

Pemrograman Jaringan 0

anton@ukdw.ac.id

Deskripsi

- Matakuliah: Pemrograman Jaringan
- SKS: 3
- Dosen: Antonius Rachmat C, S.Kom, M.Cs
- Waktu: Jumat, 07.30
- Ruang: LAB
- Deskripsi:
 - Mempelajari konsep-konsep jaringan pada layer aplikasi dan teknik pemrogramannya menggunakan Java

Kompetensi

- 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
- Mampu mengembangkan aplikasi jaringan berbasis web dengan Java Servlet

Silabus

- Silabus + Refresh Java – 20/8
- Pengantar Jaringan 1 – 27/8
 - Jaringan Komputer & Protokol
 - IP Address, Port, Socket
 - TCP dan UDP
 - Internet
- Pengantar Jaringan 2 – 3/9
 - Client/Server Model
 - Middleware
 - Konsep dasar web
 - HTTP, URI, URL, MIME

Silabus-2

- IO dan Stream – 17/9
 - File
 - Input, Output, Filter, dan Reader
- Pemrograman HTTP – 24/9
 - Protokol HTTP
 - Metode Get dan Post
 - InetAddress, URL, URI Class
 - HTTPServer dan ProxyServer
- Pemrograman Socket – 01/10
 - Connection Oriented

Silabus-3

- Threading – 22/10
 - Multithreading, Synchronization
- Socket Multithreading, JAR dan JDBC – 29/11
 - Add, insert, delete, edit
- Pemrograman Socket – 5/11
 - Connectionless Oriented

Silabus-4

- Komunikasi Antar Obyek – 12/11
 - Obyek Serialization
- Remote Method Invocation – 19/11
 - Konsep & Aplikasi
- CORBA – 26/11
 - Konsep & IDL
 - Pemrograman CORBA
- Java Servlet – 3/12
 - Konsep dan pemrograman dasar

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

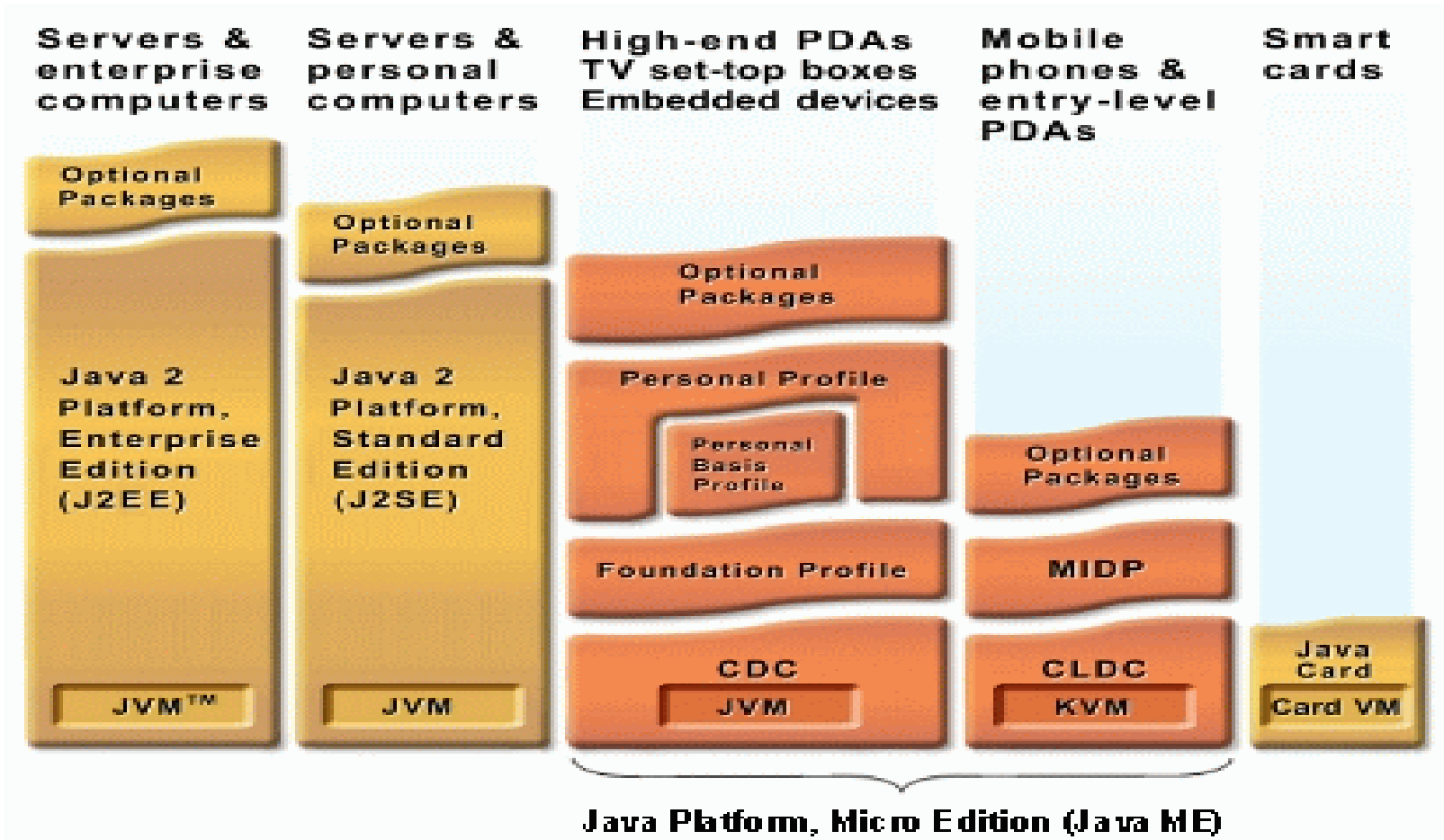
Komponen Penilaian

- TTS : 20
- TAS : 25
- Tugas Paper : 20
 - Carilah program jaringan di Internet yang **sdh jadi**, analisa, bahas source codenya, buat laporannya, kumpul saat TTS!
- Tugas Lab : 35
 - DOS, Socket, JDBC, RMI / Corba

Java

- Dibuat oleh Sun Microsystems (<http://java.sun.com>)
- Proyek awal: Green
 - Bahasa baru: OAK oleh James Gosling
 - Kemudian oleh Sun disebut Java
- Konsep Java menggunakan OOP
 - Sifatnya: Write Once Run Everywhere?
- Mendukung multiplatform language

Jenis Java



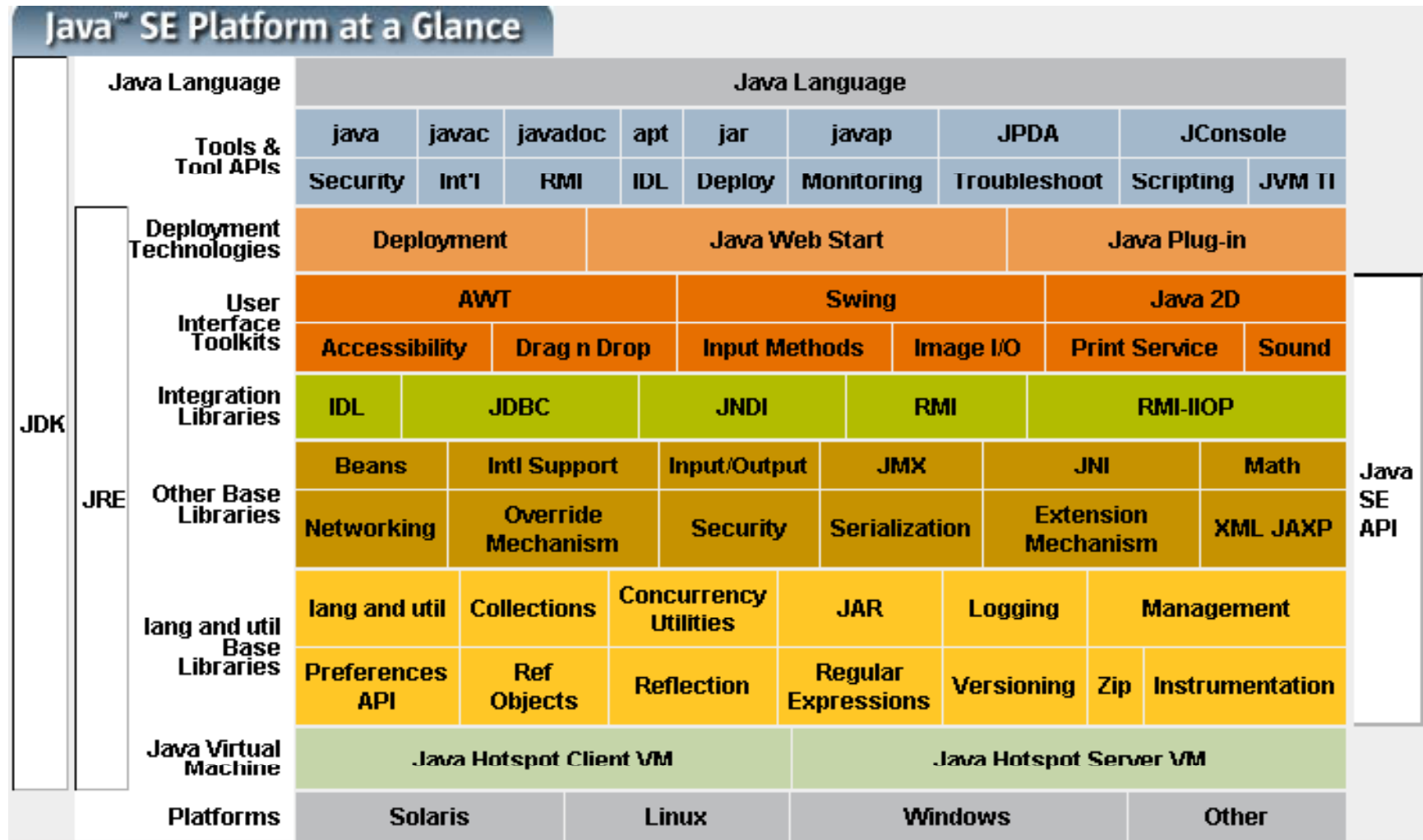
The Java programming environment

- 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
 - Just in time compilation + runtime modification of code
- Multi-threaded & concurrent programming
- Database & XML access
- Mobile application support

Program Penting pada J2SDK

- Javac -> Compiler
- Java -> Interpreter
- Jdb -> Debugger
- Javap -> Disassembler
- Appletviewer -> Penampil applet
- Javadoc -> Pengenerate documentation
- Javah -> Pengenerate header bahasa C

J2SE Platform



Tools

- JCreator

- <http://www.jcreator.com>



- Java Software Development Kit (JDK)

- <http://java.sun.com/javase/downloads/widget/jdk6.js>



- Netbeans 6.8

- <http://netbeans.org/downloads/index.html>



- Eclipse

- www.eclipse.org/downloads



- UML Editor

- www.staruml.sourceforge.net



Contoh Program Java Sederhana

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

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
```

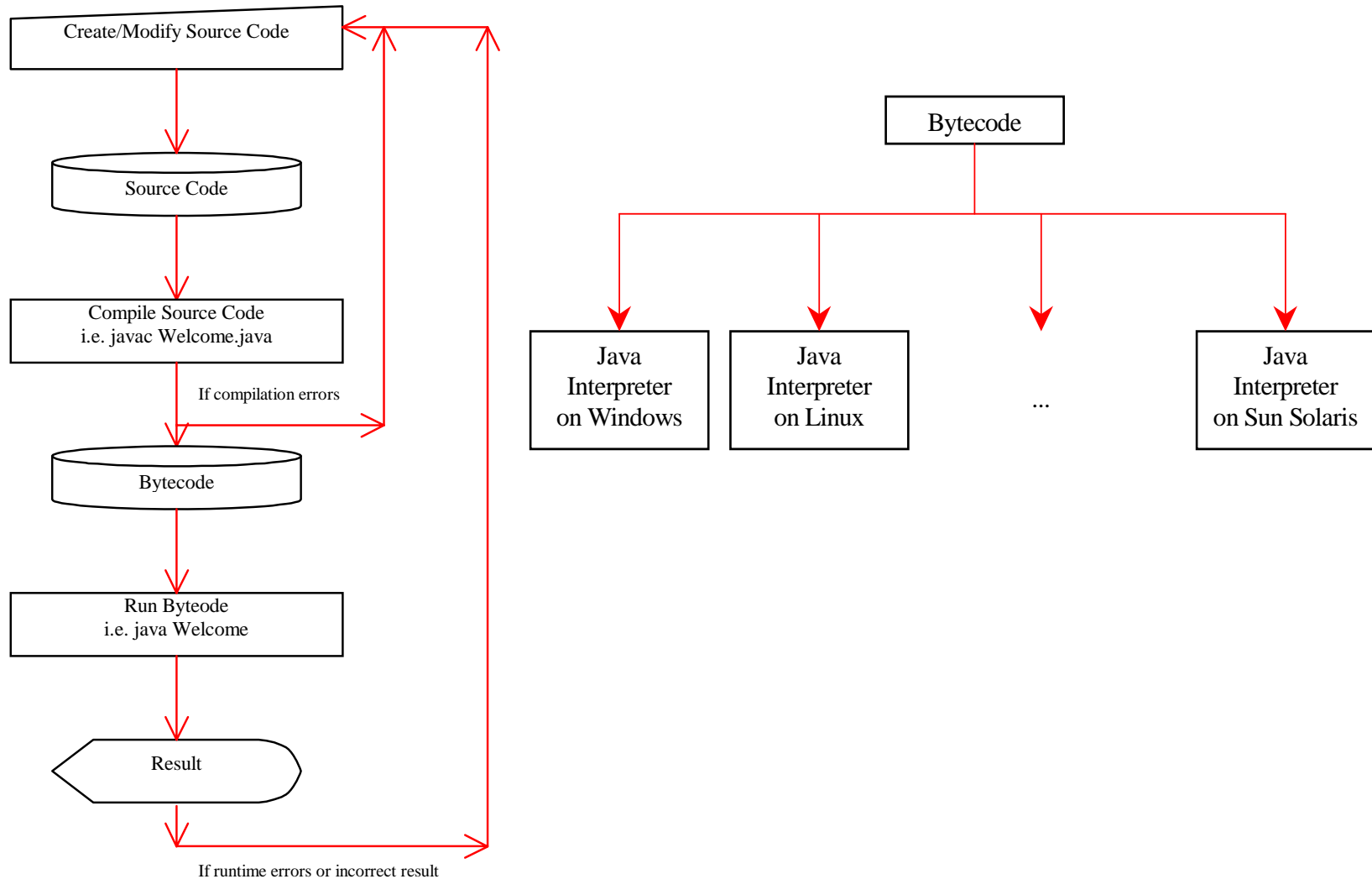
```
Output: HelloWorld.class
```

- **Run**

```
java HelloWorld
```

```
Output: Hello, World
```

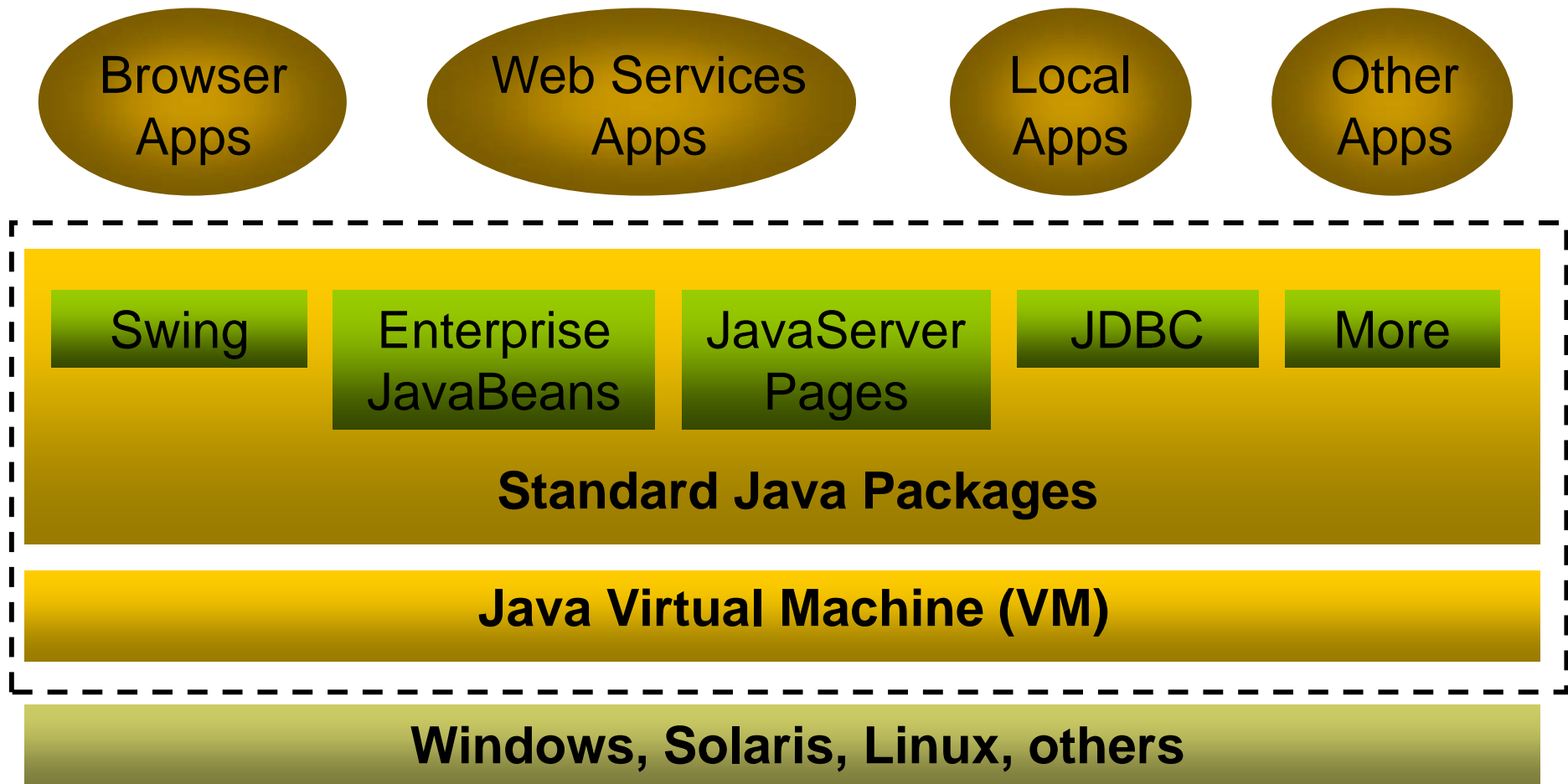
Compilation & Execution Phase



4 Aplikasi Java

- **Applications:** program standalone di komputer, dari aplikasi console sampai dengan GUI yang kompleks yang menggunakan **javax.swing**
- **Applet:** program Java yang dijalankan di web browser (client) dengan menggunakan HTML & Java
- **Servlet:** program yang melakukan generating isi webpage namun berjalan di java-enabled web server yang kemudian akan dikirimkan hasilnya ke client.
- **JSP/ JSF:** aplikasi web yang berjalan di sisi server.

The Java Environment



Instalasi JDK

- Download JDK
- Instalasi biasa
- Set PATH dan JAVA_HOME
 - set PATH=%PATH%;<your Java\Bin directory>
 - set JAVA_HOME=<your Java directory>
- Bisa juga dilakukan lewat Windows GUI
 - Control Panel > System Properties > Environment Variable

Tipe Data

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

Variabel - konstanta

```
class Coba
{
    public static void main(String[] args)
    {
        final int CONS = 12;
        System.out.println(CONS + "Hello World!");
    }
}
```

- Konstanta → variabel yang nilainya tidak bisa diubah.
- Pemberian namanya biasanya menggunakan huruf besar semua.
- Kata kuncinya menggunakan **final**.

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);
```

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, ...).

```
java.text.DecimalFormat df = new java.text.DecimalFormat("Rp  
0,00");  
df.format(300);
```
- Individual wrapper class methods, eg, `Integer.toString(i)`.
- *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 "0.3333333333333333" to s
```

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);`

Membaca data dari Keyboard

- Sejak versi 1.6x keatas:

Gunakan:

`System.console().readLine();`

- Fungsi diatas menerima inputan dari pengguna bertipe data String sehingga harus ditampung terlebih dahulu ke variabel bertipe String
- Contoh:
`String nama = System.console().readLine("Masukkan nama:");`

Contoh

```
public static void main(String[] args){  
    /*String bukan tipe data primitif, namun mengenai tipe ini  
    akan dibahas pada waktu lain, yang pasti variabel dengan  
    tipe data ini dapat menampung nilai string (lebih dari 1  
    karakter  
    */  
    String nama = "Antonius Rachmat C";  
    System.out.println("Hello "+nama);  
    /*Untuk menerima masukan dari keyboard dapat dilakukan dengan  
    cara memakai kelas System sbb.:  
    */  
    System.out.print("Who are you ? ");  
    nama = System.console().readLine();  
    System.out.println("Hello "+nama);  
}
```

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

Ciri khas OOP

- **Abstraksi** : Mendefinisikan obyek abstrak yang mampu melakukan kegiatan, mengubah state, dan berkomunikasi dengan obyek lain pada sistem
 - Membuat class yg terdiri dari atribut dan method
- **Enkapsulasi** : Menyembunyikan informasi dan detail implementasi sebuah method, serta mengatur akses terhadap atribut/method
 - Hak akses pada method
- **Polimorfisme** : Membuat obyek dari kelas dasar dapat berperilaku seperti obyek lain yang merupakan turunannya
 - Polimorfisme juga berarti banyak bentuk yg diimplementasikan pada multiple constructor class
- **Inheritance**: pewarisan atribut dan method dari class induk ke kelas anak

Java – Instantiation

**Instance
Variable Name**

```
BankAccount account = new BankAccount();
```

**Class
Name**

**Class
Constructor**

Java – Use of Instances

- Calls Methods
 - `account.deposit()`
 - `account.withdraw()`
 - `account.checkbalance()`
- Access its instance variables
 - `account.accountnumber`
 - `account.balance`
- Garbage Collection
 - Java will ***automagically*** garbage collect the object when there are no more references to it

Java – Defining a Class

```
[access][abstract/final] class className  
[extends superClassName]  
[implements interfaceNames...] {  
  
//constructors  
  
//member functions  
  
//member variables  
  
}
```

Java – Constructors

- Example (Single constructor):

```
public class BankAccount {  
    public BankAccount() {  
        ...  
    }  
}
```

Pada Java nama constructor sama dengan nama Class

Java – Overloading Constructors

- Exampe (Multiple Constructors):

```
public class BankAccount() {
```

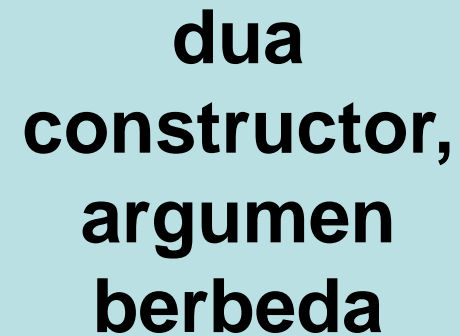
```
    public class BankAccount() {
```

```
        ...  
    }
```

```
    public class BankAccount(int initBalance) {
```

```
        ...  
    }
```

```
}
```



**dua
constructor,
argumen
berbeda**

Java – Methods

- Template:

```
[access] returnType methodName  
    ([arguments]) {  
    //method body  
  
    ...  
}
```

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));
    }
}
```

Java – Access Type

- There are 4 types of access keywords to describe which classes have access:
 - public –any other class in any package
 - protected –any subclass has access
 - (default) –only classes within the same package
 - private –only accessible from within a class
- Good for keeping **data abstraction**

Overriding

- Contoh:

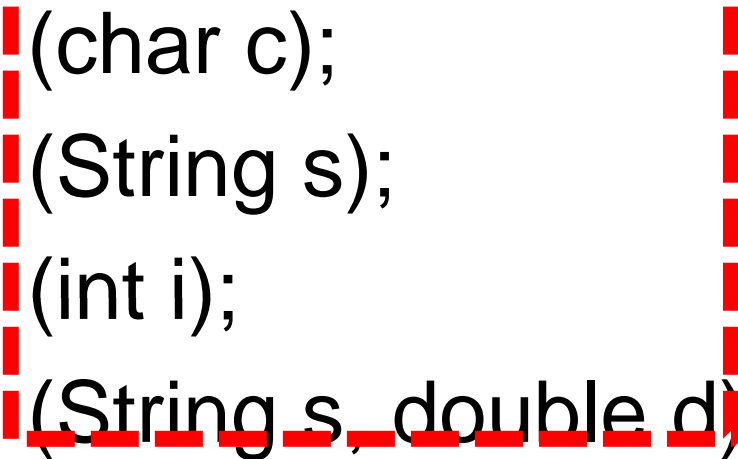
```
class Mahasiswa {  
    public void perkenalan() {  
        System.out.print("Objek mahasiswa");  
    }  
}
```

```
class MhsSI extends Mahasiswa {  
    public void perkenalan() {  
        super.perkenalan();  
        System.out.println("TI");  
    }  
}
```

Overloading Method

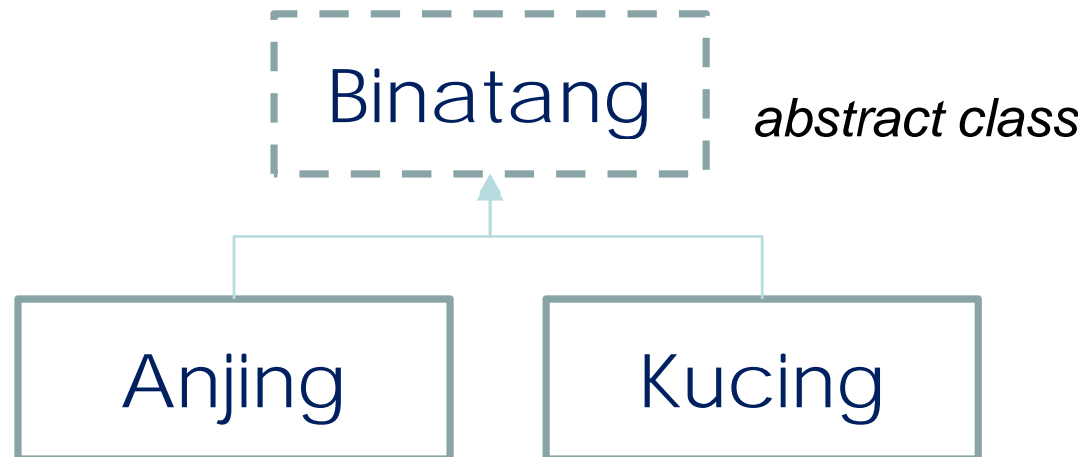
- Contoh:

```
public void println(char c);  
public void println(String s);  
public void println(int i);  
public void println(String s, double d);
```



Abstract Class

```
abstract class Binatang
{
    public abstract String bersuara();
    public void makan(int x)
    {
        System.out.println("Makan makan");
    }
}
```



Interface

- **Interface:** Kumpulan fungsi/konstanta yang tidak berisi implementasi
- Contoh:

```
public interface mp3Player {  
    public static final int STATUS;  
    List TRACKLIST; //final dan static  
    void playTrack();  
    void stopTrack();  
    void volumeUp();  
    void volumeDown();  
}
```

Polymorfisme

```
class Mortal{
    public void showMortal(){ System.out.println("Mortal"); }
    public void killed(){ }
}

class Tree extends Mortal{
    public void showTree() { System.out.println("Tree"); }
    public void killed() { System.out.println("Tree killed"); }
}

class Animal extends Mortal{
    public void showAnimal() { System.out.println("Animal"); }
    public void killed() { System.out.println("Animal killed"); }
}
```

```
E:\Documents\Dosen\pbo\latihan>java Utama
Tree
Animal
Animal killed
Tree killed
```

```
public class Utama{
    public void kill(Mortal m){
        m.killed();
    }
    public static void main(String args[]){
        Tree t = new Tree();
        t.showTree();
        Animal a = new Animal();
        a.showAnimal();
        Utama u = new Utama();

        u.kill(a);
        u.kill(t);

    }
}
```

Penggunaan Java Exception

- Terdapat 5 keywords:
try, catch, finally,
throw, throws

```
try {  
    ...  
} catch (Exception e) {  
    ...  
}  
finally {  
    ...  
}
```

- Dengan **try-catch**

```
try {  
    ...  
} catch (Exception e){  
    ...  
}
```

Penggunaan Java Exception

```
public class Example05 {  
  
    public static void main(String[] args) {  
        try {  
            demoproc();  
        } catch (NullPointerException e) {  
            System.out.println("Exception ditangkap di method main: " + e);  
        }  
    }  
  
    static void demoproc() {  
        try {  
            throw new NullPointerException("demo");  
        } catch (NullPointerException e) {  
            System.out.println("Exception ditangkap di method demoproc");  
            throw e; // rethrow the exception  
        }  
    }  
}
```

Tugas “Matriks”



- 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...
```

NEXT