

1. public class Asgdll {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

/\* Kamus \*/

float f= 20.0f;

double fll;

/\* Algoritma \*/

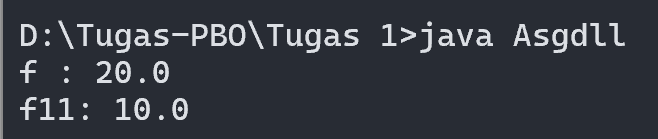
fll=10.0f;

System.out.println ("f : "+f +

"\nf11: "+fll);

}

}

Output :

Penjelasan :

Program ini digunakan untuk menyimpan serta menampilkan angka desimal. Di dalamnya, ada dua variabel: f yang bertipe float dengan nilai 20.0, dan fll yang bertipe double dengan nilai 10.0. Float digunakan untuk angka desimal dengan ukuran lebih kecil, sedangkan double bisa menyimpan angka desimal dengan lebih banyak angka di belakang koma. Setelah itu, program mencetak nilai kedua variabel tersebut ke layar menggunakan System.out.println(), sehingga hasil yang ditampilkan adalah angka 20.0 dan 10.0 dalam format yang mudah dibaca.

1. public class Asign {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

/\* Kamus \*/

int i;

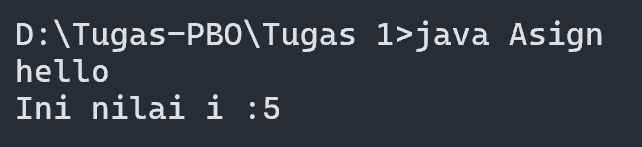
/\* Program \*/

System.out.print ("hello\n"); i = 5;

System.out.println ("Ini nilai i :" + i);

}

}

Output :

Penjelasan :

Program ini bertujuan untuk menampilkan teks ke layar dan mendeklarasikan sebuah variabel. Di dalamnya, terdapat variabel i yang bertipe int (bilangan bulat), lalu program mencetak kata **"hello"** diikuti oleh baris baru (\n). Setelah itu, variabel i diberi nilai 5, dan program mencetak teks **"Ini nilai i : 5"**. Fungsi utama dari program ini adalah menunjukkan cara mendeklarasikan variabel, memberikan nilai, serta mencetak teks dan variabel ke layar menggunakan System.out.print() dan System.out.println().

1. /\* Deskripsi : \*/

/\* Program ini berisi contoh sederhana untuk mendefinisikan \*/

/\* variabel-variabel bilangan bulat (short int, int, long int), \*/

/\* karakter, bilangan riil, \*/

public class ASIGNi {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

/\* KAMUS \*/

short ks = 1;

int ki = 1;

long kl = 10000;

char c = 65; /\* inisialisasi karakter dengan

integer \*/

char c1 = 'Z'; /\* inisialisasi karakter dengan karakter \*/

double x = 50.2f;

float y = 50.2f;

/\* Algoritma \*/

/\* penulisan karakter sebagai karakter \*/

System.out.println ("Karakter = "+ c);

System.out.println ("Karakter = "+ c1);

/\* penulisan karakter sebagai integer \*/

System.out.println ("Karakter = "+ c);

System.out.println ("Karakter = "+ c1);

System.out.println ("Bilangan integer (short) = "+ ks);

System.out.println ("\t(int) = "+ ki);

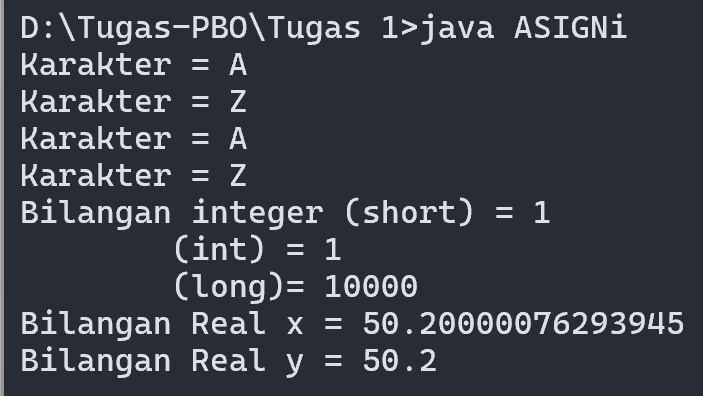
System.out.println ("\t(long)= "+ kl);

System.out.println ("Bilangan Real x = "+ x);

System.out.println ("Bilangan Real y = "+ y);

}

}

Output :

Penjelasan :

Program ini mendemonstrasikan cara mendeklarasikan berbagai jenis variabel, termasuk bilangan bulat (short, int, long), karakter (char), dan bilangan desimal (double, float). Variabel ks, ki, dan kl digunakan untuk menyimpan bilangan bulat dengan ukuran berbeda, sedangkan c dan c1 digunakan untuk menyimpan karakter, baik dalam bentuk angka ASCII (65 untuk huruf 'A') maupun langsung sebagai huruf ('Z'). Variabel x dan y menyimpan bilangan desimal dengan tipe double dan float. Program ini kemudian mencetak nilai dari semua variabel tersebut ke layar menggunakan System.out.println(), menampilkan karakter sebagai huruf dan angka, serta menampilkan nilai bilangan bulat dan desimal yang telah dideklarasikan.

1. import java.util.Scanner;

/\* contoh membaca integer menggunakan Class Scanner\*/

public class BacaData {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

/\* Kamus \*/

int a;

Scanner masukan;

/\* Program \*/

System.out.print ("Contoh membaca dan menulis, ketik nilai integer: \n");

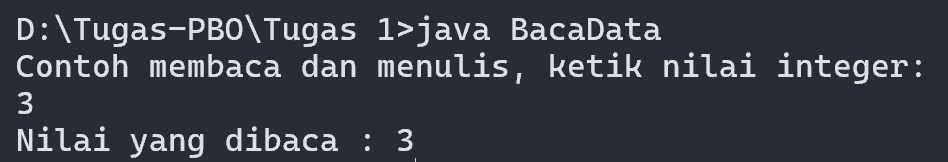
masukan = new Scanner(System.in);

a = masukan.nextInt(); /\* coba ketik : masukan.nextInt(); ; Apa akibatnya ?\*/

System.out.print ("Nilai yang dibaca : "+ a);

}

}

Output :

Penjelasan :

Program ini menggunakan **Class Scanner** untuk membaca input dari pengguna dan menampilkan kembali nilai yang dimasukkan. Pertama, program mendeklarasikan variabel a sebagai bilangan bulat (int) dan membuat objek Scanner dengan nama masukan untuk membaca input dari keyboard. Program kemudian meminta pengguna memasukkan sebuah angka dengan menampilkan pesan di layar. Setelah pengguna mengetik angka dan menekan **Enter**, angka tersebut disimpan ke dalam variabel a menggunakan masukan.nextInt(). Terakhir, program mencetak kembali angka yang telah dibaca dengan menampilkan pesan **"Nilai yang dibaca : [angka yang dimasukkan]"**. Jika masukan.nextInt(); dihapus, maka program tidak akan bisa membaca input dari pengguna, sehingga variabel a tidak akan memiliki nilai yang diberikan dari input.

1. import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Bacakar {

public static void main(String[] args) throws IOException {

// Kamus

char cc;

int bil;

// Membuat objek BufferedReader untuk input

InputStreamReader isr = new InputStreamReader(System.in);

BufferedReader dataIn = new BufferedReader(isr);

BufferedReader datAIn = new BufferedReader(new InputStreamReader(System.in));

// Algoritma

System.out.print("hello\n");

// Membaca satu karakter

System.out.print("baca 1 karakter : ");

cc = dataIn.readLine().charAt(0);

// Membaca satu bilangan

System.out.print("baca 1 bilangan : ");

bil = Integer.parseInt(datAIn.readLine());

// Menampilkan hasil

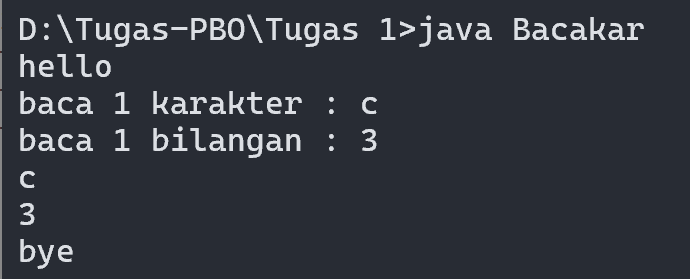
System.out.println(cc);

System.out.println(bil);

System.out.println("bye");

}

}



Output :

Penjelasan :

Program ini menggunakan **Class Scanner** untuk membaca input dari pengguna dan menampilkan kembali nilai yang dimasukkan. Pertama, program mendeklarasikan variabel a sebagai bilangan bulat (int) dan membuat objek Scanner dengan nama masukan untuk membaca input dari keyboard. Program kemudian meminta pengguna memasukkan sebuah angka dengan menampilkan pesan di layar. Setelah pengguna mengetik angka dan menekan **Enter**, angka tersebut disimpan ke dalam variabel a menggunakan masukan.nextInt(). Terakhir, program mencetak kembali angka yang telah dibaca dengan menampilkan pesan **"Nilai yang dibaca : [angka yang dimasukkan]"**. Jika masukan.nextInt(); dihapus, maka program tidak akan bisa membaca input dari pengguna, sehingga variabel a tidak akan memiliki nilai yang diberikan dari input.

1. /\*Casting menggunakan tipe data primitif\*/

public class Casting1 {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

int a=5,b=6;

float d=2.f,e=3.2f;

char g='5';

double k=3.14;

System.out.println((float)a); // int <-- float

System.out.println((double)b); // int <-- double

System.out.println((int)d); // float <-- int

System.out.println((double)e); // float <-- double

System.out.println((int)g); // char <-- int (ASCII)

System.out.println((float)g); // char <-- float (ASCII)

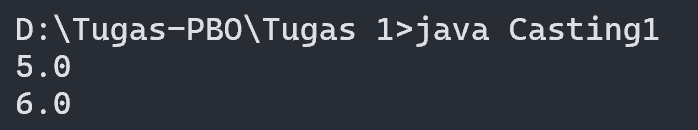
System.out.println((double)g); // char <-- double (ASCII)

System.out.println((int)k); // double <-- int

System.out.println((float)k); // double <-- float

}

}

Output :

Penjelasan :

Program ini menunjukkan **casting** atau konversi tipe data primitif, yaitu mengubah satu jenis tipe data menjadi tipe lain. Dalam program ini, terdapat variabel dengan berbagai tipe data seperti int, float, char, dan double. Program kemudian mencetak hasil konversi dari satu tipe ke tipe lainnya. Misalnya, angka a bertipe int dikonversi ke float, dan b bertipe int dikonversi ke double. Variabel d bertipe float dikonversi ke int, sementara e dikonversi ke double. Karakter g yang berisi '5' dikonversi ke int, float, dan double, yang akan menghasilkan nilai ASCII dari karakter tersebut. Variabel k yang bertipe double juga dikonversi ke int dan float. Program ini membantu memahami bagaimana tipe data bisa diubah dalam Java, terutama dalam operasi matematika dan pemrosesan data.

1. /\*Casting menggunakan tipe data Class\*/

public class Casting2 {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

int a=8,b=9;

float d=2.f,e=3.2f;

char g='5';

double k=3.14;

String n="67",m="45", l="100";

a = Integer.parseInt(n); /\*Konversi String ke Integer\*/

k = Double.parseDouble(m); /\*Konversi String ke Double\*/

d = Float.parseFloat(l); /\*Konversi String ke Float\*/

System.out.println("a : "+a+"\nk : "+k+"\nd : "+d);

n = String.valueOf(b); /\*Konversi Integer ke String\*/

m = String.valueOf(g); /\*Konversi Karakter ke String\*/

l = String.valueOf(e); /\*Konversi Float ke String\*/

System.out.println("n : "+n+"\nm : "+m+"\nl : "+l);

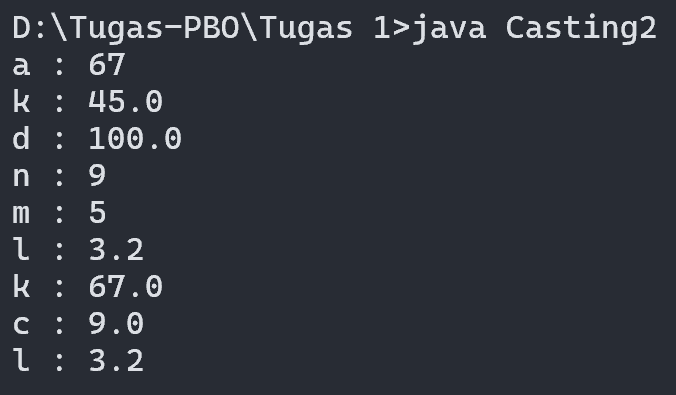
k = Double.valueOf(a).intValue(); /\*Konversi Integer ke

Double\*/

double c = Integer.valueOf(b).doubleValue();

System.out.println("k : "+k+"\nc : "+c+"\nl : "+l);

}

}

Output :

Penjelasan :

Program ini mendemonstrasikan **casting menggunakan tipe data kelas (Class Casting)**, yaitu konversi antar tipe data dengan bantuan method dari kelas bawaan Java seperti Integer, Double, Float, dan String.

Pada bagian pertama, program mengonversi tipe **String ke angka**. Variabel n, m, dan l yang berisi nilai dalam bentuk teks ("67", "45", dan "100") dikonversi ke tipe data numerik menggunakan metode berikut:

* Integer.parseInt(n): mengubah String "67" menjadi bilangan bulat int.
* Double.parseDouble(m): mengubah String "45" menjadi bilangan desimal double.
* Float.parseFloat(l): mengubah String "100" menjadi bilangan desimal float.

Pada bagian kedua, program mengonversi angka dan karakter ke **String** menggunakan String.valueOf():

* String.valueOf(b): mengubah bilangan bulat b menjadi String.
* String.valueOf(g): mengubah karakter g menjadi String.
* String.valueOf(e): mengubah angka float menjadi String.

Bagian terakhir program melakukan konversi antara **Integer dan Double** menggunakan method Double.valueOf().intValue() dan Integer.valueOf().doubleValue(), yang mengubah tipe data sesuai kebutuhan.

Setelah melakukan berbagai konversi, program mencetak hasilnya ke layar menggunakan System.out.println(). Program ini sangat berguna untuk memahami cara mengubah tipe data dalam Java, terutama saat bekerja dengan input dari pengguna atau membaca data dari file yang sering berbentuk String tetapi perlu dikonversi ke tipe angka sebelum digunakan.

1. /\* pemakaian operator kondisional \*/

public class Ekspresi {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

/\* KAMUS \*/

int x = 1;

int y = 2;

/\* ALGORITMA \*/

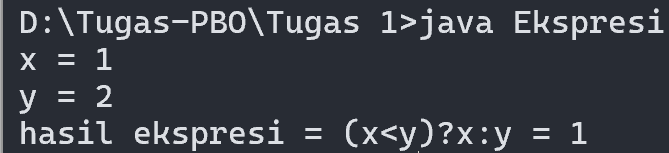
System.out.print("x = "+ x + "\n");

System.out.print("y = "+ y + "\n");

System.out.print("hasil ekspresi = (x<y)?x:y = "+ ((x < y) ? x : y)); /\*Gunakan dalam kurung "(statemen dan kondisi)" untuk menyatakan satu kesatuan pernyataan\*/

}

}

Output :

Penjelasan :

Program Java ini mendemonstrasikan **operator kondisional (ternary operator)**, yang digunakan untuk membuat keputusan sederhana dalam satu baris kode.

Pertama, program mendeklarasikan dua variabel bilangan bulat, yaitu x = 1 dan y = 2. Kemudian, program mencetak nilai x dan y ke layar. Setelah itu, program menggunakan **operator ternary** (x < y) ? x : y, yang berarti:

* Jika x lebih kecil dari y, maka hasilnya adalah x.
* Jika tidak, hasilnya adalah y.

Karena x (1) memang lebih kecil dari y (2), maka hasil ekspresi ini adalah 1. Program ini menunjukkan cara sederhana untuk memilih nilai berdasarkan kondisi tanpa menggunakan if-else, sehingga kode menjadi lebih ringkas dan mudah dibaca.

1. /\* pembagian integer, casting \*/

public class Ekspresi1 {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

/\* KAMUS \*/

int x = 1; int y = 2; float fx; float fy;

/\* ALGORITMA \*/

System.out.print ("x/y (format integer) = "+ x/y);

System.out.print ("\nx/y (format float) = "+ x/y);

/\* supaya hasilnya tidak nol \*/

fx=x;

fy=y;

System.out.print ("\nx/y (format integer) = "+ fx/fy);

System.out.print ("\nx/y (format float) = "+ fx/fy);

/\* casting \*/

System.out.print ("\nfloat(x)/float(y) (format integer) = "+ (float)x/(float)y);

System.out.print ("\nfloat(x)/float(y) (format float) = "+ (float)x/(float)y);

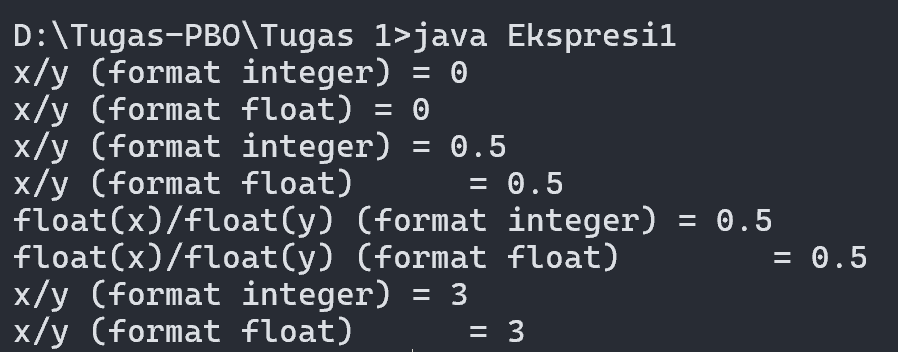
x = 10; y = 3;

System.out.print ("\nx/y (format integer) = "+ x/y);

System.out.print ("\nx/y (format float) = "+ x/y);

}

}

Output :

Penjelasan :

Program di mendemonstrasikan pembagian antara bilangan bulat (integer) dan bagaimana hasilnya bisa berbeda ketika menggunakan tipe data float melalui casting.

Pertama, program mendeklarasikan dua variabel integer, yaitu x = 1 dan y = 2. Kemudian, program mencoba mencetak hasil pembagian x/y dengan format integer. Karena x dan y adalah bilangan bulat, hasil pembagian 1/2 akan dibulatkan ke bawah menjadi 0. Selanjutnya, program mencoba mencetak hasil pembagian yang sama dengan format float, tetapi tetap menggunakan tipe data integer, sehingga hasilnya tetap 0.

Untuk menghindari hasil 0, program mengonversi x dan y ke tipe data float dengan membuat variabel fx dan fy, yang diberi nilai dari x dan y. Setelah itu, pembagian fx/fy dilakukan, sehingga hasilnya 0.5. Selanjutnya, program menggunakan **casting** eksplisit dengan (float)x/(float)y untuk memastikan bahwa x dan y dianggap sebagai float sebelum pembagian dilakukan, yang juga menghasilkan 0.5.

Kemudian, program mengubah nilai x menjadi 10 dan y menjadi 3. Ketika pembagian dilakukan dalam format integer (10/3), hasilnya adalah 3 karena desimal dibuang. Namun, jika pembagian dilakukan dalam format float, hasilnya tetap 3 karena masih menggunakan tipe data integer.

Kesimpulannya, program ini menunjukkan perbedaan antara pembagian integer dan float, serta bagaimana casting ke float dapat memberikan hasil yang lebih akurat dalam operasi pembagian.

1. public class Hello {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

/\* menuliskan hello ke layar \*/

System.out.print("Hello");

/\* menuliskan hello dan ganti baris\*/

System.out.print("\nHello ");

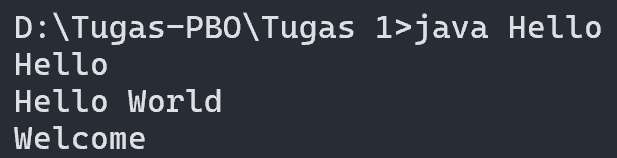
/\* menuliskan hello dan ganti baris\*/

System.out.println("World");

System.out.println("Welcome");

}

}

Output :

Penjelasan:

Program di atas adalah program sederhana yang mencetak beberapa teks ke layar. Program ini menggunakan metode System.out.print dan System.out.println untuk menampilkan output.

Pertama, program mencetak kata **"Hello"** menggunakan System.out.print("Hello");. Perintah print hanya menampilkan teks tanpa pindah ke baris baru. Kemudian, program mencetak **"Hello "** lagi menggunakan System.out.print("\nHello ");. Tanda \n digunakan untuk membuat baris baru sebelum mencetak "Hello".

Selanjutnya, program menggunakan System.out.println("World");. Berbeda dengan print, perintah println mencetak teks lalu otomatis pindah ke baris baru. Jadi, teks "World" akan muncul di baris baru setelah "Hello".

Terakhir, program mencetak **"Welcome"** menggunakan System.out.println("Welcome");, yang juga akan pindah ke baris baru setelah mencetak teks.

Secara keseluruhan, program ini mendemonstrasikan cara menampilkan teks di Java dengan dan tanpa pindah ke baris baru.

1. /\* Effek dari operator ++ \*/

public class Incr {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

/\* Kamus \*/

int i, j;

/\* Program \*/

i = 3;

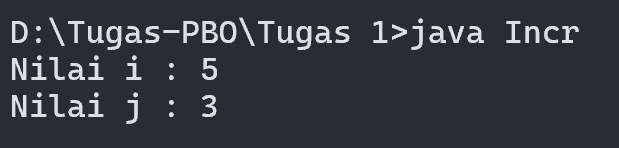
j = i++;

System.out.println ("Nilai i : " + (++i) +

"\nNilai j : " + j);

}

}

Output :

Penjelasan :

Program di atas adalah program Java yang menunjukkan efek dari **operator increment (++)**, yaitu bagaimana nilai sebuah variabel bertambah saat menggunakan **post-increment (i++)** dan **pre-increment (++i)**.

Pertama, program mendeklarasikan dua variabel integer, yaitu i dan j. Kemudian, i diberi nilai 3, dan j diberi nilai i++. Pada bagian ini, karena i++ menggunakan **post-increment**, nilai i yang lama (3) akan disimpan ke j, lalu i baru ditingkatkan menjadi 4.

Setelah itu, program mencetak nilai i dan j. Pada perintah System.out.println, terdapat ++i, yang berarti **pre-increment**. Dalam pre-increment, nilai i akan ditingkatkan terlebih dahulu sebelum digunakan. Karena sebelumnya i bernilai 4, setelah ++i, nilainya menjadi 5.

1. /\* pemakaian beberapa operator terhadap bit \*/

public class Oper1 {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

/\* KAMUS \*/

int n = 10; /\* 1010 \*/

int x = 1; /\* 1 \*/

int y = 2; /\* 10 \*/

/\* ALGORITMA \*/

System.out.println ("n = "+ n);

System.out.println ("x = "+ x);

System.out.println ("y = "+ y);

System.out.println ("n & 8 = "+ (n & 8)); /\* 1010 AND 1000 \*/

System.out.println ("x & ~ 8 = "+ (x & ~8)); /\* 1 AND

0111 \*/

System.out.println ("y << 2 = "+ (y << 2)); /\* 10 ==>

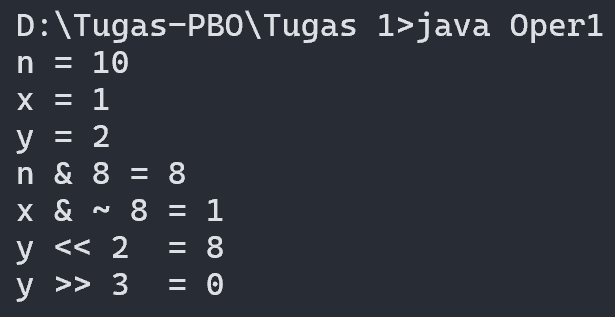
1000 = 8 \*/

System.out.println ("y >> 3 = "+ (y >>3)); /\* 10 ==>

0000 = 0 \*/

}

}

Output :

Penjelasan :

Program di atas adalah program Java yang mendemonstrasikan penggunaan **operator bitwise**, yaitu operasi yang bekerja pada level bit dari bilangan biner.

Pertama, program mendeklarasikan tiga variabel integer:

* n = 10 (bentuk biner: 1010)
* x = 1 (bentuk biner: 0001)
* y = 2 (bentuk biner: 0010)

Kemudian, program mencetak nilai dari masing-masing variabel. Setelah itu, program melakukan beberapa operasi bitwise:

1. **n & 8** → Operasi **AND bitwise (&)** antara n (1010) dan 8 (1000).
   * Hasilnya: 1010 & 1000 = 1000 (dalam desimal: 8).
2. **x & ~8** → Operasi **AND** antara x (0001) dan **komplemen bitwise (~) dari 8**, yaitu 0111.
   * Hasilnya: 0001 & 0111 = 0001 (dalam desimal: 1).
3. **y << 2** → Operasi **left shift (<<)**, yaitu menggeser bit y (0010) ke kiri sebanyak 2 posisi.
   * Hasilnya: 0010 menjadi 1000 (dalam desimal: 8).
4. **y >> 3** → Operasi **right shift (>>)**, yaitu menggeser bit y (0010) ke kanan sebanyak 3 posisi.

* Hasilnya: 0010 menjadi 0000 (dalam desimal: 0).

Secara keseluruhan, program ini menunjukkan bagaimana operator bitwise bekerja di Java dengan melakukan operasi **AND, NOT, left shift, dan right shift** untuk memanipulasi bit dari angka biner.

1. /\* pemakaian beberapa operator terhadap RELATIONAL DAN bit \*/

public class Oper2 {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

/\* KAMUS \*/

char i, j;

/\* ALGORITMA \*/

i = 3; /\* 00000011 dalam biner \*/

j = 4; /\* 00000100 dalam biner \*/

System.out.println("i = "+ (int) i);

System.out.println("j = "+ j);

System.out.println("i & j = "+ (i & j)); /\* 0: 00000000 dalam

biner \*/

System.out.println("i | j = "+ (i | j)); /\* 7:

00000111 biner \*/

System.out.println("i ^ j = "+ (i ^ j)); /\* 7:

00000111 biner Ingat!!! operator "^" pada bahasa java bukan sebagai pangkat\*/

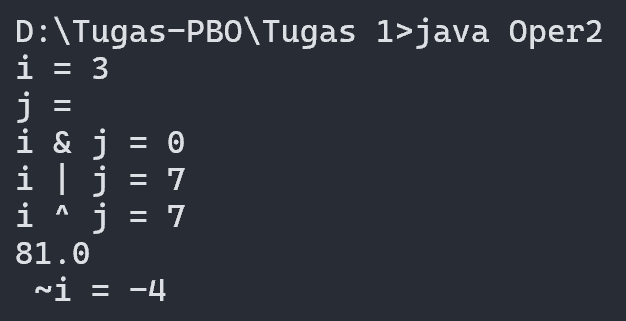
System.out.println(Math.pow(i, j)); /\* Class Math memiliki method pow(a,b) untuk pemangkatan\*/

System.out.println(" ~i = "+ ~i); /\* -4: 11111100

biner \*/

}

}

Output :

Penjelasan :

Program di atas merupakan contoh penggunaan **operator bitwise dan relasional** pada tipe data **char** di Java. Meskipun i dan j bertipe **char**, mereka diperlakukan sebagai bilangan bulat saat dilakukan operasi bitwise. Program ini mendeklarasikan dua variabel, yaitu i = 3 dan j = 4, yang dalam bentuk biner masing-masing direpresentasikan sebagai 00000011 dan 00000100.

Pertama, program mencetak nilai i dan j. Karena i bertipe **char**, maka perlu dikonversi ke integer dengan (int) i agar tampil sebagai angka, bukan sebagai karakter ASCII. Selanjutnya, program melakukan beberapa operasi bitwise. Operasi **AND (i & j)** menghasilkan 0 karena tidak ada bit yang sama-sama bernilai 1 pada kedua bilangan. Operasi **OR (i | j)** menghasilkan 7 (00000111 dalam biner) karena semua bit yang bernilai 1 dalam salah satu angka akan tetap 1. Kemudian, **XOR (i ^ j)** juga menghasilkan 7 (00000111 dalam biner), di mana bit yang berbeda diubah menjadi 1.

Program juga menggunakan **fungsi Math.pow(i, j)** untuk menghitung i pangkat j, yaitu 3^4 = 81. Terakhir, program mendemonstrasikan operator **NOT (~i)**, yang mengubah semua bit dalam bilangan i menjadi kebalikannya. Hasilnya adalah -4, karena dalam sistem bilangan **two’s complement**, operasi NOT membalik bit sekaligus mengubah tanda angka. Secara keseluruhan, program ini menunjukkan bagaimana **operator bitwise** bekerja dan bagaimana pemangkatan dilakukan menggunakan **Math.pow()**.

1. public class Oper3 {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

/\* Algoritma \*/

if (true && true){ System.out.println(true && true); }

/\* true = true and true \*/

if (true & true) { System.out.println(true & false); } /\* true & true \*/

if (true) { System.out.println(true); } /\* true

\*/

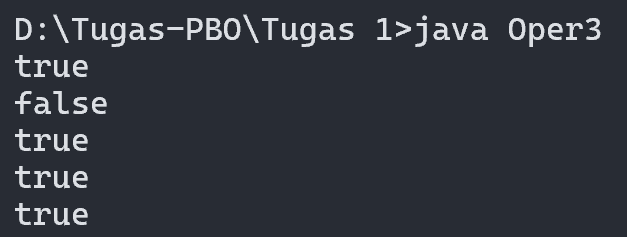
if (true || true){ System.out.println(true); } /\* true

= true or true \*/

if (true|false) { System.out.println(true|false); } /\* true|false \*/

}

}

Output :

Penjelasan :

Program di atas adalah program yang mendemonstrasikan penggunaan **operator logika (&&, ||)** dan **operator bitwise (&, |)** dalam pernyataan kondisional (if). Program ini mengevaluasi berbagai kondisi dan mencetak hasilnya jika kondisinya bernilai **true**.

Pertama, program mengevaluasi **true && true**, di mana && adalah **operator AND logis** yang hanya bernilai **true** jika kedua operand juga **true**. Karena kondisi ini benar, program mencetak **true**. Kemudian, program mencoba mengevaluasi **true & true**, di mana & adalah **operator AND bitwise**. Namun, terdapat kesalahan dalam kode karena perintah cetak menggunakan true & false, bukan true & true. Seharusnya, jika ditulis dengan benar, hasilnya adalah **true**, tetapi karena true & false bernilai **false**, program tidak mencetak apapun.

Selanjutnya, program mengevaluasi pernyataan if (true), yang selalu benar, sehingga mencetak **true**. Setelah itu, program mengevaluasi **true || true**, di mana || adalah **operator OR logis** yang akan bernilai **true** jika setidaknya salah satu operand bernilai **true**. Karena kondisi ini benar, program mencetak **true**. Terakhir, program mengevaluasi **true | false**, di mana | adalah **operator OR bitwise** yang bekerja seperti OR logis dalam kasus boolean. Karena true | false menghasilkan **true**, program mencetak **true**.

Kesimpulannya, program ini menunjukkan perbedaan antara **operator logis (&&, ||)** yang digunakan dalam ekspresi boolean, dan **operator bitwise (&, |)** yang bekerja pada level bit tetapi juga dapat digunakan untuk operasi boolean.

1. /\* Operator terner \*/

public class Oper4 {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

/\* KAMUS \*/

int i = 0; /\* perhatikan int i,j=0 bukan seperti ini \*/

int j = 0;

char c = 8; char d = 10;

int e = (((int)c > (int)d) ? c: d);

int k = ((i>j) ? i: j);

/\* ALGORITMA \*/

System.out.print ("Nilai e = "+ e);

System.out.print ("\nNilai k = "+ k);

i = 2;

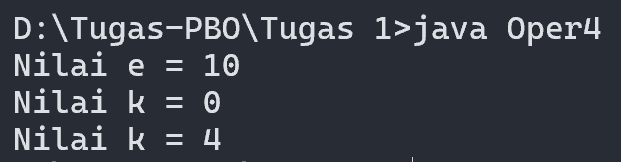
j = 3;

k = ((i++>j++) ? i: j) ;

System.out.print ("\nNilai k = "+ k);

}

}

Output :

Penjelasan :

Program di atas mendemonstrasikan penggunaan **operator ternary** di Java, yang berfungsi sebagai alternatif singkat untuk pernyataan **if-else** dalam menentukan nilai berdasarkan suatu kondisi. Pertama, program mendeklarasikan variabel i dan j dengan nilai awal 0, serta dua variabel karakter c = 8 dan d = 10. Variabel e diberikan nilai menggunakan ekspresi **ternary** (((int)c > (int)d) ? c : d), yang membandingkan c dan d. Jika c lebih besar dari d, maka e bernilai c, jika tidak, e bernilai d. Karena c (8) lebih kecil dari d (10), maka e diisi dengan 10.

Selanjutnya, variabel k juga diberi nilai menggunakan operator ternary ((i > j) ? i : j), yang memilih nilai i jika i lebih besar dari j, atau j jika tidak. Karena i dan j bernilai sama (0), kondisi i > j bernilai **false**, sehingga k diisi dengan nilai j, yaitu 0. Program kemudian mencetak **e = 10** dan **k = 0**.

Setelah itu, nilai i dan j diperbarui menjadi i = 2 dan j = 3, lalu operator ternary kembali digunakan dalam ekspresi k = ((i++ > j++) ? i : j). Di sini, i++ > j++ membandingkan nilai i (2) dengan j (3), yang hasilnya **false**. Karena kondisi ini **false**, k diberi nilai j. Namun, karena j++ digunakan, nilai j telah meningkat menjadi 4, sehingga program mencetak **k = 4**. Secara keseluruhan, program ini menunjukkan bagaimana **operator ternary** bekerja untuk memilih nilai berdasarkan suatu kondisi, serta efek dari **post-increment (i++, j++)** yang menyebabkan nilai variabel bertambah setelah perbandingan dilakukan.

1. /\* Contoh pengoperasian variabel bertype dasar \*/

public class Oprator {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

/\* Kamus \*/

boolean Bool1, Bool2, TF ; int i,j, hsl ;

float x,y,res;

/\* algoritma \*/

System.out.println ("Silahkan baca teksnya dan tambahkan perintah untuk menampilkan output");

Bool1 = true; Bool2 = false;

TF = Bool1 && Bool2 ; /\* Boolean AND \*/

System.out.println("Bool1 AND Bool2: " + TF);

TF = Bool1 || Bool2 ; /\* Boolean OR \*/

System.out.println("Bool1 OR Bool2: " + TF);

TF = ! Bool1 ; /\* NOT \*/

System.out.println("NOT Bool1: " + TF);

TF = Bool1 ^Bool2; /\* XOR \*/

System.out.println("Bool1 XOR Bool2: " + TF);

/\* operasi numerik \*/

i = 5; j = 2 ;

hsl = i+j;

System.out.println("i + j: " + hsl);

hsl = i - j;

System.out.println("i - j: " + hsl);

hsl = i \* j;

System.out.println("i \* j: " + hsl);

hsl = i / j;

System.out.println("i / j: " + hsl);

hsl = i%j ; /\* sisa. modulo \*/

System.out.println("i % j: " + hsl);

/\* operasi numerik \*/

x = 5 ; y = 5 ;

res = x + y;

System.out.println("x + y: " + res);

res = x - y;

System.out.println("x - y: " + res);

res = x / y;

System.out.println("x / y: " + res);

res = x \* y;

System.out.println("x \* y: " + res);

/\* operasi relasional numerik \*/

TF = (i==j);

System.out.println("i == j: " + TF);

TF = (i!=j);

System.out.println("i != j: " + TF);

TF = (i < j);

System.out.println("i < j: " + TF);

TF = (i > j);

System.out.println("i > j: " + TF);

TF = (i <= j);

System.out.println("i <= j: " + TF);

TF = (i >= j);

System.out.println("i >= j: " + TF);

/\* operasi relasional numerik \*/

TF = (x != y);

System.out.println("x != y: " + TF);

TF = (x < y);

System.out.println("x < y: " + TF);

TF = (x > y);

System.out.println("x > y: " + TF);

TF = (x <= y);

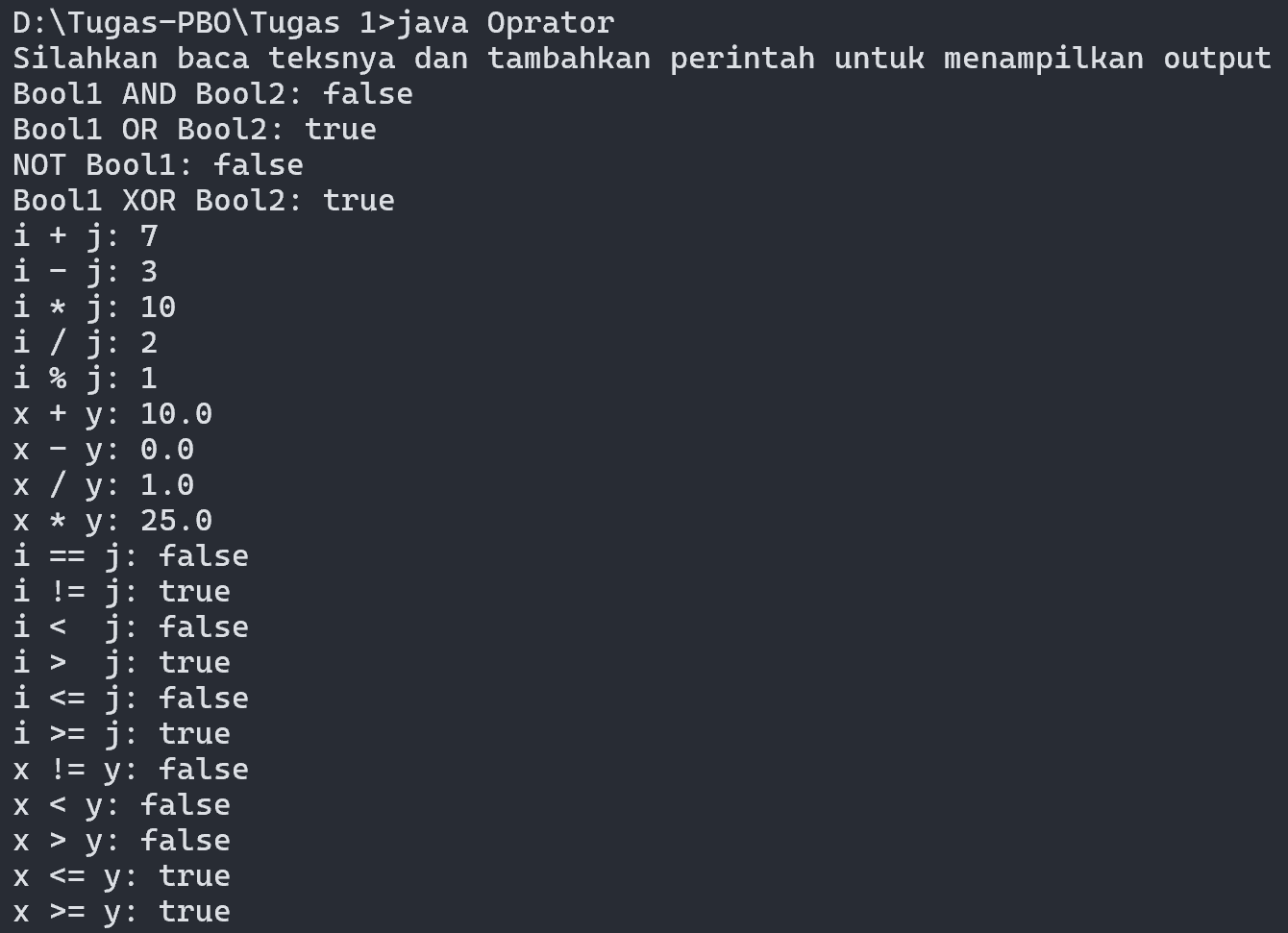
System.out.println("x <= y: " + TF);

TF = (x >= y);

System.out.println("x >= y: " + TF);

}

}

 Output :

Penjelasan :

Program di atas adalah contoh penggunaan berbagai **operator dasar** dalam Java, termasuk **operator logika, aritmetika, dan relasional**. Program ini mendeklarasikan beberapa variabel, baik yang bertipe **boolean** maupun **numerik (integer dan float)**, lalu mengoperasikannya menggunakan berbagai jenis operator.

Pertama, program melakukan **operasi logika** menggunakan dua variabel boolean, Bool1 yang bernilai **true** dan Bool2 yang bernilai **false**. Operator **AND (&&)** menghasilkan **false** karena salah satu operandnya false. Operator **OR (||)** menghasilkan **true** karena setidaknya satu operand bernilai true. Operator **NOT (!)** membalik nilai boolean, sehingga !Bool1 menjadi **false**. Operator **XOR (^)** menghasilkan **true** karena Bool1 dan Bool2 memiliki nilai berbeda.

Selanjutnya, program melakukan **operasi aritmetika** dengan dua variabel integer i = 5 dan j = 2. Operasi **penjumlahan (+)**, **pengurangan (-)**, **perkalian (\*)**, dan **pembagian (/)** menghasilkan nilai sesuai dengan aturan matematika. Perlu diperhatikan bahwa i / j menghasilkan **integer** karena kedua operand bertipe integer, sehingga hasilnya adalah **2**, bukan **2.5**. Selain itu, operator **modulo (%)** digunakan untuk mendapatkan sisa pembagian, di mana 5 % 2 menghasilkan **1**.

Kemudian, program juga melakukan **operasi aritmetika pada tipe float** menggunakan x = 5 dan y = 5. Karena tipe data float mendukung angka desimal, semua operasi seperti **penjumlahan, pengurangan, perkalian, dan pembagian** menghasilkan hasil yang lebih akurat dibandingkan operasi pada integer.

Terakhir, program melakukan **operasi relasional** untuk membandingkan nilai-nilai integer maupun float. Operator **==** digunakan untuk mengecek kesamaan nilai, sementara **!=** untuk mengecek ketidaksamaan. Operator **<, >, <=, dan >=** digunakan untuk membandingkan hubungan antara dua angka. Hasil dari operasi ini adalah boolean (true atau false). Misalnya, karena i = 5 dan j = 2, ekspresi i > j menghasilkan **true**, sedangkan i < j menghasilkan **false**. Untuk variabel float x dan y, karena nilainya sama (5), ekspresi x != y bernilai **false**, sedangkan x >= y bernilai **true**.

Secara keseluruhan, program ini menunjukkan bagaimana **operator logika, aritmetika, dan relasional** bekerja dalam Java, serta bagaimana mereka diterapkan pada berbagai tipe data.