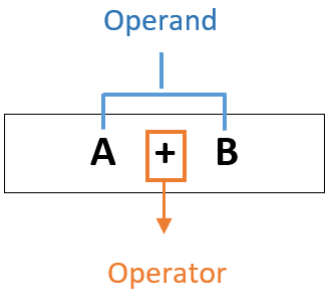
OPERATOR

OBJEKTIF :

1. Mahasiswa Mampu Memahami Operator pada Java.
2. Mahasiswa Mampu Menggunakan *Software* IntelliJ IDEA untuk Membuat Program Operator.

PENDAHULUAN

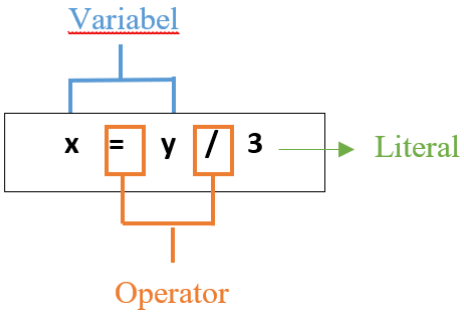
Operator adalah suatu simbol yang digunakan untuk memberikan instruksi kepada komputer untuk melakukan aksi terhadap satu atau lebih *operand*. Sedangkan *operand* adalah nilai asal yang dioperasikan oleh operator. *Operand* yang dioperasikan dapat berupa literal, variabel, atau nilai dikirim metode atau fungsi. Method adalah kumpulan pernyataan yang dikelompokkan bersama untuk melakukan operasi.



Berikut adalah tabel operator yang dipakai pada Java :

+	-	*	/	%	&	|
^	~	&&	||	!	<	>
<=	>=	<<	>>	>>>	=	?
++	--	==	+=	-=	*=	/=
%=	&=	|=	^=	!=	<<=	>>=
>>>=	.	[	]	(	)	

Operator yang diterapkan pada variabel dan literal akan membentuk suatu ekspresi. Literal adalah konstanta yang ditentukan oleh programmer. Contoh literal yaitu berupa angka. Ekspresi adalah barisan *operand-operand* dengan satu operator atau lebih yang menghasilkan suatu nilai. Ekspresi dapat dipandang sebagai suatu persamaan. Contoh ekspresi :



Ekspresi di atas adalah variabel *y* dibagi 3 menggunakan operasi pembagian '/', dan hasilnya disimpan di *x* menggunakan operator pemberian nilai '='.

Untuk menentukan operasi mana yang dilakukan terlebih dahulu daripada operasi lainnya, perhatikan tingkat level urutan pada operator. Operator yang mempunyai level lebih tinggi (angka terkecil pada tabel level operator) akan dioperasikan terlebih dahulu dibandingkan operator lain yang levelnya lebih rendah. Apabila operator memiliki level urutan yang sama, maka operasi dilakukan secara berurutan dari kiri ke kanan.

Operator	Level
()	1
!	2
*	3
/	3
%	3
+	4
-	4
<	5
<=	5
>=	5
>	5
=	6
!=	6
&&	7
||	8

Contoh :

Operasi apapun yang ada dalam tanda kurung '()' akan dieksekusi pertama kali oleh program, karena tanda kurung '()' memiliki level urutan lebih tinggi dibanding operator lain.

3.1 OPERATOR BILANGAN BULAT

Terdapat tiga tipe operasi yang dapat dilakukan pada *integer*, yaitu operator *unary*, operator *binary*, dan operator relasional.

3.1.1 OPERATOR BILANGAN BULAT UNARY

Operator *unary* adalah operator yang digunakan untuk melakukan operasi yang melibatkan 1 buah *operand*. Berikut ini adalah daftar operator bilangan bulat *unary* :

Deskripsi	Operator
Increment	++
Decrement	- -
Negasi	-
Bitwise complement	~

Operator *increment* (++) dan *decrement* (--) digunakan untuk menaikkan dan menurunkan variabel sebanyak 1 nilai. Operator-operator ini dapat digunakan dalam bentuk prefiks atau postfiks. Operator dalam bentuk prefiks melakukan perubahan nilai sebelum melakukan evaluasi suatu ekspresi. Sedangkan operator dalam bentuk postfiks melakukan perubahan nilai setelah ekspresi dievaluasi. Berikut adalah contoh penggunaan operator prefiks dan postfiks :

```
y = ++ x;
z = x--;
```

Pada contoh pertama, variabel x dioperasikan *prefix incremented* yang berarti nilai pada variabel dinaikkan sebelum diberikan ke y. Pada contoh kedua, variabel x dioperasikan *postfix incremented* yang berarti diturunkan setelah nilai diberikan ke z. (variabel z diberi nilai oleh variabel x sebelum variabel x diturunkan)

Keterangan :

- ++x adalah *prefix increment*, menambahkan 1 pada variabel x, dan kemudian menggunakannya dalam ekspresi.
- --x adalah *prefix decrement*, mengurangi 1 pada variabel x, dan kemudian menggunakannya dalam ekspresi.
- x++ adalah *postfix increment*, menggunakan nilai variabel x pada ekspresi, dan kemudian ditambahkan 1.
- x-- adalah *postfix decrement*, menggunakan nilai variabel x pada ekspresi, dan kemudian dikurangi 1.

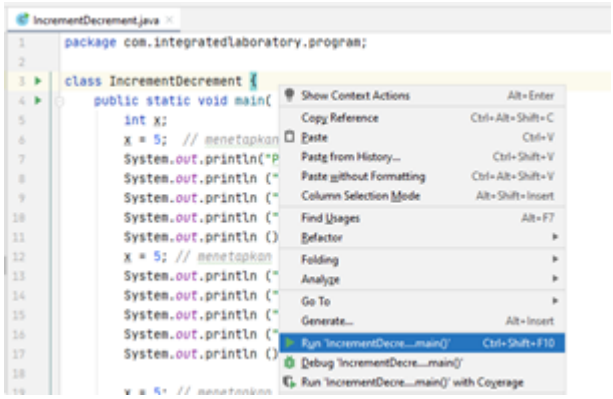
Berikut adalah contoh program penggunaan *increment & decrement* :

```
package com.integratedlaboratory.program;
class IncrementDecrement {
public static void main( String args[] ){
     int x;

  x = 5;  // menetapkan nilai 5 ke variabel x
      System.out.println("Penggunaan Postfiks Increment");
   System.out.println ("Nilai awal x adalah  " + x );
   System.out.println ("Operator x++ = " + x++);
   System.out.println ("Nilai  x saat ini = " + x);
   System.out.println ();//membuat baris kosong

x = 5; // menetapkan nilai 5 ke variabel x
      System.out.println ("Penggunaan Postfiks Decrement");
   System.out.println ("Nilai awal x adalah  " + x );
   System.out.println ("Operator x-- = " + x-- );
   System.out.println ("Nilai  x saat ini = " + x);
   System.out.println ();//membuat baris kosong
```

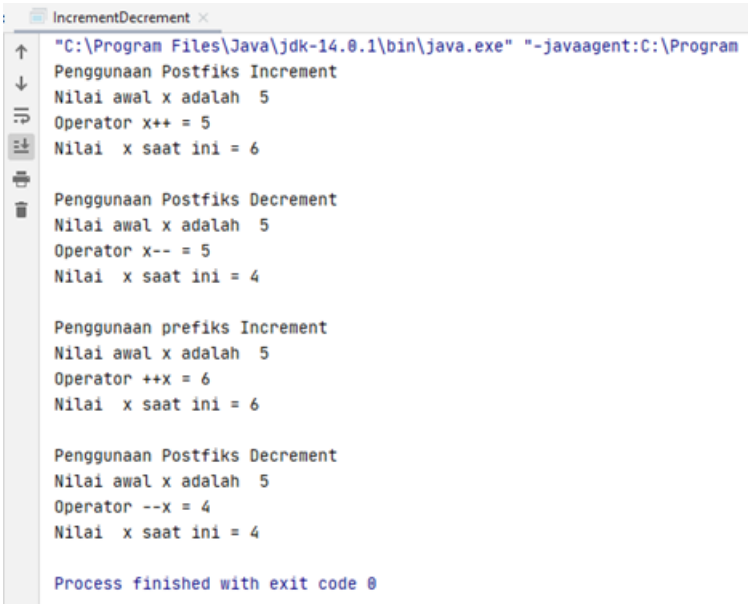
```
x = 5; // menetapkan nilai 5 ke variabel x
   System.out.println("Penggunaan prefiks Increment");
     System.out.println("Nilai awal x adalah  " + x );
   System.out.println("Operator ++x = " + ++x);
   System.out.println ("Nilai  x saat ini = " + x);
   System.out.println();//membuat baris kosong

 x = 5; // menetapkan nilai 5 ke variabel x
  System.out.println("Penggunaan Postfiks Decrement ");
  System.out.println("Nilai awal x adalah  " + x );
  System.out.println("Operator --x = " + --x);
  System.out.println ("Nilai  x saat ini = " + x);
        }
}
```

Perintah :

Tekan tombol Ctrl+Shift+F10 untuk melakukan *Run* pada IntelliJ IDEA atau dengan melakukan *klik* kanan pada *file* Java seperti berikut:



Output :

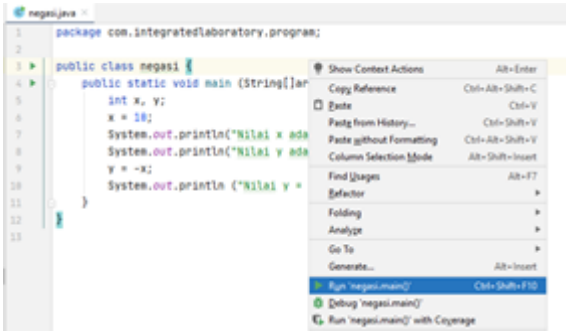


Operator negasi *unary* (-) digunakan untuk mengubah tanda nilai bilangan bulat. Perhatikan contoh berikut:

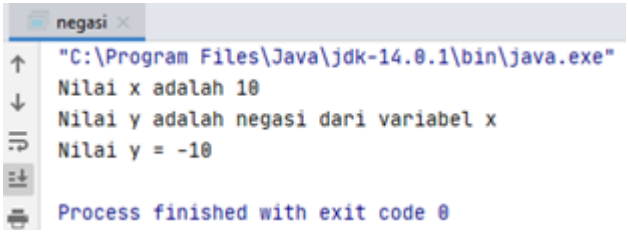
```
package com.integratedlaboratory.program;
public class negasi {
    public static void main (String[]args){
        int x, y;
        x = 10;
    System.out.println("Nilai x adalah 10");
    System.out.println("Nilai y adalah negasi dari variabel x");
    y = -x;
    System.out.println ("Nilai y = " + y);
    }
}
```

Perintah :

Tekan tombol Ctrl+Shift+F10 untuk melakukan *Run* pada IntelliJ IDEA atau dengan melakukan *klik* kanan pada *file* Java seperti berikut:



Output :



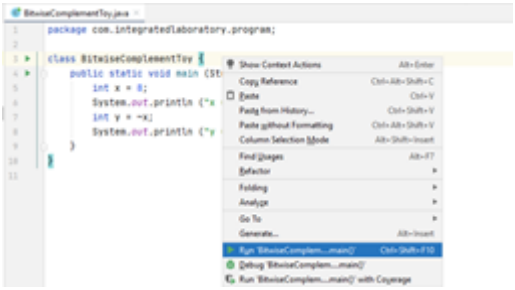
Pada contoh di atas, x diberi nilai 10 dan kemudian dinegasikan dan hasilnya disimpan ke variabel y. Maka hasilnya adalah y bernilai –10.

Bitwise complement operator (~) melakukan negasi *bitwise* nilai bilangan bulat. *Bitwise complement* berarti masing-masing bit di angka biner maka semua bilangan biner 0 menjadi 1 dan semua bilangan biner 1 menjadi 0. Berikut ini adalah contoh program penggunaan *bitwise complement* pada Java :

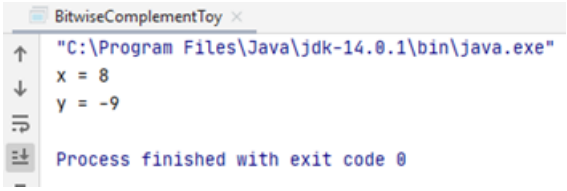
```
package com.integratedlaboratory.program;
class BitwiseComplementToy {
        public static void main (String args []) {
        int x = 8;
         System.out.println ("x = " + x);
         int y = ~x;
         System.out.println ("y = " + y);
      }
}
```

Perintah :

Tekan tombol Ctrl+Shift+F10 untuk melakukan *Run* pada IntelliJ IDEA atau dengan melakukan *klik* kanan pada *file* Java seperti berikut:



Output :



Keterangan :

- Variabel x diberi nilai 8, kemudian dioperasikan *bitwise complement* sebelum diberikan ke variabel y. Hal ini berarti semua bit di variabel x dinegasikan, hasilnya adalah bilangan bilangan bulat -9. Contoh di atas terjadi karena bilangan *int* menggunakan *two's complement.*
- Semua bilangan *int* (kecuali *char*) direpresentasikan *two's complement* yang berarti bilangan negatif direpresentasikan dengan inversi semua bit ditambah 1. Karena 8 = 00001000 maka inversi semua bit menghasilkan 11110111 adalah angka -9.
- *Two's complement* adalah suatu sistem penomoran yang diterapkan dalam beberapa jenis komputer untuk merepresentasikan nilai-nilai negatif, hanya saja dalam proses negasinya semua bit juga akan dibalik. Pada cara ini terdapat aturan bahwa nilai 0 (nol) akan direpresentasikan hanya dengan satu nilai 0 (nol).

3.1.2 OPERATOR BINER BILANGAN BULAT

Operator biner bilangan bulat adalah operator yang beroperasi pada 2 *operand* bilangan bulat. Berikut adalah tabel operator bilangan bulat biner :

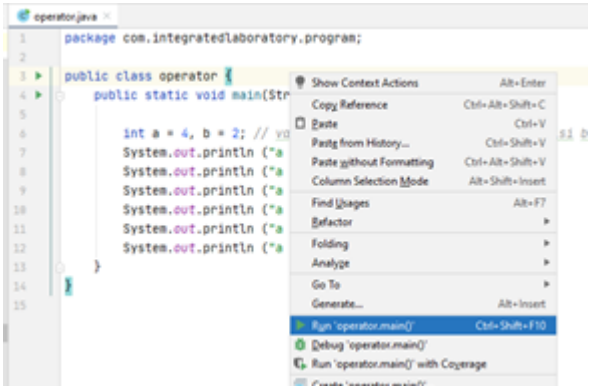
Deskripsi	Operator
Penambahan	+
Pengurangan	-
Perkalian	*
Pembagian	/
Sisa bagi	%
Bitwise AND	&
Bitwise OR	|
Bitwise XOR	^
Left-shift	<<
Right-shift	>>
Zero-fill-right-shift	>>>

Operator penjumlahan, pengurangan, perkalian dan pembagian (+, - , * dan /) melakukan operasi seperti pada aritmatika bilangan bulat. Berikut contoh program penggunaan operator :

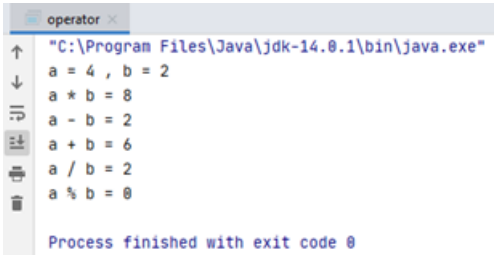
```
package com.integratedlaboratory.program;
public class operator {
public static void main(String[] args) {
    int a = 4, b = 2; // variabel a berisi bilangan 4 dan b berisi bilagan 2
        System.out.println ("a = 4 , b = 2 ");
        System.out.println ("a * b = "+(a*b));
        System.out.println ("a - b = "+(a-b));
        System.out.println ("a + b = "+(a+b));
        System.out.println ("a / b = "+(a/b));
        System.out.println ("a % b = "+(a%b));
           }
 }
```

Perintah :

Tekan tombol Ctrl+Shift+F10 untuk melakukan *Run* pada IntelliJ IDEA atau dengan melakukan *klik* kanan pada *file* Java seperti berikut:



Output :



3.1.3 OPERATOR BILANGAN BULAT RELASIONAL

Operator relasional adalah operator yang digunakan untuk membandingkan 2 buah nilai bilangan bulat. Hasil dari operator perbandingan ini adalah *boolean true* atau *false.* Berikut adalah daftar operator bilangan bulat relasional :

Deskripsi	**Operator**
Kurang dari	<
Lebih besar dari	>
Lebih kecil atau sama dengan	<=
Lebih besar dari atau sama dengan	>=
Sama dengan	==
Tidak sama dengan	!=

3.2 OPERATOR BILANGAN FLOATING-POINT

Terdapat tiga tipe operasi yang dapat dilakukan pada *floating-point*, yaitu operator *unary floating-point,* operator *binary floating-point, dan* operator relasional *floating-point*.

3.2.1 OPERATOR BILANGAN FLOATING-POINT UNARY

Operator-operator bilangan *floating-point unary* beroperasi pada satu bilangan *floating-point.* Berikut ini adalah daftar operator bilangan bulat *unary*:

Deskripsi	Operator
Increment	++
Decrement	--

Hanya terdapat dua operator *unary floating-point* yaitu *increment* (menaikkan bilangan dengan 1.0) dan *decrement* (menurunkan bilangan dengan 1.0).

3.2.2 OPERATOR BINER FLOATING-POINT

Operator biner *floating-point* adalah operator yang beroperasi pada 2 operand bilangan *floating-point*. Berikut adalah tabel operator bilangan biner *floating-point:*

Deskripsi	Operator
Penambahan	+
Pengurangan	-
Perkalian	*
Pembagian	/
Sisa bagi	%

3.2.3 OPERATOR RELASIONAL FLOATING-POINT

Operator relasional adalah operator yang digunakan untuk membandingkan 2 buah nilai bilangan *floating-point*. Hasil dari operator perbandingan ini adalah boolean *true* atau *false.*

Deskripsi	Operator
Kurang dari	<
Lebih besar dari	>
Lebih kecil atau sama dengan	<=
Lebih besar dari atau sama dengan	>=
Tidak sama dengan	!=

Pada operator ini, seharusnya tidak pernah menggunakan operator == terhadap bilangan *floating-point.* Karena operator == terdapat operasi pembulatan pada bilangan. Sedangkan pada bilangan *floating-point* mencegah terjadinya kesamaan yang sama persis di tingkat bit (dalam arti kesamaan seluruh bit secara eksak).

3.3 OPERATOR BOOLEAN

Operator *boolean* beroperasi pada tipe *boolean* dan menghasilkan tipe *boolean*. Daftar operator *boolean* sebagai berikut:

Deskripsi	Operator
Evaluation AND	*&*
Evaluation OR	|
Evaluation XOR	^
Logical AND	*&&*
Logical OR	||
Negation	*!*
Conditional	*?:*
Equal-to	==
Not-equal to	*!=*

3.3.1 OPERATOR EVALUASI DAN OPERATOR LOGIKA

Operator evaluasi (&, |, dan ^) melakukan evaluasi di kedua sisi ekspresi sebelum menentukan hasil. Operator logika (&& dan ||) mengabaikan evalusi sisi kanan ekspresi jika hasil telah dapat ditentukan dari hasil di sisi kiri. Modus demikian disebut *short-circuit evaluation* atau modus logis. Operator evaluasi (&, | , dan ^) melakukan evaluasi di kedua sisi ekspresi sebelum menentukan hasil. Bentuk umum operator evaluasi boolean :

```
boolean nama_variabel = pernyataan1 & pernyataan2;
boolean nama_variabel = pernyataan1 | pernyataan2;
```

Operator logika (&& dan ||) mengabaikan pernyataan2 pada ekspresi jika hasil telah dapat ditentukan dari hasil pada pernyataan1. Bentuk umum operator logika *boolean* :

```
boolean nama_variabel = pernyataan1 && pernyataan2;
boolean nama_variabel = pernyataan1 || pernyataan2
```

Perhatikan contoh berikut:

```
boolean result = isValid & (Count > 10) ;
boolean result = isValid && (Count > 10) ;
```

Pada ekspresi pertama digunakan operator evaluasi *AND* (&) untuk membuat penugasan *(assignment).* Pada kasus ini kedua sisi ekspresi selalu dievaluasi, tidak peduli hasil variabel-variabel yang terlibat. Modus demikian disebut *long-circuit evaluation* atau modus evaluasi. Pada contoh kedua, operator logis *AND* (&&) digunakan. Saat nilai *boolean* isValid lebih dulu diperiksa. Jika bernilai *false*, maka sisi kanan ekspresi diabaikan. Operator ini lebih efisien karena nilai *false* di sisi kiri ekspresi telah memadai untuk memberikan informasi untuk menentukan hasil

seluruhnya. Meskipun operator logis lebih efisien dibanding operator evaluasi, namun sering terdapat kebutuhan untuk memastikan seluruh ekspresi dievaluasi menggunakan operator evaluasi.

Bitwise XOR (exclusive OR) "^" adalah operator di Java yang memberikan jawaban '1' jika kedua bit dalam operannya berbeda, jika kedua bitnya sama maka operator *XOR* memberikan hasil '0' . *XOR* adalah operator biner yang dievaluasi dari kiri ke kanan. Operator "^" tidak ditentukan untuk argumen tipe *String*. Bentuk umum operator *Bitwise XOR :*

```
boolean nama_variabel1 = nilai_boolean;
boolean nama_variabel2 = nilai_boolean;
boolean nama_variabel3 = nama_variabel1 ^ nama_variabel2;
```

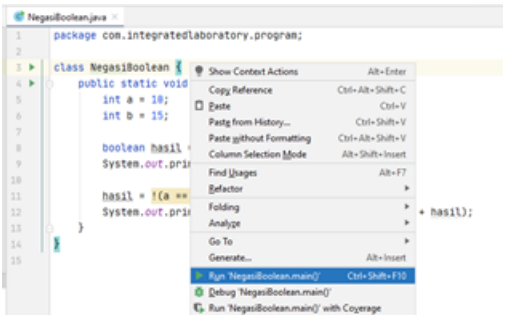
3.3.2 OPERATOR BOOLEAN NEGATION

Operator ini akan membalikan nilai *boolean true* menjadi *false* dan begitu juga sebaliknya. Berikut contoh program negasi boolean :

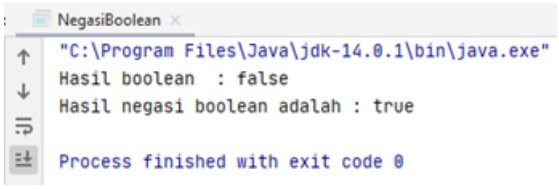
```
package com.integratedlaboratory.program;
class NegasiBoolean {
public static void main(String[] args) {
    int a = 10;
    int b = 15;

    boolean hasil = (a == b);
    System.out.println("Hasil boolean  : " + hasil);

    hasil = !(a == b);
    System.out.println("Hasil negasi boolean adalah : " + hasil);
  }
}
```

Perintah :

Tekan tombol Ctrl+Shift+F10 untuk melakukan *Run* pada IntelliJ IDEA atau dengan melakukan *klik* kanan pada *file* Java seperti berikut:



Output :



3.3.3 OPERATOR BOOLEAN EQUAL-TO DAN NOT-EQUAL-TO

Operator *Equal-to* (==) hanya akan mengembalikan nilai *true* jika dua referensi memiliki nilai (*value*) yang sama. Bentuk umum operator *equal-to* :

```
boolean nama_objek;
nama_objek = (variabel_referensi1 == variabel_referensi2);
```

Operator *Not-Equal-to* (!=) hanya akan mengembalikan nilai *true* jika dua referensi memiliki nilai (*value*) yang berbeda. Bentuk umum operator *not-equal-to:*

```
boolean nama_objek;
nama_objek = (variabel_referensi1 != variabel_referensi2);
```

3.3.4 OPERATOR BOOLEAN KONDISIONAL

Operator *boolean* kondisional (?:) adalah operator *boolean* paling unik. Operator ini disebut *ternary* operator karena memerlukan tiga buah kondisi dan dua ekspresi. Sintaks penggunaan operator *boolean* kondisional :

```
Conditional  ? Expression1 : Expression2
```

Condition merupakan *boolean* yang lebih dulu dievaluasi untuk menentukan apakah bernilai *true* atau *false*. Jika *Condition* menghasilkan nilai *true*, maka Expression1 dievaluasi. Jika *Condition* menghasilkan nilai *false*, maka Expression2 dievaluasi.

3.4 OPERATOR STRING

Seperti bilangan bulat, bilangan *floating-point,* dan *boolean*, *string* dapat dimanipulasi dengan operator. Hanya terdapat satu operator terhadap *string*, yaitu operator penyambungan (*concatenation*) yang dilambangkan simbol tambah '+'.

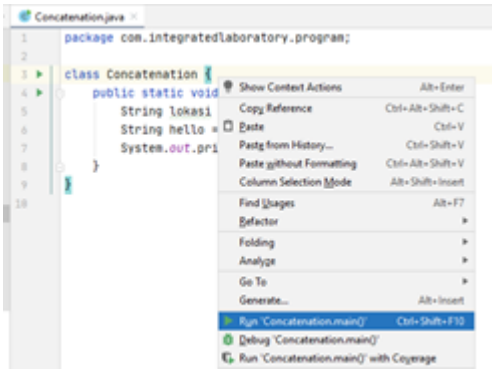
3.4.1 PENYAMBUNGAN STRING

Java menggunakan tanda tambah '+' untuk menyambungkan dua *string*. Apabila saat menyambungkan *string* dengan nilai yang bukan bertipe *string* maka nilai tersebut dikonversi terlebih dahulu menjadi *string*. Setiap objek Java dapat dikonversi menjadi *string*. Berikut adalah contoh menyambungkan string :

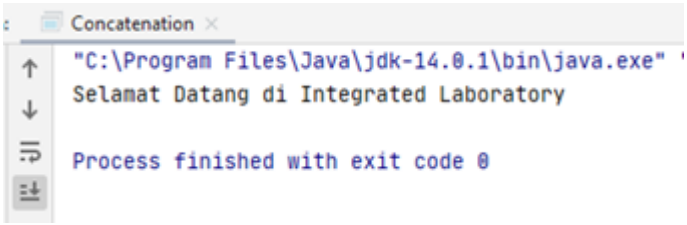
```
class Concatenation {
  public static void main(String[] args) {
  String lokasi = "Integrated Laboratory";


 String hello = "Selamat Datang di " + lokasi;
 System.out.println (hello);
   }
}
```

Perintah :

Tekan tombol Ctrl+Shift+F10 untuk melakukan *Run* pada IntelliJ IDEA atau dengan melakukan *klik* kanan pada *file* Java seperti berikut:



Output :



3.4.2 SUBSTRING

Substring digunakan untuk mengambil isi sebagian dari variabel *String* atau mengambil potongan beberapa karakter dari sebuah isi *String*. Dimana, pada parameter pertama merupakan *index* awal dari *String* dan parameter kedua merupakan batasan *index* yang akan diambil. Sintaks penggunaan *substring* :

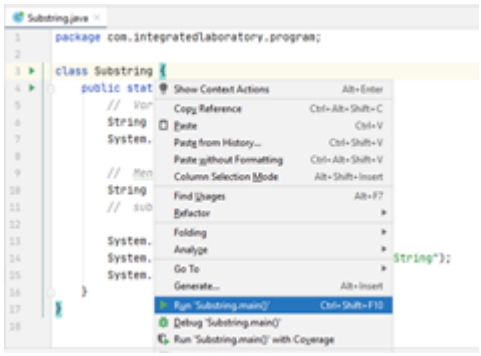
```
nama_variabel.substring(index_awal,index_akhir);
```

Contoh :

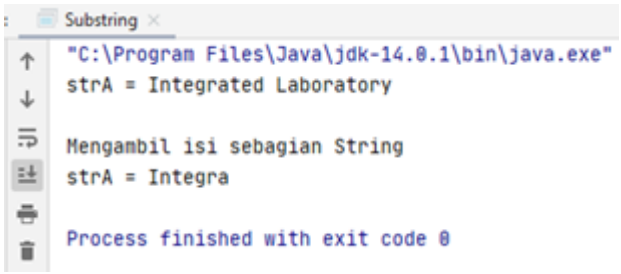
```
class Substring {

   public static void main(String[] args) {
   //  Variable String
   String strA = "Integrated Laboratory";
      //  Mengambil isi sebagian String
   System.out.println("strA = " + strA);

 String sub_strA = strA.substring(0,7);
 // substring(index_awal, index_akhir)
  System.out.println ();
  System.out.println ("Mengambil isi sebagian String");
  System.out.println ("strA = "+sub_strA);

  }
}
```

Perintah :

Tekan tombol Ctrl+Shift+F10 untuk melakukan *Run* pada IntelliJ IDEA atau dengan melakukan *klik* kanan pada *file* Java seperti berikut:



Output :



3.4.3 PENGUJIAN KESAMAAN

Pada Java, didalam *library String* terdapat sebuah *method* bernama equals(). M*ethod* tersebut digunakan untuk membandingkan 2 buah variabel yang bertipe data *String*. Jika nilai dari kedua variabel tersebut mempunyai karakter yang sama (baik dari segi huruf besar, huruf kecil, atau spasi), maka akan menghasilkan nilai *boolean true* dan jika tidak sesuai maka akan menghasilkan nilai *false*.

Jika dalam pengujian kesamaan, *method* yang digunakan saat ingin menguji tanpa memperhatikan huruf kapital atau huruf kecil yaitu *method* equalsIgnoreCase().

3.5 OPERATOR PENUGASAN

Operator penugasan adalah operator yang digunakan untuk memanipulasi dan menginisialisasi nilai pada sebuah variabel, operator tersebut identik dengan simbol '=' dan dapat digabungkan dengan simbol operator aritmatika, contohnya seperti '+=', '-=', '%=', '*=', '/='. Operator ini merupakan operator untuk mempersingkat dalam pengodeannya. Dimana operator pemberian nilai ini hampir sama dengan operator aritmatika hanya saja bedanya terletak pada pengodeannya. Dalam masalah penambahan, pengurangan dan lainnya tetap sama akan tetapi menggunakan dua variabel yang berbeda. Operator penugasan pada dasarnya bekerja dengan semua tipe data dasar. Tabel berikut adalah daftar operator penugasan :

Deskripsi	Operator	Contoh	Keterangan
Sederhana	=	A = 5	A = 5
Penambahan	+=	A += B	A = A + B
Pengurangan	-=	A -= B	A = A - B
Perkalian	*=	A *= B	A = A * B
Pembagian	/=	A /= B	A = A / B
Sisa bagi	%=	A %= B	A = A % B
AND	&=	A &= B	A = A & B
OR	|=	A |= B	A = A | B
XOR	^=	A ^= B	A = A ^ B

REFERENSI

[1] Hariyanto, Bambang. 2010. Esensi-Esensi Bahasa Pemrograman Java Revisi Ketiga. Bandung: Informatika Bandung.

[2] Athoillah, Wildan M. 10 Mei 2017. Apa itu Operator Aritmatika, Relasi, Increment & Decrement pada Java. Diambil dari : https://www.wildantechnoart.net/2017/05/tutorial-4-java-operetor-aritmatika-relasi-increment-decrement.html . (18 Juli 2020)

[3] Athoillah, Wildan M. 15 November 2017. Belajar Menggunakan Operator Assignment pada Java. Diambil dari : https://www.wildantechnoart.net/2017/11/belajar-menggunakan-operator-assignment-pada-java.html . (21 Juli 2020)

[4] Muhardian, Ahmad. 9 November 2015. Belajar Java: 6 Jenis Operator yang Harus Dipahami. Diambil dari : https://www.petanikode.com/java-operator/ . (17 Juli 2020)