JAVA BASIC PROGRAMMING

Joobshet

1. KOMPETENSI

- Mahasiswa dapat memahami jenis – jenis tipe data
- Mahasiswa dapat memahami jenis – jenis variable
- Mahasiswa dapat memahami jenis – jenis seleksi kondisi
- Mahasiswa dapat memahami jenis – jenis perulangan
- Mahasiswa dapat memahami jenis – jenis array
- Mahasiswa dapat memahami fungsi try-catch

2. PENDAHULUAN

2.1. TIPE DATA

Tipe data adalah tipe atau klasifikasi sebuah nilai(value), objek data atau operasi yang akan menentukan bagaimana nilai - nilai tersebut akan disimpan (Sharan k, 2008).

Sebuah bahasa pemrograman menyediakan beberapa tipe data yang telah ditetapkan, yang dikenal sebagai built-in tipe data. Dan juga dapat membiarkan programmer untuk menentukan jenis data mereka sendiri, yang dikenal sebagai user-defined tipe data.

Apa itu *Identifier* ?

Identifier di dalam bahasa pemrogran java adalah urutan atau rangkaian karakter dengan panjang yang tidak terbatas (Sharan k, 2008). Karakter yang bisa digunakan mencakup Unicode dan java digit. Contoh :

- Unicode : A-Z, a-z, _ (underscore) dan $,
- Java digit : 0-9 ASCII digits.

Ada beberapa hal penting yang perlu diperhatikan untuk bisa mengingat dan lebih memahami implementasi identifier di dalam java , seperti berikut :

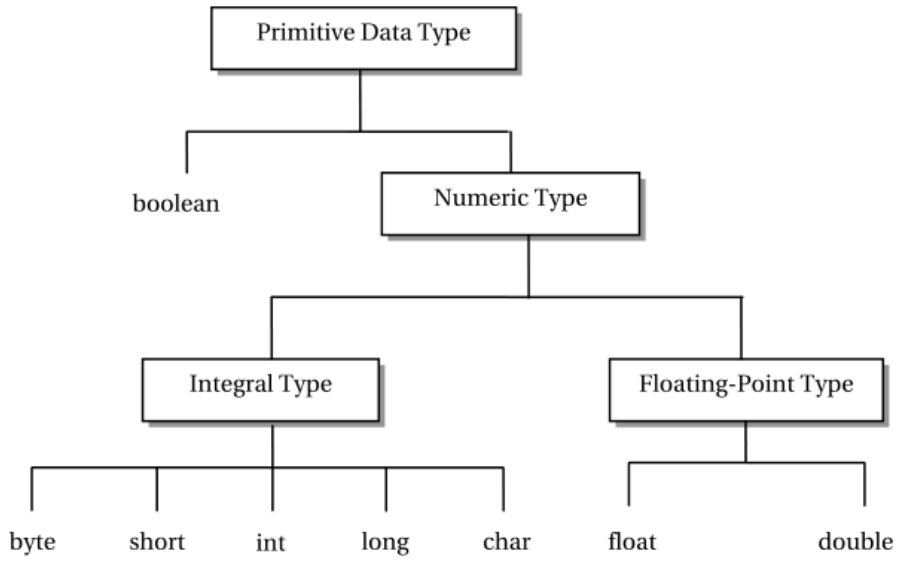
- Tidak terdapat limit,
- Mencakup Unicode dan Java Digit,
- Case sensitive, huruf kecil dan besar dibedakan.

Walaupun identifier bisa dituliskan dengan rangkaian karakter tidak terbatas, dengan Unicode ataupun Java digit, akan tetapi ada beberapa kata yang di dalam bahasa pemrograman java tidak diperbolehkan digunakan untuk sebuah identifier. Kata tersebut di dalam bahasa pemrograman java disebut dengan keywords dan Reserved Words, berikut listnya :

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Primitive tipe data

Ada 8 jenis (tipe data) primitive pada bahasa pemrograman java : byte, short, int, long, char, float, double dan Boolean (Sharan k, 2008). Dari kedelapan tipe diatas dibagi menjadi 4 group yaitu : Boolean, Numeric, Integral dan Floating point.



Sebelum melanjutkan pembahasan detail dari delapan tipe data primitive, ada sedikit pembahasan yang berbeda dari pengelompokan tipe data primitive seperti yang sudah dijelaskan diatas. Pengelompokan tipe data (Schildt H, 2011) dalam buku "Java The Complete Reference Ninth Edition" dijelaskan bahwa tipe data primitive dibagi menjadi empat yaitu :

- Integers : byte, short, int, dan long,
- Floating-point numbers : float dan double,
- Characters : char
- Boolean : boolean , mempunyai nilai true dan false.

Perbedaan pengelompokan dari (Sharan, 2008) dan (Schildt, 2011) adalah terletak dari pengelompokan dari tipe **char,** akan tetapi secara umum kesimpulan dari pembahasan tipe data

char dari kedua penulis diatas sama, yaitu di dalam pemrograman java tipe data char menggunakan 16 bit Unicode.

Tipe data integral

Tipe data integral adalah tipe data numeric yang mempunyai nilai integer. Dalam bahasa pemrograman java, tipe data integral dibagi menjadi lima : byte, short, int, long dan char.

Name	Width	Range
long	64	−9.223.372.036.854.775.808 to 9.223.372.036.854.775.807
int	32	−2.147.483.648 to 2.147.483.647
short	16	−32.768 to 32.767
char	16	0 to 65.535
byte	8	−128 to 127

Byte

Byte adalah tipe data terkecil yang ada di dalam bahasa pemrograman java, dengan range -128 sampai 127. Contoh penerapan tipe data byte :

```
byte b1 = 125;
byte b2 = -11;
```

implementasi tipe data byte tidak boleh melebihi dari range yang sudah ditentukan. Apabila seorang developer melakukan kesalahan dalam memberikan nilai pada tipe data byte, maka akan terjadi error pada implementasinya. Pesan error ini akan secara langsung bisa terdeteksi jika developer menggunakan IDE, akan tetapi tidak akan bisa terdeteksi jika hanya menggunakan standar text editor, seperti notepad, notepad++ atau sublime. Contoh kesalahan penulisan tipe data byte :

```
byte b3 =  150; // akan muncul pesan error , karena melebihi range.
```

Mengenal konversi data

Untuk bisa lebih jelas memahami tentang ***conversion***, ***casting*** dan ***promotion*** bisa dibaca dan dilihat di (https://docs.oracle.com/javase/specs/jls/se7/html/jls-5.html).

```
byte b1 = 15;  // byte
int i1 = 15; // integer
```

Walaupun nilai dari dua variable diatas adalah sama, akan tetapi kita tidak bisa melakukan konversi nilai, seperti di bawah ini :

```
b1 = i1;   // Salah
```

Konversi tipe data yang benar :

```
b1 = (byte)i1;  //  Benar atau
i1 = b1;  //  Benar
```

Percobaan 1:

Langkah percobaan :

1) Buatlah ***class*** dengan nama **TipeDataByte**, dengan nama ***identifier*** : bilangan1 bernilai 10, bilangan2 bernilai 15.
2) Setelah berhasil membuat identifier, buatlah koding untuk menampilkan kedua nilai bilangan tersebut secara terpisah :

```
System.out.println("Keterangan "+identifier);
```

```
public class TipeDataByte {
    public static void main(String[] args) {
        byte bilangan1 = 10;
        // lanjutkan koding disini ...
    }

}
```

Short

Tipe data short adalah 16 bit primitive java integer. Range datanya mulai dari -32768 sampai 32767, atau -2^{15} sampai $2^{15} - 1$.

Contoh implementasi :

```
short s1 = 12905;  // benar
short s2 = -11890;  // benar
short s3 = 569980; // salah (melebihi range).
```

Percobaan 2:

Langkah percobaan :

1) Buatlah class dengan nama **TipeDataShort** dan implementasikan konversi bilangan berikut ini :
2) Berikan keterangan tipe data mana saja yang tidak bisa di konversi (terjadi ***error***) pada list dibawah ini :

```
a) short s1 = 15; // benar
b) byte b1 = 10; // benar
c) s1 = b1; // benar
```

```
   d) b1 = s1; // ...


   e) int num1 = 10; // ...
   f) s1 = num1; // …
   g) s1 = (short)num1; // ...
   h) s1 = 35000; // ...


   i) long num2 = 555L; // ...
   j) s1 = num2; // ...
   k) s1 = (short)num2; // ...
   l) s1 = 555L; // salah, compile-time error
   m) s = (short)555L; // ...
```

Contoh Implementasi :

```
public class TipeDataShort {
    public static void main(String[] args) {
        short s1 = 15; // benar
        byte b1 = 10; // benar
        // ....
    }

}
```

Untuk mempermudah developer dalam mengingat range nilai dari tipe data short, java telah menyediakan class dengan nama ***Short***, yang mendefinisikan nilai dari range tipe data ini. implementasinya sebagai berikut :

```
short max = Short.MAX_VALUE; // untuk menampilkan nilai maksimum dari range,
short min = Short.MIN_VALUE; // untuk menampilkan nilai minimum dari range.
```

Char

Char adalah tipe data java primitive 16 – bit dan unsigned (selalu bernilai positif). Tipe data char mempunyai nilai range yang sama dengan Unicode character yaitu dimulai dari 0 sampai 65535. Dalam implementasinya tipe data char dituliskan dalam satu single quote, seperti dibawah ini :

```
char c1 = 'A';
char c2 = 'L';
char c3 = '5';
char c4 = '/';
```

Integer

Integer adalah tipe data java primitive 32-bit. Range **-2.147.483.648** sampai **2.147.483.647** (-2 31 to 2 31 - 1). Terdapat empat hal penting dalam penulisan bilangan integer , yaitu :

- Menggunakan format bilangan ***Decimal***,
- Menggunakan format bilangan ***Octal***,
- Menggunakan format bilangan ***Hexadecimal***,
- Menggunakan format bilangan ***Binary***.

Contoh implementasi :

```
int num1 = 51966; // format Decimal
int num1 = 0145376; // format Octal, dimulai dengan angka 0
int num1 = 0xCAFE; // format Hexadecimal, dimulai dengan 0x
int num1 = 0b1100101011111110; // format Binary, dimulai dengan 0b
```

Long

Tipe data primitive 64-bit dengan range mulai dari **-9.223.372.036.854.775.808** to **9.223.372.036.854.775.807** (-2 63 to 2 63 - 1).

Contoh implementasi :

```
long num1 = 25L; // Decimal format
long num1 = 031L; // Octal format
long num1 = 0X19L; // Hexadecimal format
long num1 = 0b11001L; // Binary format
```

Tipe data floating point

Tipe data ini bernilai bilangan (angka) real, seperti 3,10 , 0,25 , 75,2 dll. Bahasa pemrograman java menggunakan Standar dari IEEE 754 Floating-Point untuk penyimpanan bilangan floating point (P. W. Kahan, 1997). Tipe data floating-point dalam bahasa pemrograman java adalah float dan double.

Name	Width in Bits	Approximate Range
double	64	4.9e−324 to 1.8e+308
float	32	1.4e−045 to 3.4e+038

Boolean

Tipe data ini digunakan untuk data logic yang bernilai **benar** atau **salah**.

Percobaan 3:

Langkah Percobaan :

1) Buatlah ***class*** dengan nama **TipeBoolean**,
2) Kemudian tuliskan perintah (koding) berikut ini dan lihat hasil dari output program yang ditampilkan,

```
public class TipeBoolean {
    public static void main(String[] args) {
        boolean dataB;

        dataB = true;

        if(dataB){
            // Statement1 ini akan ditampilkan jika nilai dataB adalah true
            System.out.println("data B bernilai "+dataB);
        }else{
            // Statement2 ini akan ditampilkan jika nilai dataB adalah false
            System.out.println("data B bernilai "+dataB);
        }
    }

}
```

3) Lakukan percobaan sekali lagi dengan mengganti dataB dengan kondisi yang berbeda. Catat dan amati hasil percobaan yang sudah anda lakukan.

2.2. VARIABEL

Variabel adalah bagian dari memori yang dapat berisi nilai data. Variable harus dideklarasikan sebelum bisa digunakan.

Terdapat tiga jenis variable pada Java :

- Local variable.
- Instance variable.
- Class/static variable.

Local Variable

- Local variable dideklarasikan dalam methods, constructors, atau blocks.
- Access modifiers (pengubah akses) tidak dapat digunakana pada local variables.
- Local variable hanya dapat dilihat pada methods, constructors atau blocks yang terdeklarasi.
- Local variable diimplementasikan pada tingkat level internal.
- Tidak ada nilai default untuk local variable sehingga local variabel harus dinyatakan dan nilai awal harus ditetapkan sebelum pertama kali digunakan.

Contoh :

Pada contoh kali ini usia adalah local variable. Ini didefinisikan dalam usiaKamu() method dan ruang lingkupnya terbatas hanya untuk method ini saja.

```
public class Test{
   public void usiaKamu(){
      int usia = 0;
      usia = usia + 7;
      System.out.println("Usia kamu adalah : " + usia);
   }

   public static void main(String args[]){
      Test test = new Test();
      test.usiaKamu();
   }
}
```

Instance Variables

- Instance variables dinyatakan dalam kelas, tetapi di luar method, constructor atau block apapun.
- Access modifiers dapat diberikan pada instance variables.
- Instance variables dapat dilihat semua method, constructor dan block pada class.
- Instance variables memiliki nilai default. Nilai default adalah 0, untuk boolean adalah false dan untuk object referencess adalah null. Nilai dapat diberikan selama deklarasi atau dalam konstruktor.
- Instance variables dapat diakses langsung dengan memanggil nama variable di dalam class. Namun dalam static method dan kelas yang berbeda (misalnya saat variable diberikan aksesibilitas) harus disebut menggunakan nama yang memenuhi syarat.ObjectReference.VariableName.

Contoh :

```
public class KaryawanKontrak{
   // berikut adalah instance variable yang terlihat oleh semua child class(class anak).
   public String nama;

   // variable gaji terlihat hanya pada class KaryawanTetap saja.
   private double gaji;

   // variable nama ditentukan pada constructor.
   public KaryawanKontrak(String namaKaryawan){
      nama = namaKaryawan;
   }

   // menentukan nilai pada variable gaji.
   public void setGaji(double gajiKaryawan){
      gaji = gajiKaryawan;
   }
```

```
   // method untuk mencetak detail karyawan.
   public void cetakKaryawan(){
      System.out.println("nama : " + nama);
      System.out.println("gaji :" + gaji);
   }

   public static void main(String args[]){
      KaryawanKontrak karyawanSatu = new KaryawanKontrak("Rika");
      karyawanSatu.setGaji(3000000);
      karyawanSatu.cetakKaryawan();
   }
}
```

Class/Static Variables

- Class variables juga dikenal sebagai static variables dideklarasikan dengan kata kunci static dalam kelas, tetapi di luar method, constructor atau block.
- Hanya akan ada satu turunan dari setiap class variables pada class, terlepas dari berapa banyak objek yang diciptakan.

Contoh :

```
public class KaryawanMagang{
   // variable gaji adalah private static
   private static double gaji;

   // DEPARTMENT adalah constant
   public static final String DEPARTMENT = "Development ";

   public static void main(String args[]){
      gaji = 250000;
      System.out.println(DEPARTMENT + "gaji rata-rata :" + gaji);
   }
}
```

2.3. SELEKSI KONDISI

Bahasa pemrograman java menyediakan dua statement untuk seleksi sebuah kondisi : **if** dan **switch**. Statement seleksi ini mengijinkan developer untuk mengontrol alur dari eksekusi program berdasarkan kondisi yang diinginkan selama proses run-time program.

IF

Statement berdasar dari kondisi Boolean (bernilai benar dan salah).

```
if(kondisi){
   statement;  // jika kondisi bernilai benar
}else{
   Statement;  // jika kondisi bernilai salah
}
```

Percobaan 4:

Langkah Percobaan :

1) Buatlah class dengan nama KondisiIf,
2) Implementasikan koding di bawah ini, analisa dan pelajari hasil output dari koding.

```
public class KondisiIf {
    public static void main(String[] args) {
        int a = 10, b = 12;

        if(a<b){
            System.out.println("Bilangan "+a+", lebih kecil dari "+b);
        }else{
            System.out.println("Bilangan "+a+", lebih besar dari "+b);
        }
    }

}
```

3) Setelah itu tambahkan variable c dengan nilai 20.
4) Rubah kondisi di dalam if menjadi seperti ini : `if(a<b && c> a+b)`.
5) Amati hasil dari percobaan. Jika statement tidak sesuai dengan perhitungan logic yang sebenarnya, coba betulkan kondisi yang seharusnya seperti apa.

Nested ifs

Statement if di dalam if.

```
if(kondisi1){ // Jika kondisi benar (kondisi kedua) dieksekusi.
   if(kondisi2){
      Statement1;
   }else{
      Statement2;
   }
else{
  Statement3;  // Jika kondisi pertama salah , maka statement ini akan dieksekusi
}
```

Percobaan 5:

Langkah percobaan :

1) Buatlah ***class*** dengan nama **KondisiNestedIf**,
2) Kemudian tuliskan koding dibawah ini dan lihat hasil output yang ditampilkan.
3) Lakukan beberapa penggantian nilai dari variable dan analisa hasil outputnya.

```
public class KondisiNestedIf {
    public static void main(String[] args) {
        String gender = "Mr";
        int usia = 25;

        if(gender=="Mr"){
            if(usia > 20){
                System.out.println("Diterima");
            }else{
                System.out.println("Ditolak");
            }
        }else{
            System.out.println("Khusus Laki-laki");
        }
    }

}
```

if-else-if

```
if(kondisi1){
   Statement; // Jika kondisi pertama bernilai benar.
}else if(kondisi2){
   Statement; // Jika kondisi pertama salah, dan kondisi kedua benar.
}else{
   Statement; // jika kondisi pertama dan kedua bernilai salah.
}
```

Percobaan 6:

Langkah percobaan :

1) Buatlah *class* dengan nama **KondisiIfElseIf**,
2) Kemudian implementasikan koding di bawah ini dan lihat hasil outputnya.
3) Lakukan penggantian nilai pada tipe data dan lihat kembali hasil outputnya.

```
public class KondisiIfElseIf {
    public static void main(String[] args) {
        int nilai = 76;
        char grade;
```

```
        if (nilai >= 90) {
            grade = 'A';
        } else if (nilai >= 80) {
            grade = 'B';
        } else if (nilai >= 70) {
            grade = 'C';
        } else if (nilai >= 60) {
            grade = 'D';
        } else {
            grade = 'F';
        }
        System.out.println("Grade = " + grade);
    }

}
```

4) Rubah kondisi , jika grade A hanya boleh di tampilkan ketika nilai adalah 100.

Switch

```
switch (ekspresi){ // ekspresi yang dikerjakan
  case 0 :
             Statement1
             break; // untuk menghentikan statement1
  case 1 :
             Statement2
             break; // untuk menghentikan statement2
 default :
             Statement3 // dieksekusi jika tidak ada case yang benar.
}
```

Percobaan 7:

Langkah percobaan :

1) Buatlah *class* dengan nama StatementSwitch,
2) kemudian implementasikan koding dibawah ini. Lihat dan amati hasil outputnya.

```
public class StatementSwitch {
    public static void main(String[] args) {
        for(int i = 0; i<5; i++)
            switch(i){
                case 0:
                    System.out.println("Nilai dari i adalah 0");
                    break;
                case 1:
                    System.out.println("Nilai dari i adalah 1");
```

```
                    break;
                default :
                    System.out.println("Nilai dari i lebih dari 1");
            }
    }

}
```

3) Rubah ekspresi agar case yang ditampilkan hanya yang pertama saja.

2.4. PERULANGAN

Terdapat tiga perulangan dalam bahasa pemrograman java yaitu for, while dan do-while.

While

Perulangan while digunakan untuk eksekusi sebuah statement secara berulang – ulang ketika kondisinya bernilai benar.

```
While (kondisi){
   Statement; // statement (loop) ini akan terus dikerjakan , jika kondisi masih bernilai benar.
}
```

Percobaan 8 :

Langkah percobaan :

1) Buatlah *class* dengan nama **PerulanganWhile**, kemudian implementasikan koding di bawah ini. Lihat dan amati hasil outputnya.

```
public class PerulanganWhile {
    public static void main(String[] args) {
        int i = 10;

        while (i>0){
            System.out.println("Nilai i "+i);

            i--;
        }
    }
}
```

2) Rubah body loop , agar hanya bisa menampilkan bilangan genap saja.

Do-while

Perulangan do-while secara prinsip hamper sama dengan perulangan while, hanya saja pada perulangan while, jika dalam proses eksekusi terdapat kondisi yang bernilai false, maka body loop tidak akan dieksekusi, sedangkan untuk do-while akan selalu mengeksekusi statement dalam body loop.

```
do{
```

```
Statement;
}while(Kondisi);
```

Percobaan 9 :

Membuat perulangan do-while

Langkah percobaan :

1) Buat *class* dengan nama **PerulanganDoWhile**, kemudian implementasikan koding berikut ini :

```
public class PerulanganDoWhile {
    public static void main(String[] args) {
        int i = 10;
        int j = 0;
        do{
            j +=i;
            i--;

        }while(i>5);
        System.out.println("Nilai j "+j);
        System.out.println("Nilai i "+i);
    }

}
```

2) Rubah statement agar nilai akhir j kurang dari 25.

For

Struktur for pada umumnya digunakan untuk melakukan pengulangan yang banyaknya sudah pasti atau sudah diketahui sebelumnya. Dalam pengulangan for kita harus menentukan nilai awal pengulangan dan nilai akhir pengulangan.

Pengulangan for tidak membutuhkan counter untuk menaikan variabel karena sudah disebutkan pada salah satu parameter pengulangan. Bentuk umum pengulangan for adalah sebagai berikut:

```
for (nilai inisialisasi awal; kondisi loop; iterasi;) {
//body loop, statement yang akan diulang
}
```

Percobaan 10 :

Membuat perulangan for

Langkah percobaan :

1) Buatlah class dengan nama PerulanganFor, kemudian implementasikan koding dibawah ini :

```
public class PerulanganFor {
    public static void main(String[] args) {
```

```
        for(int i=0; i<=10; i++){
            System.out.println("Nilai dari i "+i);
        }
    }

}
```

2) Setelah koding dieksekusi, rubah ekspresi agar hasil nilai i yang ditampilkan hanya berupa bilangan ganjil.

2.5. ARRAY

Array adalah sekelompok data sejenis yang disimpan ke dalam variabel dengan nama yang sama, dengan memberi indeks pada variabel untuk membedakan antara yang satu dengan yang lain.

Cara mendeklarasikan suatu array adalah sebagai berikut:

```
tipe_array nama_array[];
tipe_array[] nama_array;
```

Format penulisannya adalah sebagai berikut:

```
nama_array = new tipe_array[total_elemen_array];
```

Contoh :

```
int nilai[];
nilai = new int[5];
```

Kita dapat membuat array multi dimensi dengan cara menambahkan tanda [] sebanyak dimensi yang ingin dibuat. Sebagai contoh adalah sebagai berikut:

```
int x[][] = new int[3][4];
```

Ket :
Baris statement diatas berarti kita ingin membuat array berdimensi 2, dengan 3 elemen di dimensi ke-1 dan 4 elemen di dimensi ke-2.

Untuk mengetahui panjang dari suatu array yang telah kita buat, kita dapat memakai properti length. Adapun format untuk menggunakan length adalah sebagai berikut:

var_array.length	=>	total elemen array pada dimensi 1
var_array[i].length	=>	total elemen array pada dimensi 2 untuk indeks ke-i pada dimensi 1
var_array[i][j].length	=>	total elemen array pada dimensi 3 untuk indeks ke-i pada dimensi 1 dan indeks ke-j pada dimensi 2 dan seterusnya.

Isi dari suatu array dapat kita kopi pada array yang lain dengan memanfaatkan method arraycopy() pada class System. Format penulisannya sebagai berikut :

```
System.arraycopy(array1, p1, array2, p2, n);
```

Ket :
array1 = array asal/sumber pengkopian
array2 = array tujuan pengkopian
p1 = posisi indeks awal pengkopian pada array asal
p2 = posisi indeks awal pengkopian pada array tujuan
n = banyaknya elemen array yang akan dikopi

Suatu array juga dapat me-refer (merujuk) ke array yang lain, dengan kata lain merujuk pada alamat memori yang sama. Sebagai contoh adalah program berikut ini:

```
int nilai[] = {10, 20, 30};
int result[];
result = nilai;
```

Percobaan 11 :

Mengakses elemen array

```
public class Array1 {
	public static void main(String args[]) {
		int nilai[]=new int[3];
		nilai[0]=70;
		nilai[1]=80;
		nilai[2]=65;
		double ratarata=0.0;

		for(int i=0; i<nilai.length; i++) ratarata+=nilai[i];
			ratarata/=nilai.length;
			System.out.println("Nilai rata-rata = " + ratarata);
	}
}
```

Percobaan 12 :

Mengakses elemen array berdimensi 2

```
import java.text.NumberFormat;
public class Array2 {
	public static void main(String args[]) {
		NumberFormat nf=NumberFormat.getInstance();
		nf.setMaximumFractionDigits(3);
```

```
            int nilai[][]=new int[2][3];
            nilai[0][0]=85;
            nilai[0][1]=81;
            nilai[0][2]=78;
            nilai[1][0]=65;
            nilai[1][1]=73;
            nilai[1][2]=71;
            String MK[]={"RPL", "PBO"};
            double rataRataMK[]=new double[nilai.length];
            for (int i=0; i<nilai.length; i++) {
                  for (int j=0; j<nilai[0].length; j++) {
                        rataRataMK[i]+=nilai[i][j];
                  }
                  rataRataMK[i]/=nilai[0].length;
            }
            System.out.println("Nilai Mata Kuliah\n");
            System.out.println("MK\tMinggu1\tMinggu2\tMinggu3\t
                                          Rata-Rata");
            for (int i=0; i<nilai.length; i++) {
                  System.out.print(MK[i] + "\t");
                  for (int j=0; j<nilai[0].length; j++) {
                        System.out.print(nilai[i][j] + "\t");
                  }
                  System.out.print(nf.format(rataRataMK[i])+"\n");
            }
      }
}
```

2.6. TRY-CATCH

Untuk menangani ***error*** di Java, digunakan sebuah statement yang bernama ***try - catch***. Statement tersebut digunakan untuk mengurung eksekusi yang menampilkan ***error*** dan dapat membuat program tetap berjalan tanpa dihentikan secara langsung. ***Error*** yang ditangani oleh ***try - catch*** biasa disebut dengan ***exception***.

Ada beberapa hal yang perlu diingat ketika akan menggunakan try - catch di Java:

- Kita dapat membuat multiple try-catch,
- Kita dapat menambahkan statement finally untuk menangani berbagai hal ketika error terjadi atau tidak,
- Kita dapat membuat exception sendiri disamping menggunakan bawaan Java.

Percobaan 13 :

Langkah percobaan :

1) Buat *class* dengan nama **SourceErrorExp**, yang berisi source code yang mengandung error seperti dibawah ini :

```
class SourceErrorExp {
    public static void main (String[] args){
        System.out.println("awal program");

        int x = 10;

        x = x / 0;

        System.out.println(x);

        System.out.println("akhir program");
    }
}
```

2) Bila kita jalankan di ***console***, maka akan muncul output seperti berikut:

```
$ javac SourceErrorExp.java
$ java SourceErrorExp
awal program
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at SourceErrorExp.main(SourceErrorExp.java:7)
```

3) Sebelum melakukan percobaan ke – 14, perbaiki error coding agar hasil print out bisa ditampilkan.

Berbeda bila kita kurung operasi pembagian nol diatas dengan try - catch, maka hasil eksekusi program akan sedikit berbeda:

Percobaan 14 :

Langkah percobaan :

1) Buatlah *class* dengan nama **SourceErrorExp2** , dan tuliskan kode program seperti dibawah ini :

```
class SourceErrorExp2{
    public static void main (String[] args){
        System.out.println("awal program");

        int x = 10;

        try {
            x = x / 0;
```

```
        }
        catch (Exception e){
            e.printStackTrace();
            System.out.println("error karena pembagian nol");
        }

        System.out.println(x);

        System.out.println("akhir program");
    }
}
```

2) Sekarang kita eksekusi kembali program diatas. Maka program akan dijalankan sampai bagian akhir program walaupun terjadi error di tengah - tengah eksekusi:

```
$ javac SourceErrorExp2
$ java SourceErrorExp2
awal program
java.lang.ArithmeticException: / by zero
    at SourceErrorExp2.main(SourceErrorExp2.java:10)
error karena pembagian nol
10
akhir program
```

Percobaan 15 :

Mari kita coba lagi contoh yang lain, dimana kita akan menggunakan statement finally. Tuliskan source code berikut:

```
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.io.FileNotFoundException;

class SourceErrorExp3 {
    public static void main (String[] args) {
        System.out.println("awal program");

        try {
            System.out.println("Baris ini tidak akan dieksekusi - 1.2");

            File f = new File ("hello.txt");
            InputStream fis = new FileInputStream(f);

            System.out.println("Baris ini tidak akan dieksekusi - 1.3");
```

```
        }
        catch (Exception e){
            e.printStackTrace();
        }
        finally {
            System.out.println("Baris di dalam finally akan dieksekusi");
        }

        System.out.println("akhir program");
    }
}
```

Bila kita jalankan source code diatas, maka bagian finally akan dijalankan baik terjadi error ataupun tidak:

```
$ javac SourceErrorExp3.java
$ java SourceErrorExp3
awal program
Baris ini tidak akan dieksekusi - 1.2
java.io.FileNotFoundException: hello.txt (No such file or directory)
    at java.io.FileInputStream.open0(Native Method)
    at java.io.FileInputStream.open(FileInputStream.java:195)
    at java.io.FileInputStream.<init>(FileInputStream.java:138)
    at SourceErrorExp3.main(SourceErrorExp3.java:14)
Baris di dalam finally akan dieksekusi
akhir program
```

Kode program jika kita hapus bagian pembacaan file:

```
$ javac SourceErrorExp3.java
$ java SourceErrorExp3
awal program
Baris ini tidak akan dieksekusi - 1.2
Baris ini tidak akan dieksekusi - 1.3
Baris di dalam finally akan dieksekusi
akhir program
```

Dynamic Input

Ada beberapa cara untuk meminta inputan dari java. Salah satunya menggunakan pustaka java sendiri yaitu class scanner. Untuk menggunakan kelas ini caranya import java.until.scanner, penempatan kodenya di luar kelas.

```
import java.util.Scanner; // lokasi import class scanner
```

```
public class DynamicInput {
    public static void main(String[] args) {

    }

}
```

Sekarang kita bisa langsung membuat instance seperti ini dalam main.

```
import java.util.Scanner;

public class DynamicInput {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in); // lokasi instance
    }

}
```

Ket :
Implementasi intance persis seperti code di atas kecuali pada *identifier* input bisa di ganti dengan *identifier* yang lain (penamaan yang lain , missal : masukan) .Perlu di perhatikan kita dapat meletakan code di atas tidak cuma di dalam main saja . Dapat kita meletakannya di bagian global (di dalam kelas tidak di dalam method) dengan menambahkan kata static di depannya .Kemudian kita gunakan di dalam method juga bisa.

Contoh jika di letakan di global:

```
import java.util.Scanner;

public class DynamicInputGlobal {
    static Scanner input = new Scanner(System.in); // lokasi global
    public static void main(String[] args) {

    }
}
```

Contoh jika diletakan di dalam method:

```
import java.util.Scanner;

public class DynamicInputMethod {
    static public void dynamicScanner(){
        Scanner input = new Scanner(System.in);
```

```
    }
    public static void main(String[] args) {

    }
}
```

Percobaan 16:

Implementasi class scanner :

Langkah percobaan :

1) Buat class dengan nama CobaScanner,
2) Masukkan koding seperti di bawah ini :

```
import java.util.Scanner;
public class CobaScanner {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.println("Masukkan Angka :");
        int angka = input.nextInt(); // variable dengan tipe data tertentu (inputan)

        System.out.println("==============");
        System.out.println("Anda memasukkan angka : "+angka);
    }

}
```

3) Setelah memasukkan kode diatas , amati hasil output dari eksekusi program. Berikan inputan dengan nilai yang berbeda (tetap dalam satu tipe data ***int***), setelah itu berikan inputan dengan tipe data yang berbeda (missal ***string***). Teliti dan amati apa yang terjadi setelah diberi perlakuan diatas.
4) Buatlah inputan kedua dengan tipe data yang berbeda. Amati dan catat hasil percobaan anda.

3. TUGAS

Buatlah program kalkulator sederhana yang didalamnya terdapat (dynamic input (scanner) , try-catch (aritmatic), percabangan (if dan switch)).

Dengan tampilan output seperti dibawah ini :

```
run:
PILIH :
1.Tambah
2.Kurang
3.Kali
4.Bagi

1
Angka 1 :
20
Angka 2 :
22
angka1 + angka2 = 42.0

Tekan Y atau YA jika ingin berhenti, dan T untuk melanjutkan
```