
package com.tutorial;

import java.util.*;

public class Main {
    Run | Debug
    public static void main(String[] args) {

        Hero hero1 = new Hero(name:"Ucup",health:100);
        Hero hero2 = new Hero(name:"Otong",health:100);
        Hero heroAgility = new Hero(name:"Dudung", health:50);
        Hero heroIntel = new Hero(name:"Johny", health:20);

        ArrayList<Hero> listHero = new ArrayList<>();

        // operasi satu: menambah member dengan add
        System.out.println(x:"Operasi 1: add");
        listHero.add(hero1);
        listHero.add(hero2);
        listHero.add(heroAgility);

        System.out.println(listHero);
        System.out.println();
```

Operasi merubah (Class Main (2))

```
// operasi dua: merubah member dengan set
System.out.println(x:"Operasi 2: set");
listHero.set(index:1,heroIntel);
System.out.println(listHero);
System.out.println();
```

Operasi menghapus (Class Main (3))

```
// operasi ketiga: menghapus member dengan remove
System.out.println(x:"Operasi 3: remove");
listHero.remove(index:1);
System.out.println(listHero);
System.out.println();
```

Operasi mengakses (Class Main (4))

```
// operasi keempat: mengakses member dengan get
System.out.println(x:"Operasi 4: get");
System.out.println(listHero);
Hero heroAmbil = listHero.get(index:0);
heroAmbil.setName(name:"Farhan");
System.out.println(listHero);
heroAmbil.display();
System.out.println();
```

Foreach pada ArrayList (Class Main (5))

```
for(Hero list: listHero){
    System.out.println(list);
}
```

Operasi lain-lain (Class Main (6))

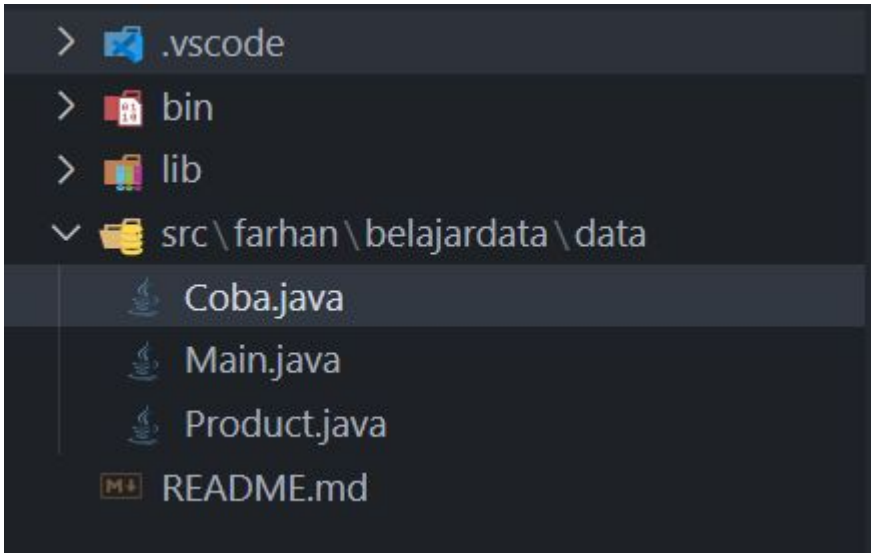
```
// method pada arraylist
System.out.println(x:"method2");
System.out.println("1. size()\t: " + listHero.size());
System.out.println("2. isEmpty()\t: " + listHero.isEmpty());
System.out.println("3. containts(hero2)\t: " + listHero.contains(hero2));
System.out.println("   containts(heroAgility)\t: " + listHero.contains(heroAgility));
System.out.println("4. indexOf(hero1)\t: " + listHero.indexOf(hero1));
System.out.println("   indexOf(heroAgility)\t: " + listHero.indexOf(heroAgility));
```

Package

Package

- Saat kita membuat aplikasi, bisa dipastikan kita akan banyak sekali membuat class
- Jika class terlalu banyak, kadang akan menyulitkan kita untuk mencari atau mengklasifikasikan jenis-jenis class
- Java memiliki fitur package, yaitu fitur mirip folder / direktori, dimana kita bisa menyimpan class-class kita di dalam package
- Sama seperti folder / direktori, package juga bisa nested, kita bisa menggunakan tanda titik (.) untuk membuat nested package
- Ketika kita menyimpan class di dalam package, maka diatas file Java nya, kita wajib menyebutkan nama package nya

Tampilan Hirarki Project



Kode: Class Product

```java
package farhan.belajardata.data;

public class Product {
    protected String name;
    protected int price;

    Product(String name, int price){
        this.name = name;
        this.price = price;
    }
}
```

Kode: Class Main (1)

```java
package farhan.belajardata.data;

import java.util.ArrayList;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        Product product1 = new Product(name:"Pantene", price:25000);
        Product product2 = new Product(name:"Sensodyne", price:20000);
        Product product3 = new Product(name:"Pepsodent", price:15000);
        Product product4 = new Product(name:"Lite", price:12000);

        System.out.println(product1.name);
        System.out.println(product2.name);

        ArrayList<Product> listProduct = new ArrayList<>();

        System.out.println(x:"\nProduk dalam ArrayList: ");
        listProduct.add(product1);
        listProduct.add(product2);
        listProduct.add(product3);
        listProduct.add(product4);
```

Kode: Class Main (2)

```
        for(Product product:listProduct){
            System.out.println(product.name);
        }

        listProduct.remove(product2);

        System.out.println(x:"\nProduk setelah remove product2: ");
        for(Product product:listProduct){
            System.out.println(product.name);
        }

        listProduct.set(index:0,product2);

        System.out.println(x:"\nProduk setelah diubah ");
        for(Product product:listProduct){
            System.out.println(product.name);
        }
    }
}
```

Hirarki Package (Access Modifier)

Hirarki Package

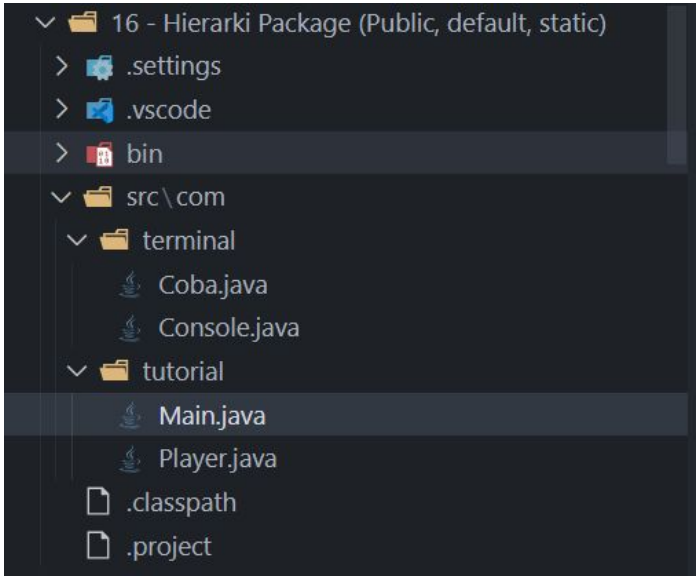
- Pada pertemuan sebelumnya kita sudah belajar tentang access modifier, sekarang kita akan belajar lebih dalam lagi tentang bagaimana perilaku access modifier pada class ketika terdapat package.
- Pada class yang memiliki access modifier default, maka class tersebut hanya bisa diakses oleh class lain yang satu package dengan class default tersebut.
- Pada class yang memiliki access modifier public, maka class tersebut bisa diakses oleh semua class pada semua package. Akan tetapi jika class lain diluar package ingin mengaksesnya harus menyertakan import ke lokasi package class yang dituju.
- Pada class yang memiliki access modifier private dan protected, class tersebut tidak bisa diakses oleh class lain dan class tersebut hanya berlaku untuk member dari suatu class, seperti field, method, atau nested class
- Dalam Java, class tidak dapat memiliki access modifier private. Hanya access modifier public, abstract, dan final yang dapat digunakan untuk class.

Import

Import

- Import adalah kemampuan untuk menggunakan class yang berada di package yang berbeda
- Syarat class yang bisa digunakan jika package nya berbeda adalah class yang harus public

Tampilan Hirarki Project



Kode: Import

```
// ini akan mengimport class Console
import com.terminal.Console;

// ini akan mengimport class Coba
import com.terminal.Coba;
```

Import Semua Package

- Jika kita ingin mengimport semua class di dalam sebuah package, kita bisa menggunakan tanda *, misal
- import com.terminal.*

Default Import

- Secara default, semua class yang ada di package java.lang sudah auto import, jadi kita tidak perlu melakukan import secara manual
- Contoh class String, Integer, Long, Boolean, dan lain-lain terdapat di package java.lang. Oleh karena itu, kita tidak perlu meng-import nya secara manual

Kode: Class Coba

```java
package com.terminal;

public class Coba {
    public static void log(String message){
        Terminal.log(message);
    }
}
```

Kode: Class Console

```java
package com.terminal;

// visibilitas untuk semuanya
public class Console {
    public static void log(String message){
        System.out.println(message);
    }
}

// ini hanya akan bisa diakses dari package com.terminal
class Terminal {
    public static void log(String message){
        System.out.println(message);
    }
}
```

Kode: Class Player

```java
package com.tutorial;

import com.terminal.Console;

// visibility default
class Player {
    private String name;

    Player(String name){
        this.name = name;
    }

    String getName(){
        return this.name;
    }

    void show(){
        // System.out.println("Player name : " + this.name);
        Console.log(message:"Menggunakan Console");
        Console.log("Player name : " + this.name);
    }
}
```

Kode: Class Main

```java
package com.tutorial;

// ini akan mengimport class Console
import com.terminal.Console;

// ini akan mengimport class Coba
import com.terminal.Coba;

// mengimport static method dari console;
import static com.terminal.Console.log;


class Main{
    Run | Debug
    public static void main(String[] args) {
        Player player1 = new Player(name:"Otong");
        player1.show();

        Console.log(message:"Menampilkan data dengan console.log");
        Console.log(player1.getName());

        Coba.log(message:"Hello");


        log(message:"Menampilkan data dengan log saja");
        log(player1.getName());
    }
}
```

Latihan

Manajemen Inventaris Toko Elektronik

Anda ditugaskan untuk membuat program manajemen inventaris untuk sebuah toko elektronik. Program ini akan menggunakan ArrayList untuk menyimpan daftar barang yang tersedia di toko. Setiap barang harus memiliki informasi seperti nama, harga, jumlah stok, dan deskripsi singkat. Program Anda harus mampu melakukan operasi berikut:

1. Menambahkan Barang Baru: Fungsi ini menerima nama barang, harga, jumlah stok, dan deskripsi, lalu menambahkannya sebagai barang baru ke dalam daftar.
2. Menghapus Barang: Berdasarkan nama barang, fungsi ini akan mencari barang dalam daftar dan menghapusnya. Jika ada lebih dari satu barang dengan nama yang sama, maka semua barang tersebut harus dihapus.
3. Mencari Barang: Diberikan nama barang, fungsi ini mencari semua barang yang memiliki nama tersebut dan menampilkannya. Jika ada lebih dari satu barang dengan nama yang sama, tampilkan semua.
4. Menampilkan Semua Barang: Fungsi ini menampilkan semua barang yang tersimpan dalam daftar.

Batasan dan Persyaratan

- Gunakan kelas Item untuk merepresentasikan sebuah barang. Kelas ini harus memiliki atribut nama, harga, jumlah stok, dan deskripsi.
- Gunakan ArrayList<Item> untuk menyimpan daftar barang.
- Implementasikan fungsi untuk setiap operasi yang disebutkan di atas dalam kelas InventoryManager.
- Buatlah validasi untuk memastikan bahwa harga dan jumlah stok yang dimasukkan saat menambahkan barang baru adalah angka positif.
- Buatlah sebuah kelas Main untuk mendemonstrasikan fungsionalitas dari InventoryManager dengan menyediakan menu interaktif bagi pengguna untuk menambahkan, menghapus, mencari, dan menampilkan semua barang.

Tampilan

```
Inventory Manager Menu:
1. Add Item
2. Remove Item
3. Search Item
4. Display All Items
5. Exit
Enter your choice:
```

Tampilan Add Item

```
Inventory Manager Menu:
1. Add Item
2. Remove Item
3. Search Item
4. Display All Items
5. Exit
Enter your choice: 1
Enter item name: Pantene
Enter item price: $30000
Enter item quantity: 20
Enter item description: Shampoo
Item added successfully.
```

Tampilan Display All Item

```
Inventory Manager Menu:
1. Add Item
2. Remove Item
3. Search Item
4. Display All Items
5. Exit
Enter your choice: 4
All Items:
Name: Pantene, Price: $30000.0, Quantity: 20, Description: Shampoo
```

Tampilan Search Item

```
Inventory Manager Menu:
1. Add Item
2. Remove Item
3. Search Item
4. Display All Items
5. Exit
Enter your choice: 3
Enter item name to search: Pantene
Name: Pantene, Price: $30000.0, Quantity: 20, Description: Shampoo
```

Tampilan Delete Item

```
Inventory Manager Menu:
1. Add Item
2. Remove Item
3. Search Item
4. Display All Items
5. Exit
Enter your choice: 2
Enter item name to remove: Pantene
Item removed successfully.
```