

Pemrograman Jaringan 0

anton@ukdw.ac.id

Deskripsi

- Matakuliah: Pemrograman Jaringan
- SKS: 3
- Dosen: Antonius Rachmat C, S.Kom, M.Cs
- Waktu: Senin, 7.30-10.20
- Ruang: LAB E
- Deskripsi:
 - Mempelajari konsep-konsep jaringan pada layer aplikasi dan teknik pemrogramannya menggunakan Java

Tujuan

- memahami bagaimana Internet bekerja, arsitekturnya dan protokol TCP/IP
- memahami bagaimana input dan output pada Java
- mampu mengembangkan program client dan server dengan menggunakan protokol *User Datagram Protocol* (UDP) dan *Transport Control Protocol* (TCP)
- mampu mengembangkan aplikasi multithread
- memahami protokol Hyper-Text Transfer Protocol (HTTP), dan mengetahui bagaimana mengakses *World Wide Web* menggunakan Java
- mampu mengembangkan aplikasi terdistribusi seperti *Remote Method Invocation* (RMI) dan CORBA

Silabus

- Perkenalan + Refresh Java
- Pengantar Jaringan 1
 - Jaringan Komputer & Protokol
 - IP Address, Port, Socket
 - TCP dan UDP
 - Internet
- Pengantar Jaringan 2
 - Client/Server Model
 - Middleware
 - Konsep dasar web
 - HTTP, URI, URL, MIME

Silabus-2

- Java OOP – mungkin tidak perlu?
 - Class, Object
 - Polymorfism, Inheritance, Encapsulation
 - JavaDoc
- IO dan Stream
 - File
 - Input, Output, Filter, dan Reader
- Pemrograman HTTP
 - Protokol HTTP
 - Metode Get dan Post
 - InetAddress, URL, URI Class

Silabus-3

- Pemrograman Socket
 - Connection Oriented
- Threading
 - Multithreading, Synchronization
- Socket Multithreading, JAR dan JDBC
 - Add, insert, delete, edit
- Pemrograman Socket
 - Connectionless Oriented

Silabus-4

- Komunikasi Antar Obyek
 - Obyek Serialization
- Remote Method Invocation
 - Konsep & Aplikasi
- CORBA
 - Konsep & IDL
 - Pemrograman CORBA

Daftar Pustaka

- Budi Susanto, Pemrograman Client/Server dengan Java 2, 2003, Jakarta : PT. Elexmedia Komputindo
- Elliotte Rusty Harold, Java Network Programming, 3rd Edition, 2004, O'Reilly
- Vinay Chhabra, A Beginners Guide to RMI, www.universalteacher.com
- Java™ Network Programming and Distributed Computing by David Reilly & Michael Reilly, Addison Wesley, 2002
- An Introduction to Network Programming with Java, Jan Graba, Springer, 2007
- Java Cookbook, 2nd Edition, Ian F. Darwin, O'Reilly, 2004

Distribusi Nilai

- 85-100 A
- 80-<85 A-
- 75-<80 B+
- 70-<75 B
- 65-<70 B-
- 60-<65 C+
- 55-<60 C
- 45-<55 D
- <45 E

Komponen Penilaian

- TTS : 25
- TAS : 25
- Tugas : 15
 - Carilah program jaringan di Internet yang **sdh jadi**, analisa, buat laporannya, kumpul TTS!
- Tugas Lab: 35
 - DOS, Socket, JDBC, RMI / Corba

Pengantar Java

- Java adalah bahasa pemrograman yang sangat powerfull
- Write Once Run Everywhere - multiplatform
- Mendukung OOP murni
- Versi terbaru 1.6
- J2RE = runtime environment
- J2SE = standard edition
 - J2SE can be used to develop client-side standalone applications or applets.
- J2EE = enterprise edition
 - J2EE can be used to develop server-side applications such as Java servlets and Java ServerPages.
- J2ME = micro edition
 - J2ME can be used to develop applications for mobile devices such as cell phones.

Sejarah

- Java pertama lahir dari The Green Project, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992.
- Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, **James Gosling** dan Bill Joy, beserta sembilan pemrogram lainnya dari Sun Microsystems.
- Lahirlah maskot Duke yang dibuat oleh Joe Palrang



Sejarah (2)

- Nama Java pertama adalah **project Oak**, diambil dari pohon oak yang tumbuh di depan jendela ruangan kerja James Gosling.
- Nama Oak ini tidak dipakai untuk versi release Java karena sebuah perangkat lunak sudah terdaftar dengan merk dagang tersebut, sehingga diambil nama penggantinya menjadi "Java".
- Nama Java ini diambil dari kopi murni yang digiling langsung dari biji (kopi tubruk) kesukaan Gosling.

The Java programming environment

- Compared to C++: *simple*
 - no header files, macros, pointers and references, unions, operator overloading, templates, etc.
- *Object-oriented*
- *Distributed*: RMI, Servlet, Distributed object programming.
- *Robust*: Strong typing + no pointer + garbage collector
- *Secure*: Type-safety + access control
- *Architecture neutral*
- *Portable*
- *Compiled & Interpreted*
 - *High performance*
 - Just in time compilation + runtime modification of code
- *Multi-threaded*

The Java programming environment

- *Java byte code*: Intermediate representation for Java programs
- *Java compiler*: Transform Java programs into Java byte code
- *Java interpreter*: Read programs written in Java byte code and execute them
- *Java virtual machine*: Runtime system that provides various services to running programs
- *Java programming environment*: Set of libraries that provide services such as GUI, data structures, etc.
- *Java enabled browsers*: Browsers that include a JVM + ability to load programs from remote hosts

Editor

- Rekomendasi text editor: Jcreator dan KAWA Java Editor
- Rekomendasi GUI editor: NetBeans, Jbuilder, dan Eclipse, Xcode IDE (for MAC)

Contoh Program Java Sederhana

```
public class HelloJava {  
    public static void main(String[] args) {  
        System.out.println("Hello Java, " +  
            "salam kenal dengan Anda!");  
    }  
}
```

Prinsip Pembuatan

- pada satu file source java (.java) hanya diperbolehkan terdapat definisi satu public class.
- dalam satu public class java akan mencari sebuah method (subprogram) yang bersifat statis dan public yang bernama **main**.
- method **main** ini menerima parameter array string.
- Jika ditemukan maka Java akan mengeksekusi blok perintah dalam method main tersebut. Jika tidak ada method main maka Java tidak akan mengeksekusi apapun.
- Tanpa method main, program Java bisa dikompilasi, tidak bisa dieksekusi
- Sintaks penulisan keyword ataupun identifier dalam Java bersifat case-sensitive.

How are Java programs written?

- Define a class HelloWorld and store it into a file: HelloWorld.java:

```
public class HelloWorld {  
    public static void main (String[] args) {  
        System.out.println("Hello, World");  
    }  
}
```

- Compile HelloWorld.java

```
javac HelloWorld.java
```

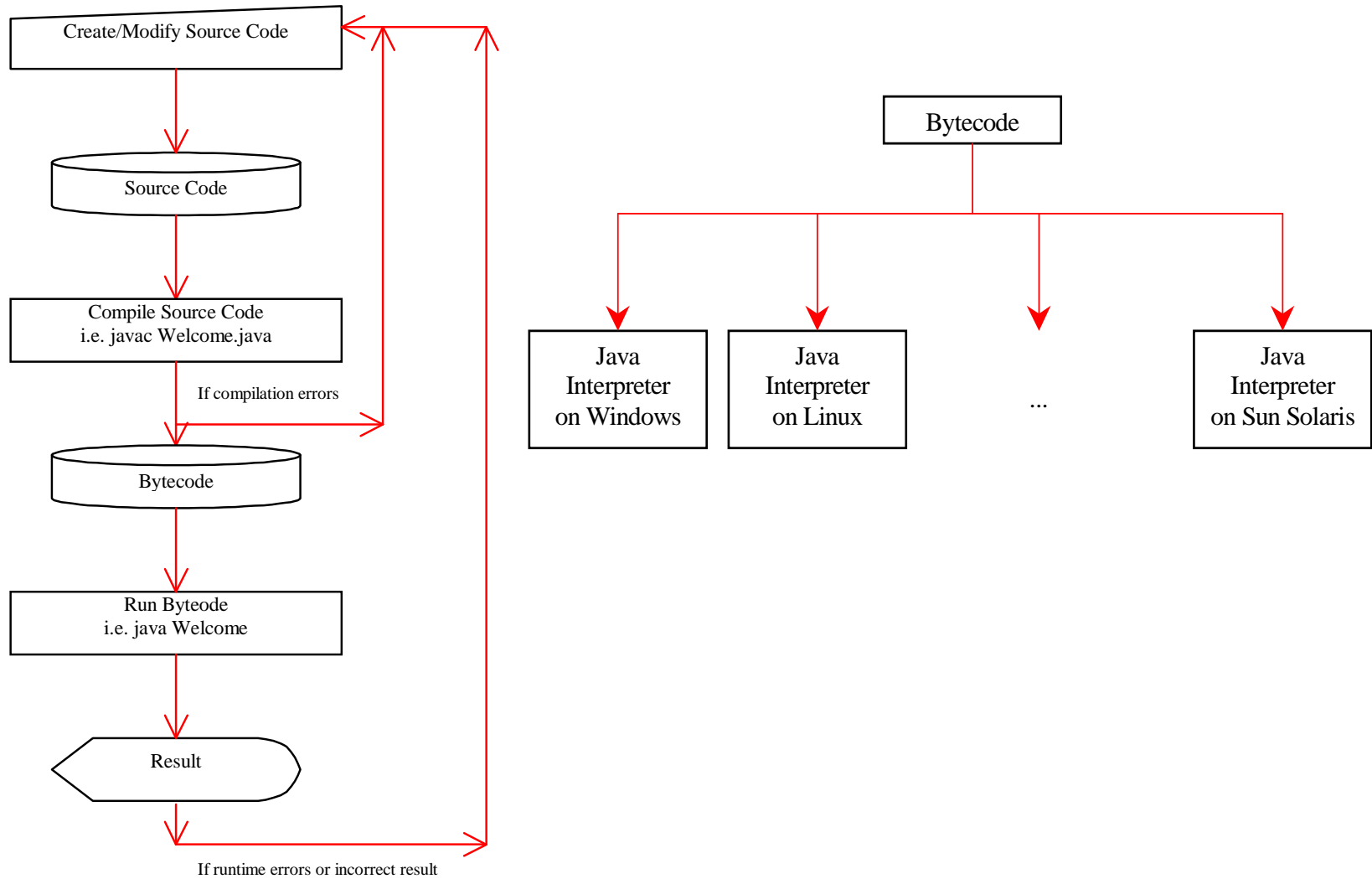
```
Output: HelloWorld.class
```

- Run

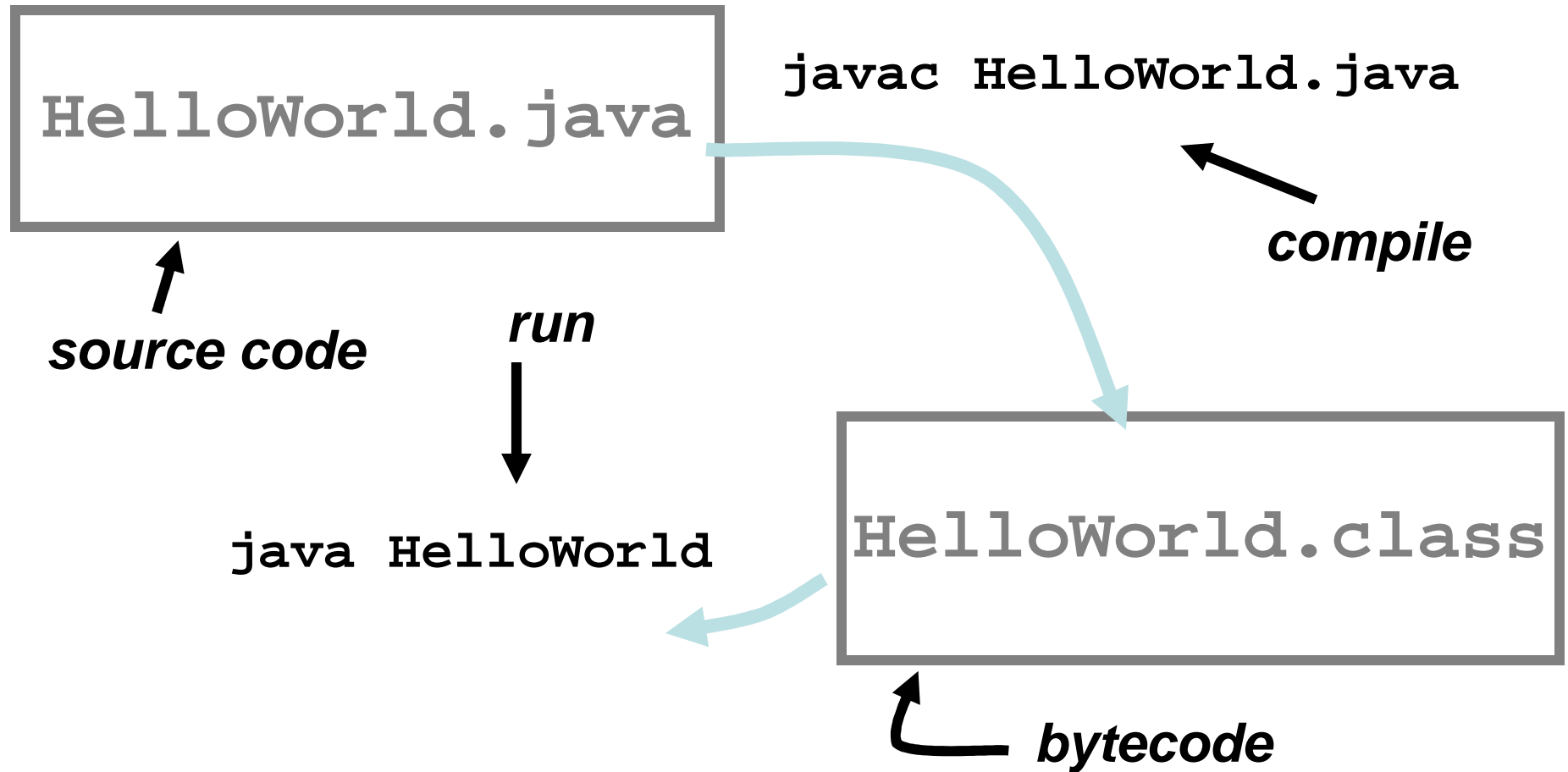
```
java HelloWorld
```

```
Output: Hello, World
```

Compilation & Execution Phase



Compiling and Running



Java bytecode and interpreter

- bytecode is an intermediate representation of the program (class).
- The Java interpreter starts up a new “Virtual Machine”.
- The VM starts executing the users class by running it's `main()` method.

PATH and CLASSPATH

- The *java_home/bin* directory is in your \$PATH
- If you are using any classes outside the java or javax package, their locations are included in your \$CLASSPATH

Anatomy of a Java Program

- Comments : // dan /* ... */
- Package : dianalogikan sebagai folder
- Modifiers : public, private, protected
- Statements : diakhiri dengan semicolon (;)

- Blocks :

```
public class Test { ←  
    public static void main(String[] args) { ←  
        System.out.println("Welcome to Java!"); ←  
    } ←  
} ←
```

Class block
Method block

- Classes : a template or blueprint for objects
- Methods : behaviour / function of class
- The main method

Tipe Data

- Terdapat beberapa tipe data primitif atau dasar :
 - Numerik bulat: int, byte, short, long
 - Numerik pecahan: float, double
 - Logika: boolean
 - Karakter: char

Reference Types

- Objects and Arrays are *reference types*
- Primitive types are stored as values.
- Reference type variables are stored as references (pointers that we can't mess with).
- There are significant differences!

Primitive vs. Reference Types

```
int x=3;
```

```
int y=x;
```

```
Point p = new Point(2.3,4.2);
```

```
Point t = p;
```

```
Point p = new Point(2.3,4.2);
```

```
Point t = new Point(2.3,4.2);
```

How are variables declared?

Fibonacci:

```
class Fibonacci {  
    public static void main(String[] arg) {  
        int lo = 1;  
        int hi = 1;  
        System.out.println(lo);  
        while (hi < 50) {  
            System.out.println(hi);  
            hi = lo + hi;  
            lo = hi - lo;  
        }  
    }  
}
```

How to define expressions?

- Arithmetic: +, -, *, /, %, =

8 + 3 * 2 / 4

Use standard precedence and associativity rules

- Predicates: ==, !=, >, <, >=, <=

```
public class Demo {  
    public static void main (String[] argv) {  
        boolean b;  
        b = (2 + 2 == 4);  
        System.out.println(b);  
    }  
}
```

Casting

- Casting diperlukan ketika kita akan “memaksa” penyesuaian dari satu tipe data ke tipe data lain.
- Pada pemrograman berbasis objek casting diperlukan untuk menyesuaikan suatu tipe objek (class) ke tipe objek (class) lain.

Contoh Casting

```
public class cast {
    public static void main(String[] args) {
        int i, j;
        double r;

        i = (int) (9.0/4.0) ;
        System.out.println(i) ;

        //bagaimana dengan yang ini? Berapa hasilnya?
        i = 10;
        j = 5;
        r = i/j;
        System.out.println(r) ;

        //bagaimana dengan yang ini? Berapa hasilnya?
        r = (double)i / (double)j ;
        System.out.println(r) ;
    }
}
```

Konversi/Casting

- Widening conversions
 - `int a = 123123123;`
 - `float b = a; //ok`
- Narrowing conversions
 - `long a = 123123L`
 - `int b = a; //compiler error`
 - `int b = (int) a; //ok`
 - `long d = 123123123123L`
 - `int e = (int) d; //loss of magnitude`

Konversi Tipe Data

- Konversi String ke Numerik

- `int i = Integer.valueOf("22").intValue();`
- `long l = Long.valueOf("23132323").longValue();`
- `double x = Double.valueOf("20100.025").doubleValue();`
- `float y = Float.valueOf("200.45").floatValue();`

Atau

<i>type</i>	<i>Example statement</i>
<code>int</code>	<code>i = Integer.parseInt(s);</code>
<code>long</code>	<code>l = Long.parseLong(s);</code>
<code>float</code>	<code>f = Float.parseFloat(s);</code>
<code>double</code>	<code>d = Double.parseDouble(s);</code>

Konversi Tipe Data

- Non Decimal Integer

<i>type</i>	<i>Example statement</i>
int	<code>i = Integer.parseInt(s, radix);</code>
long	<code>l = Long.parseLong(s, radix);</code>

- To convert string containing the hexadecimal number "F7" to an integer

`i = Integer.parseInt("F7", 16)`

Number to string conversion

- Concatenation (+): Anything concatenated to a string is converted to string (eg, "weight = " + kilograms).
- *java.text.DecimalFormat* gives you precise control over the formatting of numbers (number of decimal places, scientific notation, locale formatting, ...).
- Individual wrapper class methods, eg, `Integer.toString(i)`. concatenation works as well for the simple cases, but there are some interesting additional conversions here.
- *No conversion required.* Some common system methods will take any type and convert it, eg, `System.out.println()`.

Contoh

- Contoh 1:

```
float price = 23.99f;
```

```
String priceStr = "" + price;
```

- Contoh 2:

```
int years = 22;
```

```
String yearsStr = Integer.toString(years);
```

Concatenation

```
int x = 42;
String s;
s = x;           // ILLEGAL
s = "" + x;     // Converts int 42 to String "42"
s = x + " is OK"; // Assigns "42 is OK" to s.
s = "" + 3.5;   // Assigns "3.5" to s
s = "" + 1.0/3.0; // Assigns "0.33333333333333333333" to s
```

How are simple methods defined?

Every method is defined inside a Java class definition

```
public class Movie {
    public static int movieRating(int s, int a, int d) {
        return s+a+d;
    }
}
public class Demo {
    public static void main (String argv[]) {
        int script = 6, acting = 9, directing = 8;
        displayRating(script, acting, directing);
    }
    public static void displayRating(int s, int a, int d){
        System.out.print("The rating of this movie is");
        System.out.println(Movie.movieRating(s, a, d));
    }
}
```

```

import java.util.*;

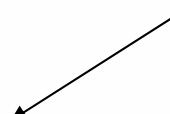
public class Bool {
    public static void main(String[] args) {
        Random rand = new Random();
        int i = rand.nextInt() % 100;
        int j = rand.nextInt() % 100;

        prn("i = " + i);
        prn("j = " + j);
        prn("i > j is " + (i > j));
        prn("i < j is " + (i < j));
        prn("i >= j is " + (i >= j));
        prn("i <= j is " + (i <= j));
        prn("i == j is " + (i == j));
        prn("i != j is " + (i != j));
        //prn("i && j is " + (i && j));
        //prn("i || j is " + (i || j));
        //prn("!i is " + !i);
        prn(" (i < 10) && (j < 10) is "
            + ((i < 10) && (j < 10)) );
        prn(" (i < 10) || (j < 10) is "
            + ((i < 10) || (j < 10)) );
    }

    static void prn(String s) {
        System.out.println(s);
    }
}

```

Tidak bisa
pada int



Flow Control

- IF Syntax :
 - if(kondisi) <statement>
 - If(kondisi) {
 <statements>
} else {
 <statements>
}

```
public class IfElse {
    static int test(int testval, int target) {
        int result = 0;
        if(testval > target)
            result = +1;
        else if(testval < target)
            result = -1;
        else
            result = 0;
        return result;
    }

    public static void main(String[] args) {
        System.out.println(test(10, 5));
        System.out.println(test(5, 10));
        System.out.println(test(5, 5));
    }
}
```

Flow Control

- Switch

```
switch(integral-selector) {  
    case integral-value1 : statement; break;  
    case integral-value2 : statement; break;  
    case integral-value3 : statement; break;  
    case integral-value4 : statement; break;  
    case integral-value5 : statement; break;  
    // ...  
    default: statement;  
}
```

```
import java.util.*;  
  
public class VokalKonsonan {  
    public static void main(String[] args) {  
        char c = (char) (Math.random() * 26 + 'a');  
        System.out.print(c + ": ");  
        switch(c) {  
            case 'a':  
            case 'e':  
            case 'i':  
            case 'o':  
            case 'u':  
                System.out.println("huruf vokal");  
                break;  
            default:  
                System.out.println("Konsonan");  
        }  
    }  
}
```

Perulangan

- `while(kondisi) { <statements> }`
- `do{ <statements> } while(kondisi);`
- `for(<init> ; <kondisi> ; <inc/dec>) { <statements> }`

- `break` dan `continue`

Inputan

- Menerima input dari user:
 - Menggunakan **java.util.Scanner**
Scanner s = new Scanner(System.in);
System.out.print("nama : ");
String nama = s.next();
System.out.println("nama anda : " + nama);
 - Menggunakan **Argumen** dari parameter String args[] dalam method main.
 - Masing-masing inputan dipisahkan menggunakan spasi.
 - Setiap input diterima sebagai String sesuai urutannya.
 - Menggunakan **BufferedReader**
String userInput = null;
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
userInput = br.readLine();
 - Menggunakan **JOptionPane**
String coba = JOptionPane.showInputDialog(null,"Inputkan
angka","Input",JOptionPane.OK_CANCEL_OPTION);

Menggunakan Argumen

```
class baca
{
    public static void main(String[] args)
    {
        int x, cacah = args.length;
        System.out.println("Parameter Anda ada : "+cacah);
        System.out.println("Yaitu :");
        for(x=0;x<cacah;x++)
        {
            System.out.print(args[x]+" , ");
        }
    }
}
```

```
C:\Javaku>java baca 1 2 5 3
Parameter Anda ada : 4
Yaitu :
1, 2, 5, 3,
```

Array pada Java

- `int[] myArray = {1,2,3};`
- `int[] myArray2 = new int[4];`
 - `myArray2[0] = 1;`
- `int[][] duaD = new int[2][2];`
 - `duaD[i][j] = 1;`
- Gunakan `length` untuk mengetahui jml elemen array
- If the value of an index is negative or greater than the array length then an `ArrayIndexOutOfBoundsException` is thrown

Matriks

Matriks
Class

Fields

- baris
- elemen
- kolom

Methods

- getBaris
- getElemen
- getKolom
- getSemuaElemen
- isBujurSangkar
- isIdentitas
- Matriks
- operasiKonstanta
- setElemen
- setIsi
- tampil
- tampilTranspose

MatriksError
Class
→ Exception

Methods

- MatriksError

TesMatriks
Class

Methods

- kali
- main
- tambah

- Buatlah kelas bernama Matriks
- Buatlah kelas implentasi bernama TesMatriks

```

Elemen [1,1] = 1
Elemen [1,2] = 2
Elemen [1,3] = 3
Elemen [2,1] = 4
Elemen [2,2] = 5
Elemen [2,3] = 6
Elemen [3,1] = 7
Elemen [3,2] = 8
Elemen [3,3] = 9
=====Tampilan M1=====
1      2      3
4      5      6
7      8      9
Bujur Sangkar? true
Identitas? false
=====Tampilan Transpose M1=====
1      2      3
4      5      6
7      8      9
=====Tampilan Setelah M1 + 3=====
4      5      6
7      8      9
10     11     12

```

```

Elemen [1,1] = 1
Elemen [1,2] = 2
Elemen [1,3] = 3
Elemen [2,1] = 4
Elemen [2,2] = 5
Elemen [2,3] = 6
Elemen [3,1] = 7
Elemen [3,2] = 8
Elemen [3,3] = 9
=====Tampilan M2=====
1      2      3
4      5      6
7      8      9

=====Tampilan M1*M2=====
66     81     96
102    126    150
138    171    204
=====Tampilan M1+M2=====
5      7      9
11     13     15
17     19     21
Press any key to continue...

```

Tugas

- Buatlah sebuah class bernama UserInput yang berguna untuk menerima inputan dari pengguna dari Console, menggunakan BufferedReader
 - Method getInt
 - Method getDouble
 - Method getString
- Kemudian gunakan untuk membuat program perhitungan luas segitiga, persegi panjang, dan lingkaran (dalam bentuk menu)