



## BAB V OPERATORS

### 5.1. Operators

Operator adalah suatu simbol yang digunakan untuk memberikan perintah kepada komputer untuk melakukan aksi terhadap satu atau lebih operand. Operand adalah sesuatu yang dioperasikan oleh operator (literal, variabel, atau *return value* dari fungsi). Contoh operasi pada operator dapat dilihat pada Gambar di bawah ini.



Dalam Java, terdapat beberapa operator, seperti operator Aritmatika, Pembandingan, Logika, Penugasan, dan operator Bit, serta operator ternari yang dapat digunakan untuk menghasilkan sebuah nilai. Simbol-simbol operator di Java seperti terlihat pada Tabel di bawah ini.

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

Pada Tabel di atas merupakan kumpulan dari seluruh operator dalam Java, operator-operator tersebut dikelompokkan menjadi beberapa jenis sebagai berikut:





### 5.1.1. Operator Aritmatika

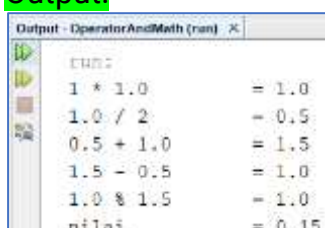
Operator aritmatika merupakan jenis operator yang digunakan untuk melakukan perhitungan aritmatik seperti tambah, bagi, kali, dan pengurangan.

Operator	Makna	Contoh
-	Unary minus	-1
+	Unary plus	+1
*	Perkalian	$2 * 3 \rightarrow 6$ $2.0 * 3 \rightarrow 6.0$
/	Pembagian	$7 / 2.0 \rightarrow 3.5$
%	Sisa pembagian	$7 \% 2 \rightarrow 1$ $8 \% 2 \rightarrow 0$
+	Penjumlahan (juga digunakan dalam penyambungan String)	$2 + 3 \rightarrow 5$
-	Pengurangan	$3 - 2 \rightarrow 1$

Contoh Program:

```
public class Aritmatika {  
  
    public static void main(String[] args) {  
        double up = +1; //mengindikasikan nilai positif  
        double um = -0.1; //mengindikasikan nilai negatif, meniadakan ekspresi  
        double n1 = 1 * up;  
        double n2 = n1 / 2;  
        double n3 = n2 + n1;  
        double n4 = n3 - n2;  
        double n5 = n4 % n3;  
        double nilai = n1 - n2 * n3 / n4 % n5 + um;  
  
        System.out.println("1 * " + up + "\t\t= " + n1);  
        System.out.println(n1 + " / 2\t\t= " + n2);  
        System.out.println(n2 + " + " + n1 + "\t= " + n3);  
        System.out.println(n3 + " - " + n2 + "\t= " + n4);  
        System.out.println(n4 + " % " + n3 + "\t= " + n5);  
        System.out.println("nilai\t\t= " + nilai);  
    }  
}
```

Output:



1 * 1.0	= 1.0
1.0 / 2	= 0.5
0.5 + 1.0	= 1.5
1.5 - 0.5	= 1.0
1.0 % 1.5	= 1.0
nilai	= 0.15





### 5.1.2. Operator Pembanding

Operator pembanding merupakan jenis operator yang digunakan untuk melakukan perbandingan terhadap dua buah nilai. Hasil perbandingan berupa 1 (benar/true) atau 0 (salah/false). Simbol dan fungsi operator pembanding dapat dilihat pada Tabel di bawah ini.

Operator	Keterangan
>	Lebih besar daripada
>=	Lebih besar atau sama dengan
<	Kurang dari
<=	Kurang dari atau sama dengan
!=	Tidak sama dengan
==	Sama dengan

#### Contoh Perbandingan:

1 > 9	→ Salah
1 >= 9	→ Salah
1 < 9	→ Benar
1 <= 9	→ Benar
1 != 9	→ Benar
1 == 9	→ Salah

#### Kode Program

```
/**
 *
 * @author nishom
 */
public class Pembanding {
    public static void main(String[] args) {
        boolean b1 = 1 > 9;
        boolean b2 = 1 >= 9;
        boolean b3 = 1 < 9;
        boolean b4 = 1 <= 9;
        boolean b5 = 1 != 9;
        boolean b6 = 1 == 9;
        System.out.println("b1: "+b1);
        System.out.println("b2: "+b2);
        System.out.println("b3: "+b3);
        System.out.println("b4: "+b4);
        System.out.println("b5: "+b5);
        System.out.println("b6: "+b6);
    }
}
```

Output - OperatorAndMath (run) ×

```
run:
b1: false
b2: false
b3: true
b4: true
b5: true
b6: false
```





### 5.1.3. Operator Logika

Operator logika digunakan untuk membentuk suatu ekspresi perbandingan dari satu atau dua buah ekspresi perbandingan. Jenis-jenis operator logika adalah sebagai berikut:

Operator	Keterangan
&&	Operator "dan". Apabila operand yang berada di kiri && bernilai salah maka operand di sebelah kanan && tidak diproses
&	Operator "dan". Apabila operand yang berada di kiri & bernilai salah, maka operand di kanan & tetap diproses
	Operator "or". Apabila operand yang berada di kiri    bernilai benar maka operand di sebelah kanan    tidak diproses
	Operator "or". Apabila operand yang berada di kiri   bernilai benar, maka operand di kanan   tetap diproses
!	Operator "not"
?:	Operator ternary (berkondisi): Mempunyai efek seperti if-else.

#### Contoh Kode Program:

```
/**
 *
 * @author nishom
 */
public class Logika {

    public static void main(String[] args) {
        boolean b1 = true && 1 > 3;
        boolean b2 = 1 < 9 & false;
        boolean b3 = true || 1 > 3;
        boolean b4 = 1 < 9 & false;
        boolean b5 = !b4;
        String b6 = 1 > 3 ? "OK": "default";
        System.out.println("b1 :"+b1);
        System.out.println("b2 :"+b2);
        System.out.println("b3 :"+b3);
        System.out.println("b4 :"+b4);
        System.out.println("b5 :"+b5);
        System.out.println("b6 :"+b6);
    }
}
```

Output - OperatorAndMath (run)

```
run:
b1 :false
b2 :false
b3 :true
b4 :false
b5 :true
b6 :default
BUILD SUCCESSFUL
```



#### 5.1.4. Operator Penugasan

Operator penugasan digunakan untuk memberikan nilai ke suatu variabel. Simbol operator penugasan dapat dilihat pada tabel berikut:

Operator	Keterangan
=	<b>Pemberian nilai</b>
+=	Penambahan bilangan
-=	<b>Pengurangan bilangan</b>
*=	Pengalian bilangan
/=	Pembagian bilangan
%=	Untuk memperoleh sisa bagi

#### Contoh Kode Program:

```
/**
 *
 * @author nishom
 */
public class Penugasan {

    public static void main(String[] args) {
        double n = 1;
        System.out.println("n1: "+n);
        n += 3;
        System.out.println("n2: "+n);
        n -= 2;
        System.out.println("n3: "+n);
        n *= 0.75;
        System.out.println("n4: "+n);
        n /= 4;
        System.out.println("n5: "+n);
        n %= 0.25;
        System.out.println("n6: "+n);
    }
}
```

Output - OperatorAndMath (run)

```
run:
n1: 1.0
n2: 4.0
n3: 2.0
n4: 1.5
n5: 0.375
n6: 0.125
BUILD SUCCESSFUL
```







### 5.1.5. Operator Bit

Java mendukung 7 buah operator yang beroperasi pada level bit (0 dan 1), yaitu:

Operator	Keterangan
&	Dan (untuk biner)
	Atau (untuk biner)
^	Atau (eksklusif)
~	Bukan (untuk biner)
<<	Geser bit ke kiri
>>	Geser bit kanan
>>>	Geser kanan tak bertanda

Sebelum membahas lebih lanjut tentang fungsi dan definisi dari setiap symbol operator bit pada table di atas, sebaiknya Anda memahami dulu tentang bilangan biner dan bilangan decimal berikut:

Biner	1	1	1	1	1	1	1	1	11111111
Desimal	128	64	32	16	8	4	2	1	255
Pangkat	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	$2^{1-7}$

- Operator & berguna untuk melakukan operasi “dan” pada tataran bit. Seperti pada Table berikut:

Bit 1	Bit 2	Hasil
0	0	0
0	1	0
1	0	0
1	1	1

**Catatan:** hasil berupa 1 jika kedua bit yang dikenai operator & bernilai 1





**Contoh:**

**9 & 10 → 8**

```
1 0 0 1 → 9
1 0 1 0 → 10
    &
1 0 0 0 → 8
```

**29 & 7 → 5**

```
0 0 0 1 1 1 0 1 → 29
0 0 0 0 0 1 1 1 → 7
    &
0 0 0 0 0 1 0 1 → 5
```

- Operator **&** berguna untuk melakukan operasi “atau” pada tataran biner. Seperti pada Tabel berikut:

Bit 1	Bit 2	Hasil
0	0	0
0	1	1
1	0	1
1	1	1

**Catatan:** hasil berupa 0 jika kedua bit yang dikenai operator **&** bernilai 0

**Contoh:**

9 | 10 → 11

29 | 7 → 31

```
1 0 0 1 → 9
1 0 1 0 → 10
    |
1 0 1 1 → 11
```

```
0 0 0 1 1 1 0 1 → 29
0 0 0 0 0 1 1 1 → 7
    |
0 0 0 1 1 1 1 1 → 31
```

- Operator **^** berguna untuk melakukan operasi “atau eksklusif” pada tataran bit. Seperti pada Tabel berikut:



Bit 1	Bit 2	Hasil
0	0	0
0	1	1
1	0	1
1	1	0

**Catatan:** hasil berupa 1 jika salah satu bit yang dikenai operator ^ bernilai 1

**Contoh:**

$$9 \wedge 10 \rightarrow 3$$

$$29 \wedge 7 \rightarrow 26$$

$$\begin{array}{l} 1001 \rightarrow 9 \\ 1010 \rightarrow 10 \\ \wedge \\ 0011 \rightarrow 3 \end{array}$$

$$\begin{array}{l} 00011101 \rightarrow 29 \\ 00000111 \rightarrow 7 \\ \wedge \\ 00011010 \rightarrow 26 \end{array}$$

- Operator ~ memberikan hasil dengan masing-masing bit berupa kebalikan dari bit operand. Contoh:

$$\sim 185 \rightarrow -186$$

$$\begin{array}{l} 000..010111001 \rightarrow 185 \\ 111..101000110 \rightarrow \sim 185 = -186 \end{array}$$

- Operator << (geser kiri) berfungsi untuk menggeser bit ke kiri. Jumlah pergeseran ditentukan oleh operand yang terletak di kanan operator.

**Contoh:**

$$29 \ll 1 \rightarrow 58$$

$$29 \ll 2 \rightarrow 116$$

$$\begin{array}{l} 29 \rightarrow 00011101 \\ 58 \leftarrow 00111010 \end{array}$$







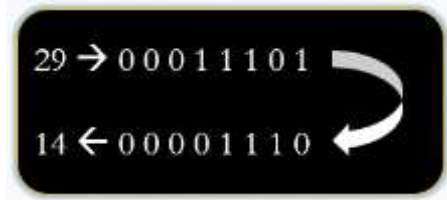
**Catatan:** pada dasarnya, pergeseran sebuah bit dengan operator << identik dengan perkalian dengan bilangan 2. secara umum,  $x \ll n \rightarrow x * 2^n$

- Operator >> (geser kanan) berfungsi untuk menggeser bit ke kanan. Jumlah pergeseran ditentukan oleh operand yang terletak di kanan operator.

**Contoh:**

29 >> 1 → 14

29 >> 2 → 7



- Operator >>> (geser kanan) berfungsi seperti operator >>, tetapi bit terkanan diisi dengan nol. Akibatnya, jika terdapat bilangan negatif digeser ke kanan dengan operator ini, maka hasilnya selalu positif. Contoh Operator dapat dilihat pada kode program di bawah ini

```
public class OperatorBit {  
    public static void main(String[] args) {  
        System.out.println("9 & 10 = " + (9 & 10));  
        System.out.println("29 & 7 = " + (29 & 7));  
        System.out.println("9 | 10 = " + (9 | 10));  
        System.out.println("29 | 7 = " + (29 | 7));  
        System.out.println("~185 = " + (~185));  
        System.out.println("-29 & 7 = " + (-29 & 7));  
        System.out.println("29 << 1 = " + (29 << 1));  
        System.out.println("29 >> 1 = " + (29 >> 1));  
        System.out.println("-29 >> 1 = " + (-29 >> 1));  
        System.out.println("29 >>> 1 = " + (29 >>> 1));  
    }  
}
```

Masing masing operator dalam satu ekspresi memiliki prioritas pengerjaan yang berbeda-beda. Itulah sebabnya jika terdapat suatu ekspresi yang melibatkan sejumlah operator, maka pengerjaannya ditentukan oleh prioritas masing-masing. Misalnya:



**Contoh:**

$$1 + 4 * 2$$



### Prioritas Operator

Operator	Keterangan
. [] ()	Prioritas Tinggi
++, -- ! ~ instanceof	
new (type) ekspresi	
- (negatif), + (plus)	
* / %	
+ -	
<< >> >>>	
-e, -r	
<<= >>=	
== !=	
&	
^	
&&	Prioritas Rendah
?:	
= += -= *= /= %= ^=	
&=  = <<= >>= >>>=	





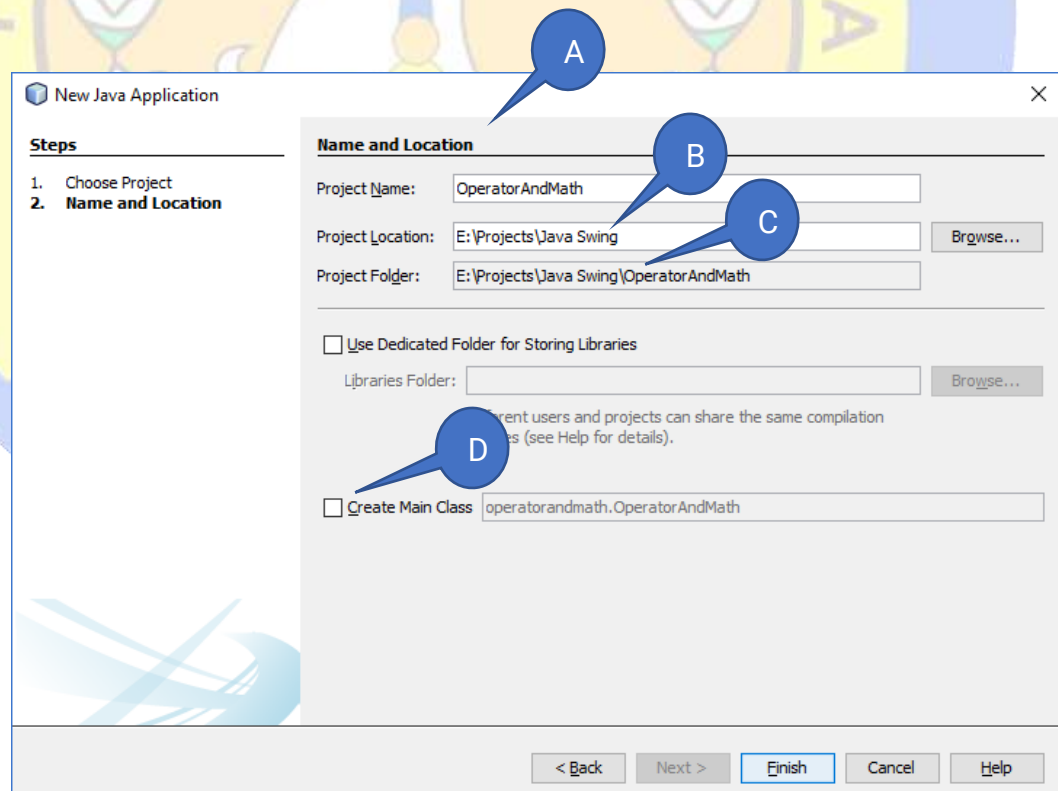
## 5.2. Practice

Praktikum pada bab ini adalah membuat project sederhana berbasis GUI dengan mengimplementasikan operator Aritmatika, pembandingan, logika, penugasan, bit, serta operator ternary.

### 5.2.1. Membuat *Project* Baru

Untuk membuat *project* baru di Netbeans ikuti langkah-langkah berikut:

1. Klik menu **File** → **New Project**, selanjutnya akan terbuka dialog **New Project**
2. Pada dialog **New Project**, untuk *Categories* pilih **Java**, dan untuk jenis **Projects** pilih **Java Application** → klik **Next** untuk melanjutkan ke tahapan berikutnya.
3. Pada dialog **New Java Application**, isi field dan pilihan yang ada sebagai berikut:
  - A. Project Name : **OperatorAndMath**
  - B. Project Location: Lokasi opsional, sesuai dengan keinginan anda
  - C. Project Folder : <<Default>>
  - D. *Create Main Class*: **Hilangkan tanda centang**



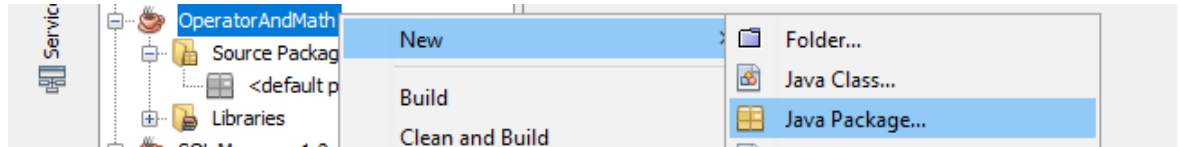
4. Klik **Finish**



### 5.2.2. Membuat *Package* Baru

Untuk membuat package baru ikuti langkah-langkah berikut:

1. Klik kanan pada “**Source Package**” → kemudian pilih **New** → **Java Package**. Selanjutnya akan terbuka dialog “New Java Package”.



Gambar 5.5 Membuat *Package* Baru

2. Pada dialog “New Java Package” isikan pada “Package Name” = **operator**
3. Klik **Finish**

### 5.2.3. Membuat Kelas (*class*) Baru

Buat kelas baru dengan mengikuti langkah-langkah berikut:

1. Klik kanan pada package **operator** → pilih **New** → pilih **Java Class**.
2. Lakukan konfigurasi atau pengisian sebagai berikut:
  - A. Class Name : **Konstanta**
  - B. Project : <<Default>>
  - C. Location : <<Default>>
  - D. Package : <<Default>>
  - E. Created File : <<Default>>
3. Klik **Finish**

**\*) Catatan:** Buat 1 (satu) kelas lagi dengan nama **Methods**

### 5.2.4. Mengubah Kode Program pada Kelas Konstanta dan Methods

Pertama, ubah kode program pada kelas “**Konstanta**” menjadi seperti kode di bawah ini:

```
public class Konstanta {  
    //operator aritmatika  
    public static final int TAMBAH = 1;  
    public static final int KURANG = 2;  
    public static final int BAGI = 3;  
    public static final int KALI = 4;  
    //undefined  
    public static final int UNDEFINED = 0;  
}
```



Selanjutnya ubah kode program pada kelas “**Methods**” menjadi seperti berikut:

```
1  package operator;
2
3  /**
4   *
5   * @author nishom
6   */
7  public class Methods {
8
9      public static double aritmatika_pemula(int pilihan, double operand1,
10         double operand2){
11          switch (pilihan) {
12              case Konstanta.TAMBAH:
13                  return operand1 + operand2;
14              case Konstanta.KURANG:
15                  return operand1 - operand2;
16              case Konstanta.BAGI:
17                  return operand1 / operand2;
18              case Konstanta.KALI:
19                  return operand1 * operand2;
20              default:
21                  break;
22          }
23          return 0;
24      }
25
26      public static double aritmatika_medium(int pilihan, double... operand){
27          double hasil = operand[0];
28          for (int i = 1; i < operand.length; i++) {
29              double opr = operand[i];
30              switch (pilihan) {
31                  case Konstanta.TAMBAH:
32                      hasil += opr;
33                      break;
34                  case Konstanta.KALI:
35                      hasil *= opr;
36                      break;
37                  case Konstanta.BAGI:
38                      hasil /= opr;
39                      break;
40                  case Konstanta.KURANG:
41                      hasil -= opr;
42                      break;
43                  default:
44                      break;
45              }
46          }
47          return hasil;
48      }
49  }
```







### 5.2.5. Mendesain Form Input

Setelah kita membuat method yang kita butuhkan, selanjutnya buat sebuah *Form* (seperti pada Gambar 5.3) untuk menerima data masukan dengan cara sebagai berikut:

1. Klik kanan pada *package operator* → **New** → **JFrame Form**, selanjutnya akan muncul dialog “New JFrame Form”. Isikan properti sebagai berikut:

- A. Class Name : **OperatorAritmatika**
- B. Project : <<Default>>
- C. Location : <<Default>>
- D. Package : <<Default>>
- E. Ceated File : <<Default>>

2. Klik **Finish**.

Langkah selanjutnya adalah mendesain tampilan dengan memasukkan komponen yang dibutuhkan. Adapun desain yang harus anda buat adalah sebagai berikut:



Shape	Komponen (Palette)	Nama Variabel (Variable Name)	Text/ /Model	Status
<b>A</b>	JLabel		OPEERATOR ARITMATIKA	
<b>B</b>	JSeparator			
<b>C</b>	JLabel		Operand 1	
<b>D</b>	JLabel		Operand 2	
<b>E</b>	JTextField	txtOperand1		
<b>F</b>	JTextField	txtOperand2		
<b>G</b>	JLabel		Pilih Operator	
<b>H</b>	JComboBox	cmbOperator	==Pilih== + - / x	
<b>I</b>	JSeparator			
<b>J</b>	JButton	btnHitung	Hitung	
<b>K</b>	JTextField	txtHasil		disabled





### 5.2.6. Menambah Aksi pada Tombol


Untuk menambahkan aksi pada tombol hitung ikuti langkah berikut:

1. Klik kanan pada Button "Hitung" → pilih **Events** → **Action** → **actionPerformed**.
2. Tambahkan kode program berikut ini:

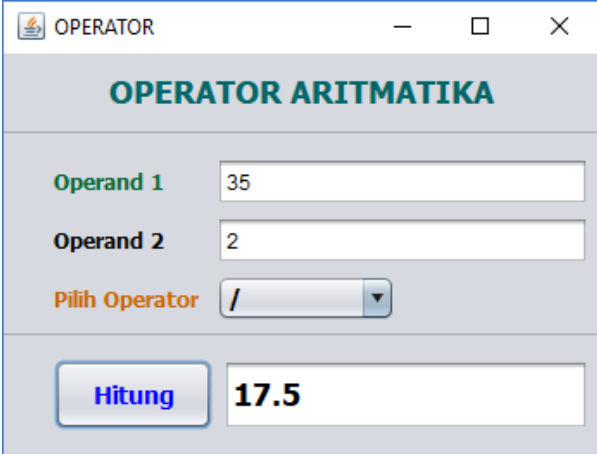
```
private void btnHitungActionPerformed(java.awt.event.ActionEvent evt) {  
    String operand1 = txtOperand1.getText();  
    String operand2 = txtOperand2.getText();  
    if(!operand1.isEmpty() && !operand2.isEmpty()){  
        double hasil;  
        double op1 = Double.valueOf(operand1);  
        double op2 = Double.valueOf(operand2);  
        int operator = cmbOperator.getSelectedIndex();  
        if(operator != Konstanta.UNDEFINED) {  
            hasil = Methods.aritmatika_pemula(operator, op1, op2);  
            String str = String.valueOf(hasil);  
            String hasil_hitung;  
            if(hasil %1 == 0){  
                hasil_hitung = str.substring(0, str.lastIndexOf("."));  
            }else{  
                hasil_hitung = str;  
            }  
            txtHasil.setText(""+hasil_hitung);  
        }  
    }  
}
```

### 5.2.7. Menjalankan Project

Setelah pengkodean selesai, selanjutnya jalankan *Project* dengan cara

1. Menekan nombol **Shift+F6** atau;
2. Meng-klik toolbar 
3. Memilih menu **Run** → **Run Project** (VKL)

### 5.2.8. Hasil Project Latihan Operator





### 5.3. Exercise

#### Ex. 1:

1. Buat Kelas (Java Main Class) dengan nama "RumusPitagoras"
2. Buat method untuk menghitung rumus pitagoras, rumusnya  $c = \sqrt{a^2 + b^2}$
3. Buat variabel "v1" dan "v2" dengan tipe data "double"
4. Buat variabel "hasil" dengan tipe data "double"
5. Buat 2 (dua) "System input" menggunakan kelas "Scanner"
  - Untuk inputan pertama, simpan nilai input dalam variable "v1"
  - Untuk inputan kedua, simpan nilai input dalam variabel "v2"
6. Simpan hasil perhitungan dalam sebuah variabel "hasil"
7. Tampilkan hasil perhitungan

#### Ex. 2:

Tulis program java untuk menyelesaikan persamaan kuadrat berikut :

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

### 5.4. Coursework

Buatlah kalkulator sederhana dengan minimal fungsi dan tampilan sebagai berikut:

