Contents

[Run Program 2](#_Toc166249607)

[Data Type 3](#_Toc166249608)

[Ternary 4](#_Toc166249609)

[Class and Object 5](#_Toc166249610)

[Access instance from class 5](#_Toc166249611)

[Method 6](#_Toc166249612)

[Method Overloading 6](#_Toc166249613)

[Array 8](#_Toc166249614)

[Multidimensional Array 8](#_Toc166249615)

# Run Program

Buat file bernama Hello.java

class Hello

{

    public static void main(String a[])

    {

        System.out.println("hello");

    }

}

Run di terminal

>javac Hello.java

dan

> java Hello

javac akan mengkompile java code kita menjadi binary yg extension nya .class

kemudian kita dapat melihat hasilnya dengan command java<nama class>

# Data Type

class Hello

{

    public static void main(String a[])

    {

        int num1 = 3;

        int num2 = 2;

        int result = num1 +num2;

        byte by = 127;

        short sh = 558;

        long lg = 5854l;

        float marks = 7.5f;

        double marks2 = 7.5;

        char k = 'k';

        String sentence = "hello world";

        boolean flag1 = true;

        boolean flag2 = false;

        //literals

        int num3 = 10\_000\_000;

        char c = 'a';

        c++;

    }

}

Output:

5

127

5854

=================

7.5

7.5

=================

k

hello world

=================

true

false

=================

10000000

B

# Ternary

class Hello

{

    public static void main(String a[])

    {

        int n = 5;

        int result = 0;

        result = n%2==0 ? 10 : 20;

        System.out.println(result);

    }

}

# Class and Object

class Calculator

{

    int a; //variabel

    public int add(int n1, int n2) // method return int

    {

        int r = n1 + n2;

        return r;

    }

}

public class Hello

{

    public static void main(String a[])

    {

        int num1=4;

        int num2=5;

        Calculator calc = new Calculator(); //new object / reference

        int result = calc.add(num1, num2);

        System.out.println(result);

    }

}

## Access instance from class

class Calculator

{

   int a = 8;

    public int add(int n1, int n2)

   {

        return n1 + n2;

   }

}

public class Hello

{

    public static void main(String a[])

    {

        Calculator obj = new Calculator();

        System.out.println(obj.a); // access instance / variable of class Calculator

    }

}

# Method

class Computer

{

    public void playMusic() // void method have no return

    {

        System.out.println("Music Playing");

    }

    public String getMeAPen(int cost) //string method

    {

        return "Pen";

    }

}

public class Hello

{

    public static void main(String a[])

    {

        Computer com = new Computer();

        com.playMusic();

        String str = com.getMeAPen(10);

        System.out.println(str);

    }

}

## Method Overloading

class Calculator

{

   public int add(int n1, int n2)

   {

        return n1 + n2;

   }

   public int add(int n1, int n2, int n3)

   {

        return n1 + n2 + n3;

   }

   public double add(double n1, int n2)

   {

        return n1+n2;

   }

}

public class Hello

{

    public static void main(String a[])

    {

        Calculator obj = new Calculator();

        int r1 = obj.add(3,4);

        int r2 = obj.add(3,4, 5);

        System.out.println(r1);

        System.out.println(r2);

    }

}

Perhatikan untuk method double add(), double add() memiliki 2 param sama dengan add(int n1, int n2), tapi tipe data param nya berbeda, yg satu int dan int, yg satu double dan int

# Looping (enchance for/foreach)

public static void main(String a[])

    {

        int nums[] = new int[4];

        nums[0] = 7;

        nums[1] = 3;

        nums[2] = 2;

        nums[3] = 9;

        for(int value: nums)

        {

            System.out.println(value);

        }

    }

# Array

public class Hello

{

    public static void main(String a[])

    {

        int nums[] = {5,6,7};

        nums[2] = 8;

        int dnums[] = new int[4]; // new empty dynamic array of size 4

        dnums[1] = 12;

        System.out.println(nums[1]); //6

        System.out.println(nums[2]); //8

        System.out.println(dnums[0]); //0

        System.out.println(dnums[1]); //12

        //looping array

        for(int i = 0; i<3 ; i++){

            System.out.println(nums[i]);

        }

    }

}

## Multidimensional Array

public class Hello

{

    public static void main(String a[])

    {

        int nums [][] = new int [3][4]; // 3 rows, 4 colums

        for(int i=0; i<3; i++)

        {

            for(int j=0; j<4; j++)

            {

                nums[i][j] = (int)(Math.random() \* 10); // assigned random number into array

                System.out.print(nums[i][j] + " ");

            }

                System.out.println();

        }

        // Loop Array using enhance loop / foreach

        for(int n[]: nums) // assign n as array with value of nums

        {

            for(int m: n) // assign m with value of n

            {

                System.out.print(m + " ");

            }

            System.out.println();

        }

    }

}

## Array of Object

class Student

{

    int rollno;

    String name;

    int marks;

}

public class Hello

{

    public static void main(String a[])

    {

        Student s1 = new Student();

        s1.rollno = 1;

        s1.name = "Navia";

        s1.marks = 88;

        Student s2 = new Student();

        s2.rollno = 2;

        s2.name = "Candance";

        s2.marks = 78;

        Student s3 = new Student();

        s3.rollno = 3;

        s3.name = "John";

        s3.marks = 68;

        Student students[] = new Student[3];// Define 1 Array of Object with 3 length

        students[0] = s1;

        students[1] = s2;

        students[2] = s3;

        for (int i=0; i < students.length; i++)

        {

            System.out.println(students[i].name + " : " + students[i].marks );

        }

 //atau menggunakan foreach/enchance foreach

        for(Student objValue : students)

        {

            System.out.println(objValue.name + " : " + objValue.marks );

        }

    }

}

# Immutable String & Mutable String

Immutable = tidak bisa di ubah

Mutable = bisa diubah

By default String merupakan Immutable

Jika kita tetap ingin mengubah string, bisa menggunakan StringBuffer dan StringBuilder

## StringBuffer

public static void main(String a[])

    {

        StringBuffer sb =new StringBuffer("Navin");

        sb.append(" ready");

        System.out.println(sb);

    }

# Static Variable

Menjadikan instance/variable dalam scoop class, bukan lagi menjadi scoop object

Contoh scoop object

class Mobile

{

    String brand;

    int price;

    String name;

    public void show()

    {

        System.out.println(brand + " : " + price + " : " + name);

    }

}

public class Hello

{

    public static void main(String a[])

    {

        Mobile obj1 = new Mobile();

        obj1.brand = "Apple";

        obj1.price = 1500;

        obj1.name = "Apple Phone";

        Mobile obj2 = new Mobile();

        obj2.brand = "Samsung";

        obj2.price = 1100;

        obj2.name = "Samsung Phone";

        obj1.name = "My Apple Phone";

       obj1.show(); // Apple : 1500 : My Apple Phone

        obj2.show(); // Samsung : 1100 : Samsung Phone

    }

}

Pada contoh diatas kita mengubah name dari obj1 menjadi My Apple Phone, dan yg berubah hanya name pada obj1

Contoh scoop class menggunakan static (static String name):

class Mobile

{

    String brand;

    int price;

    static String name;

    public void show()

    {

        System.out.println(brand + " : " + price + " : " + name);

    }

}

public class Hello

{

    public static void main(String a[])

    {

        Mobile obj1 = new Mobile();

        obj1.brand = "Apple";

        obj1.price = 1500;

        obj1.name = "Apple Phone";

        Mobile obj2 = new Mobile();

        obj2.brand = "Samsung";

        obj2.price = 1100;

        obj2.name = "Samsung Phone";

        obj1.name = "My Apple Phone";

        obj1.show(); // Apple : 1500 : My Apple Phone

        obj2.show(); // Samsung : 1100 : My Apple Phone

    }

}

Disini meski kita hanya mengganti name di obj1, name di obj2 juga ikut berubah, karena name telah menjadi static variabel(class scoope). Obj1 dan obj2 merupakan objek dari class yg sama (Mobile)

Penulisan static yg benar untuk object yg ingin mengakses nya adalah <Nama class>.instance

Contoh yg benar:

class Mobile

{

    String brand;

    int price;

    static String name;

    public void show()

    {

        System.out.println(brand + " : " + price + " : " + name);

    }

}

public class Hello

{

    public static void main(String a[])

    {

        Mobile obj1 = new Mobile();

        obj1.brand = "Apple";

        obj1.price = 1500;

        Mobile.name = "Apple Phone";

        Mobile obj2 = new Mobile();

        obj2.brand = "Samsung";

        obj2.price = 1100;

        Mobile.name = "Samsung Phone";

        Mobile.name = "My Apple Phone";

        obj1.show(); // Apple : 1500 : My Apple Phone

        obj2.show(); // Samsung : 1100 : My Apple Phone

    }

}

# Static block

class Mobile

{

    String brand;

    int price;

    static String name;

    static {

        name = "phone";

        System.out.println("this is static block");

    }

    public Mobile(){ //constructor

        brand = "";

        price = 200;

        System.out.println("in constructor");

    }

}

public class Hello

{

    public static void main(String a[])

    {

       Mobile obj1 = new Mobile();

      Mobile obj2 = new Mobile();

    }

}

Output:

this is static block

in constructor

in constructor

Static akan di jalankan saat object memanggil class dan hanya di jalankan 1x, constructor akan di panggil setelahnya dan akan di panggil ulang setiap membuat object baru

## Call Static Block without create object

class Mobile

{

    String brand;

    int price;

    static String name;

    static {

        name = "phone";

        System.out.println("this is static block");

    }

    public Mobile(){ //constructor

        brand = "";

        price = 200;

        System.out.println("in constructor");

    }

}

public class Hello

{

    public static void main(String a[]) throws ClassNotFoundException

    {

        Class.forName("Mobile");

    }

}

# Static Methods

class Mobile

{

    String brand;

    int price;

    static String name;

    public static void show1(Mobile obj)

    {

        System.out.println("this is static methods");

        System.out.println(obj.brand + " : " + name); // apple : null

    }

}

public class Hello

{

    public static void main(String a[])

    {

        Mobile obj1 = new Mobile();

        obj1.brand = "apple";

        Mobile.show1(obj1);

    }

}

Kita hanya bisa memanggil variabel static di dalam method static, agar bisa mengkases variabel non static, dapat di lakukan dengan kirim parameter

# Encapsulation(private)

## Private

Hanya bisa di akses di class yg sama, atau menggunakan method

class Human

{

    private int age;

    private String name;

    public int getAge()

    {

        return this.age; // same class, jadi bisa akses private

    }

    public String getName()

    {

        return this.name;

    }

    public void setAge(int a)

    {

        this.age = a;

    }

    public void setName(String b)

    {

        this.name = b;

    }

}

public class Hello

{

    public static void main(String a[])

    {

        Human obj = new Human();

        // obj.age = 30; // error krn akses private langsung

        obj.setAge(30);

        obj.setName("Navin");

        System.out.println(obj.getName() + " : " + obj.getAge());

    }

}

# Constructors

class Human

{

    private int age;

    private String name;

    public Human(){ //default constructor

        this.age = 12;

        this.name = "cancdance";

    }

    public Human(int a, String n){ //parameter constructor

        this.age = a;

        this.name = n;

    }

    public int getAge()

    {

        return this.age; // same class, jadi bisa akses private

    }

    public String getName()

    {

        return this.name;

    }

    public void setAge(int a)

    {

        this.age = a;

    }

    public void setName(String b)

    {

        this.name = b;

    }

}

public class Hello

{

    public static void main(String a[])

    {

        Human obj = new Human();

        Human obj2 = new Human(17, "diluc");

        // obj.age = 30; // error krn akses private langsung

        System.out.println(obj.getName() + " : " + obj.getAge());

        System.out.println(obj2.getName() + " : " + obj2.getAge());

    }

}

# Super() dan this() method

Pada super()

class A

{

    public A(){

        super();

        System.out.println(" this in A");

    }

    public A(int a){

        super();

        System.out.println(" this in A int " + a);

    }

}

class B extends A

{

    public B(){

        super();

        System.out.println(" this is B");

    }

    public B(int b){

        super();

        System.out.println(" this is B int");

    }

}

public class Hello

{

    public static void main(String a[])

    {

        B obj = new B(5);

    }

}

Method super() ada di dalam constructor, meski kita tidak menulisnya. Method super akan memanggil constructor dari super class nya

Pada contoh diatas Class B extends Class A, artinya B adalah SubClass, A adalah SuperClass(parent)

Saat kita membuat new object dari B, akan di jalankan terlebih dahulu constructor di super class nya (A) sesuai dengan parameter nya, baru menjalankan constructor di sub class nya(B) sesuai dengan param/tidak

Pada this()

class A

{

    public A(){

        super();

        System.out.println(" this in A");

    }

    public A(int a){

        super();

        System.out.println(" this in A int " + a);

    }

}

class B extends A

{

    public B(){

        super();

        System.out.println(" this is B");

    }

    public B(int b){

        this(); // menjalankan constructor default pada class yg sama(class B)

        System.out.println(" this is B int");

    }

}

public class Hello

{

    public static void main(String a[])

    {

        B obj = new B(5);

    }

}

Outputnya:

this in A

this is B

this is B int