Pertemuan ke 7

JAVA

Language Features

Simple

- Sintaks berdasar pada C++ (transisi lebih mudah bagi programmer)
- Menghilangkan feature yang jarang dipakai
  contoh : explicit pointers, operator overloading, automatic coercions
- Penambahan memory management (mengacu pada count/garbage collection hybrid)

Object-oriented

- Fasiltas OOP mirip dengan C++
- OOP murni – seluruhnya adalah class, tidak ada independent functions

Robust

- ketiadaan pointers dan memori manajemen menghindari banyak error
- *libraries of useful, tested classes increases level of abstraction
  arrays & strings are ADTs, well-defined interfaces

Portable

- byte kode akan run pada versi manapun dari Java Virtual Machine (JVM)

Language Features (cont.)

- Platform independence
  - Dapat menjalankan Java code pada multiple platforms
  - Netralitas dicapai dengan mencampur compilation & interpretation

    Java programs diterjemahkan ke dalam *byte code* oleh Java compiler
    - byte code is a generic machine code

    byte code kemudia dieksekusi oleh interpreter (Java Virtual Machine)
    - must have a byte code interpreter for each hardware platform
  - Sebuah Applet adalah sebuah special form dari Java application

    byte code di download pada page, JVM di-embedded pada browser

- Architecture-neutral
  - Tidak ada implementation bergantung pada features (contoh : size dari primitive types)

- High-performance
  - Lebih cepat daripada traditional interpretation karena byte code mendekati native code
  - Masih sedikit lebih lambat daripada compiled language (contoh : C++)

Language Features (cont.)

Distributed

- Extensive libraries untuk mengatasi TCP/IP protokol seperti HTTP & FTP
- Aplikasi Java dpt mengakses remote URL'S sama seperti halnya mengakses file lokal

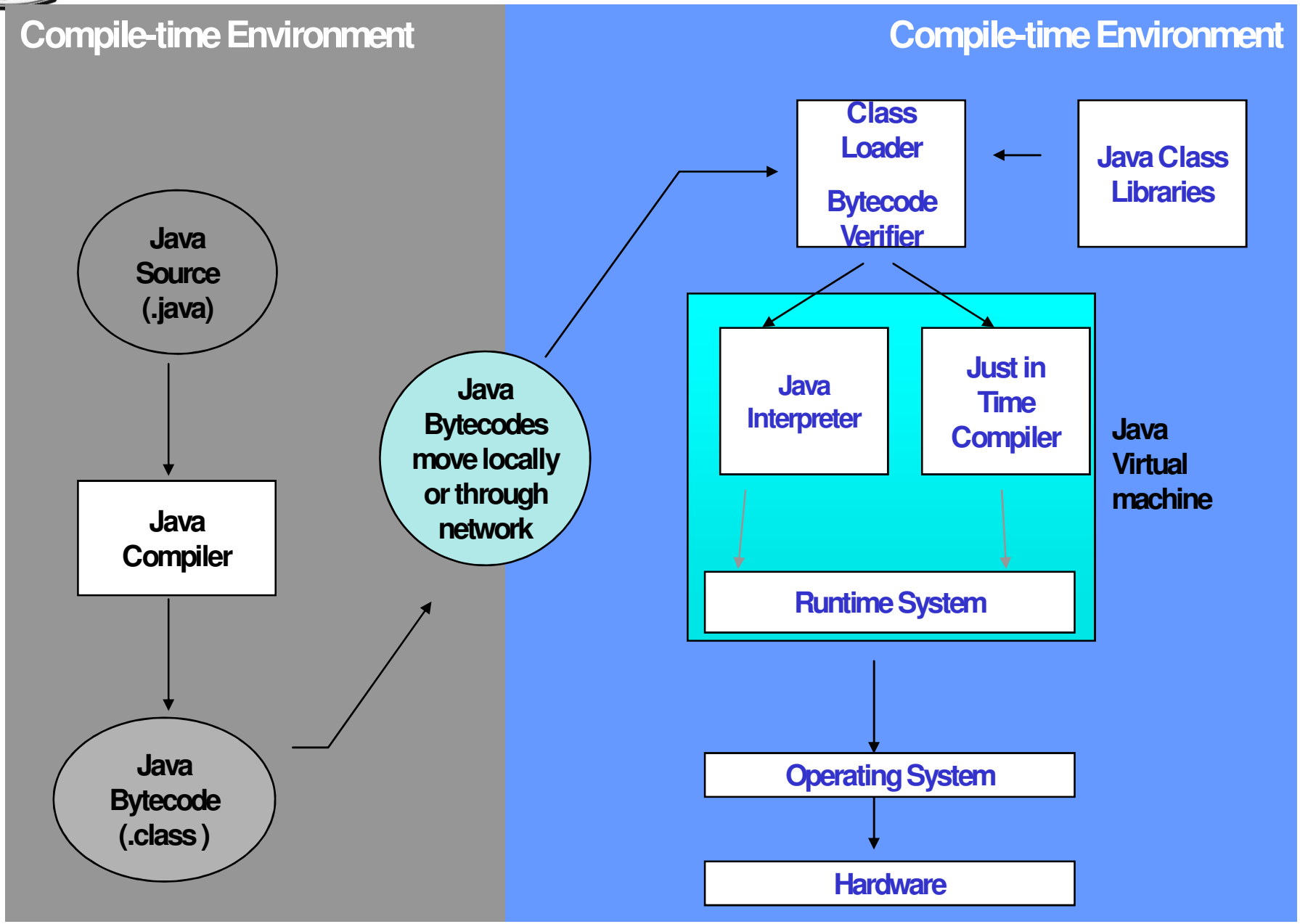
Multi-threaded

- Sebuah *thread* seperti sebuah program terpisah, dijalankan secara berbarengan
- Dapat menulis program Java dimana beberapa pekerjaan dpt dilakukan sekaligus dgn mendefinisikan multiple threads (same shared memory, but semi-independent execution)
- Threads penting untuk multi-media, web applications

Secure

- Aplikasi Java applications tidak dapat langsung mengakses ke lokasi memory
  - Akses memory adalah virtual, dipetakan oleh JVM ke lokasi fisik
  - Downloaded appletstidak dapat membuka, membaca atau menyalin local files
- JVM memeriksa autentifikasi dari class juga memeriksa autentifikasi dari class yg di-load
- *Sun meng-klaim: Model execution memungkinkan virus-free*, tamper-free* systems*

Bagaimana Java Bekerja...!



Bagaimana Java Bekerja . . .! (lanj)

Java independent hanya untuk satu alasan :

- Hanya bergantung pada Java Virtual Machine (JVM),
- Code dikompiled ke *bytecode*, yang di-interpreted oleh resident JVM,
- JIT (just in time) compilers mencoba untuk meningkatkan kecepatan.

Keamanan - Java

- Pointer denial – mengurangi kesempatan virulent programs merusak host

- Applets lebih terbatas lagi -
  - Tidak bisa
    Menjalankan local executables,
    Read atau write ke local file system,
    Berkomunikasi dengan beberapa server lain selain dengan originating server.

Object-Oriented

- Java mendukung OOD
  - Polymorphism
  - Inheritance
  - Encapsulation

- Program Java berisi tak lain hanya definisi dan instantiation dari class
  - Semuanya di-encapsulate dalam sebuah class!

Keuntungan Java

- Portable - Write Once, Run Anywhere
- Keamanannya sudah dipikirkan secara matang
- Memory management sempurna
- Didesain untuk network programming
- Multi-threaded (berbagai tugas simultaneous)
- Dynamic & extensible (loads of libraries)
  - Class disimpan pada file yang terpisah
  - Loaded hanya jika dibutuhkan

Java Programming Models

- Java *applications* are stand-alone programs
  - Harus dikompile menjadi Java byte code dengan Java compiler
  - Dieksekusi oleh sebuah interpreter (Java Virtual Machine)
- Java *applets* provide for client-side programming
  - dikompile menjadi Java byte code, kemudian di-*downloaded sebagai bagian dari sebuah Web page*
  - Dieksekusi oleh JVM *embedded dalam Web browser*
  - Tidak seperti JavaScript, Java full-featured dengan extensive library support
  - Java dan APIs-nya telah menjadi standard industri
    - Pendefinisian language dikontrol oleh Sun, untuk meyakinkan compatibility
    - Applications Programming Interfaces standardize the behavior of useful classes and libraries of routines*
- Java *servlets* provide similar capabilities on the server-side
  - Merupakan alternative dari CGI programs, lebih terintegrated dengan Web server

Java Applets

- Important point : Java applets & applications look different!
  - Jika ingin mendefinisikan stand-alone application, buat sebuah aplikasi

    membutuhkan `public static void main` function, sama dengan C++ main
  - Jika ingin meng-embed code pada sebuah Web page, buat sebuah applet

    membutuhkan `public void paint`, `public void init`, ...

- As with JavaScript, security is central
  - Ketika sebuah Java applet di-downloaded, pemeriksa bytecode dari JVM memeriksa agar dapat melihat pada saat bytecode berisi byte yang terbuka, terbaca dan tertulis dalam lokal disk.
  - Java applet dapat membuka sebuah window baru dengan Java logo untuk pencegahan, yaitu dengan menyembunyikan system windows (contohnya: pencurian passwords)
  - Java applet tidak membolehkan untuk terhubung ke server lain kecuali ke host.
  - Kondisi yang aman/secure ini disebut *sand box model*

First Java Applet

```
import java.awt.*;
import java.applet.*;

/**
 * This class displays "Hello world!" on the applet window. */
public class HelloWorld extends Applet
{  public void paint(Graphics g)
  {g.drawString("Hello world!", 10, 10); // writes starting 10 pixels over & down
  }
}
```

libraries: Java menyediakan provides extensive library support dalam bentuk class

- Library dipanggil menggunakan import (mirip dengan `#include` di C++)

  `java.awt` : berisi contains Abstract Window Toolkit (untuk GUI classes & routines)

  `java.applet` : berisi definisi applet class

Comments : `//` dan `/* */` berfungsi sama seperti pada C++

`/** */` menandakan komentar

First Java Applet

```
import java.awt.*;
import java.applet.*;
/**
 * This class displays "Hello world!" on the applet window. */
public class HelloWorld extends Applet
{  public void paint(Graphics g)
  { g.drawString("Hello world!", 10, 10); // writes starting 10 pixels over & down
  }
}
```

Pendefinisian class di Java

- Sama dengan pada C++ (tetapi tidak ada titik koma di akhir)
  - Dapat berisi instance variables (data fields) & methods(member functions)
  - Didahului dengan pendefinisian class & method dengan *public* untuk membuatnya tersedia bagi semua program
- Tidak ada fungsi stand-alone di Java*
- Harus disimpan pada sebuah file dengan nama yang sama dengan ekstension .java
  - Contoh : `HelloWorld.java`

First Java Applet

```
import java.awt.*;
import java.applet.*;
/**
 * This class displays "Hello world!" on the applet window. */
public class HelloWorld extends Applet
{  public void paint(Graphics g)
  { g.drawString("Hello world!", 10, 10);  // writes starting 10 pixels over & down
  }
}
```

Seluruh applets mewarisi dari Applet class (pada java.applet)

default methods termasuk :

- `init()` : memanggil saat page di-load untuk membuat/inisialisasi variables

  by default, does nothing

- `paint(Graphics g)` : called to draw (after init) or redraw (after being obscured)

  here, the paint method is overridden to display text on the applet window

Embedding Applet di HTML

to include an applet in a Web page, use either

- APPLET tag (deprecated)

  CODE menentukan applet name, HEIGHT dan WIDTH menentukan window size

  text antara APPLET tags ditampilkan jika tidak dapat dieksekusi (e.g., Java not enabled)

- OBJECT tag

  Lebih dipilih HTML 4, tetapi tidak secara universal mendukung

```
<html>
<!- COMP519    hello1.html    18.09.2005 -->
<head>  <title>Hello World Page</title> </head>
<body>
<p>
  <applet code="HelloWorld.class" height=100 width=100>
    You must use a Java-enabled browser to view this applet.
  </applet>
</p>
</body> </html>
```

HTML & Applets

```
<html>
<!- COMP519    hello2.html    18.09.2005 -->
<head>
  <title>Hello World Page</title>
</head>
<body>

<p>
<div align="center">
<table border=1>
<tr><td>

<applet code="HelloWorld.class" height=200 width=200>
  You must use a Java-enabled browser to view this applet.
</applet>
</td></tr>
</table>
</div>
</p>

</body>
</html>
```

Sebuah applet dapat di-embedded dalam HTML elements seperti element lainnya

Berguna untuk format dan layout

Parameters di HTML

```
<html>
<!- COMP519    hello3.html      18.09.2005 -->
<head>
<title>Hello World Page</title>
</head>
<body>
<p>
<div align="center">
<table border=1>
<tr><td>
<applet code="HelloWorld1.class" height=35 width=300>
  <param name="name" value="Chris">
  <param name="age" value=20>
  You must use a Java-enabled browser to view this applet.
</applet>

</td></tr>
</table>
</div>
</p>

</body> </html>
```

Dapat menentukan parameter APPLET ketika di-embedded di HTML

- setiap parameter harus mempunyai PARAM tag sendiri dalam APPLET element
- Menentukan parameter name dan value

Applet Parameters

```
import java.awt.*;
import java.applet.*;

/**
 * This class displays a message based on parameters. */
public class HelloWorld1 extends Applet
{    public void paint(Graphics g)
    {   String userName = getParameter("name");
        int userAge = Integer.parseInt(getParameter("age"));

        String message1 = "Hello " + userName + ".";
         String message2 = "On your next birthday, you will be " +
                          (userAge+1) + " years old.";

        g.drawString(message1, 10, 10);
        g.drawString(message2, 10, 30);
    }
}
```

can access parameters passed in from the HTML document

`getParameter` mengakses nilai dari parameter (must know its name)

- Jika parameter ditunjukkan angka, harus `parseInt` atau `parseFloat`

Java Constructs

- Akan dikenali oleh C/C++ programmers
- Tipe nama yang indentik; Tipe yang dijamin untuk diartikan secara benar dan teliti
- Source dalam .java files, compiled code dalam .class files; downloaded (biasanya) dalam .jar files (Java archive)

Java Development

- Langkah-langkah Pengembangan
  - Buat source files di editor
  - Kompile menggunakan *javac*
  - Jalankan/test menggunakan *java*

Primitive Types dan Variables

- Boolean, char, byte, short, int, long, float, double dsb.
- Tipe dasar ini adalah satu-satunya tipe yang bukan objects
- Ini berarti bahwa kita tidak menggunakan operator baru untuk membuat primitive variable.
- Pendeklarasian primitive variables:

```
float initVal;
int retVal, index = 2;
double gamma = 1.2, brightness
boolean valueOk = false;
```

Initialisation

- Jika tidak ada nilai di berikan sebelumnya untuk digunakan, compiler akan memberikan kesalahan
- Java men-set primitive variables menjadi zero atau false pada kasus dari sebuah boolean variable
- Seluruh object references awalnya di-set null
- Sebuah array adalah sebuah object
  - Set null pada deklarasi
  - Elements to zero false or null on creation

Declarations

```
int index = 1.2;                                    // compiler error
boolean retOk = 1;                                  // compiler error
double fiveFourths = 5 / 4;                         // no error!
float ratio = 5.8f;                                 // correct
double fiveFourths = 5.0 / 4.0;         // correct
```

- 1.2f adalah float value akurasi sampai 7 decimal places.
- 1.2 adalah double value akurasi sampai 15 decimal places.

Assignment

- All Java assignments are right associative

```
int a = 1, b = 2, c = 5
a = b = c
System.out.print(
"a= " + a + "b= " + b + "c= " + c)
```

- What is the value of a, b & c
- Done right to left: `a = (b = c);`

Basic Mathematical Operators

- * / % + – are the mathematical operators
- * / % have a higher precedence than + or –

```
double myVal = a + b % d – c * d / b;
```

- Is the same as:

```
double myVal = (a + (b % d)) – ((c * d) / b);
```

Statements & Blocks

- Sebuah statement sederhana adalah pernyataan yang diakhiri titik koma A :

  name = “Fred”;

- Sebuah block adalaj gabungan pernyataan yang ditutup dalam tanda kurung kurawal :

  {

  name1 = “Fred”; name2 = “Bill”;

  }

- Blocks dapat terdiri dari beberapa blocks

Flow of Control

- Java menjalankan satu statement berurutan sesuai urutan penulisan
- Beberapa statement Java adalah statement flow of control :

Alternation : if, if else, switch

Looping : for, while, do while

Escapes : break, continue, return

If – The Conditional Statement

- Statement if meng-evaluasi sebuah expression & jika evaluasi benar maka tindakan tertentu akan dijalankan. Misal jika nilai dari x lebih kecil dari 10, maka x sama dengan 10

```
if ( x < 10 ) x = 10;
```

- Hal ini dapat ditulis seperti :

```
if ( x < 10 )
x = 10;
```

- Atau, alternative lain :

```
if ( x < 10 ) { x = 10; }
```

Relational Operators

==	Equal (careful)	!=	Not equal
>=	Greater than or equal	<=	Less than or equal
>	Greater than	<	Less than

If… else

- The if … else statement evaluates an expression and performs one action if that evaluation is true or a different action if it is false.

```
if (x != oldx) {
   System.out.print("x was changed");
 }
 else {
   System.out.print("x is unchanged");
 }
```

Nested if … else

```
if ( myVal > 100 ) {
   if ( remainderOn == true) {
      myVal = mVal % 100;
   }
   else {
    myVal = myVal / 100.0;
   }
}
else
{
   System.out.print("myVal is in range");
}
```

else if

- Berguna untuk memilih beberapa alternative :

```
if ( n == 1 ) {
   // execute code block #1
}
else if ( j == 2 ) {
   // execute code block #2
}
else {
   // if all previous tests have failed, execute code block
   #3
}
```

Peringatan...

WRONG!

```
if( i == j )
      if ( j == k )
        System.out.print("i equals k");
      else
        System.out.print("i is not equal to j");
```

CORRECT!

```
if( i == j ) {
   if ( j == k )
   System.out.print("i equals k");
}
else
   System.out.print("i is not equal to j");  // Correct!
```

Switch Statement

```
switch ( n ) {
   case 1:
     // execute code block #1
     break;
   case 2:
     // execute code block #2
     break;
     default:
     // if all previous tests fail then                    //execute
   code block #4
     break;
}
```

for loop

- Loop n times

```
for ( i = 0; i < n; n++ )
{
   // this code body will execute n times
   // ifrom  0 to n-1
}
```

- Nested for:

```
for ( j = 0; j < 10; j++ ) {
   for ( i = 0; i < 20; i++ ){
     // this code body will execute 200 times
   }
}
```

while loops

```
while(response == 1) {
   System.out.print( "ID =" + userID[n]);
   n++;
   response = readInt( "Enter ");
}
```

do {…} while loops

```
do {
   System.out.print( "ID =" + userID[n] );
   n++;
   response = readInt( "Enter " );
    } while (response == 1);
```

Berapa kali paling sedikit loop di-executed?

Berapa kali paling banyak loop di-executed?

Break

- Break statement menyebabkan keluar dari innermost yg berisi while, do, for or switch statement.

```
for ( int i = 0; i < maxID, i++ ) {
   if ( userID[i] == targetID ) {
     index = i;
     break;
   }
} // program jumps here after break
```

Continue

- Hanya dapat digunakan dengan while, do atau for.
- Continue statement menyebabkan innermost loop mulai perulangan berikutnya

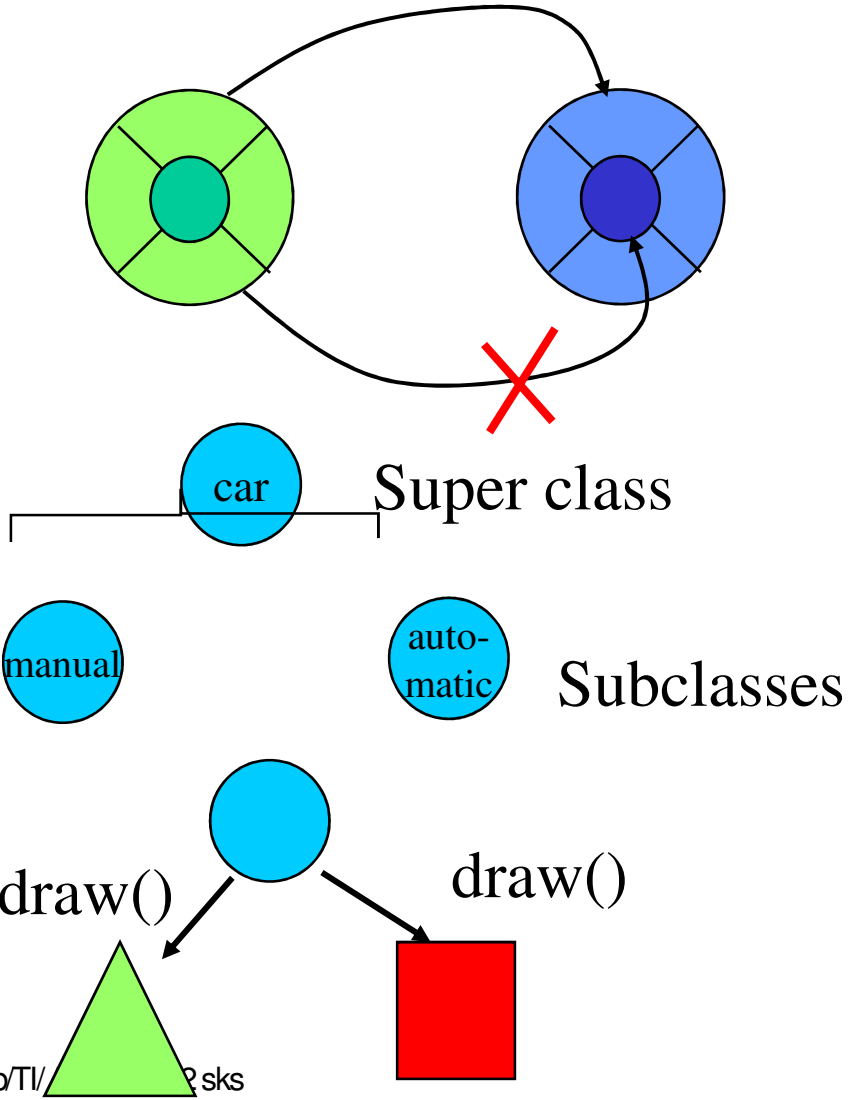
```
for ( int i = 0; i < maxID; i++ ) {
   if ( userID[i] != -1 ) continue;
   System.out.print( "UserID " + i + " :" + userID);
}
```

Classes ARE Object Definitions

- OOP - object oriented programming
- code dibangun dari object-object
- Java seperti ini disebut ***classes***
- Setiap pendefinisian class di-codekan dalam file .java yang berbeda
- Nama dari setiap object harus sama dengan nama class/object

Tiga Prinsip OOP

- Encapsulation
  - Object menyembunyikan fungsi mereka (**methods**) dan data (**instance variables**)
- Inheritance
  - Setiap **subclass** mewarisi seluruh variable dari **superclass-nya**
- Polymorphism
  - Interface sama menghubungkan tipe data berbeda



Class dan Method Sederhana

```
Class Fruit {
        int grams;
        int cals_per_gram;

        int total_calories() {
                return(grams*cals_per_gram);
        }
}
```

Methods

- Suatu method adalah suatu urutan code yang diberi nama yang dapat dilibatkan oleh Java code lain
- Suatu metoda mengambil beberapa parameter, melaksanakan beberapa perhitungan dan kemudian secara bebas mengembalikan nilai (atau obyek).
- Methods dapat digunakan sebagai bagian dari statement expression.

```
public float convertCelsius(float tempC) {
      return( ((tempC * 9.0f) / 5.0f) + 32.0 );
 }
```

Method Signatures

- Sebuah method signature menentukan :
  - Nama dari method.
  - Type dan nama dari setiap parameter.
  - Type dari value (atau object) yang dikembalikan oleh method.
  - Bermacam-macam method modifiers.
  - *modifiers type name (parameter list) [throws exceptions]*

```
public float convertCelsius (float tCelsius ) {}
public boolean setUserInfo ( int i, int j, String name ) throws IndexOutOfBoundsException {}
```

Public/private

- Methods/data dapat dideklasikan ***public*** atau ***private*** yang artinya method/data tersebut dapat atau tidak dapat diakses oleh code pada class lain …
- Good practice:
  - keep data private
  - keep most methods private
- Interface yang didefinisikan dengan baik antar class – menolong menghilangkan error

Using objects

- Code pada sebuah class akan membuat sebuah instance dari class lain …

```
Fruit plum=new Fruit();
int cals;
cals = plum.total_calories();
```

- ***Dot operator*** membolehkan kita untuk mengakses (public) data/methods dalam Fruit class

Constructors

- The line

  plum = new Fruit();

- invokes sebuah constructor method dimana kita dapat men-set initial data dari sebuah object
- Kita dapat memilih beberapa type yg berbeda dari constructor dgn argument lists yg berbeda

  eg Fruit(), Fruit(a) ...

Overloading

- Dapat memiliki beberapa versi dari sebuah method dalam class dengan tipe/jumlah arguments yang berbeda

  ```
  Fruit() {grams=50;}
  Fruit(a,b) { grams=a; cals_per_gram=b;}
  ```

- Dengan memperhatikan pada argument, Java memutuskan versi mana yang digunakan

Java Development Kit

- javac - The Java Compiler
- java - The Java Interpreter
- jdb - The Java Debugger
- appletviewer - Tool to run the applets

- javap - to print the Java bytecodes
- javaprof - Java profiler
- javadoc - documentation generator
- javah - creates C header files

Java vs. C++

Sintaks Java meminjam dari C++ (dan C)

- primitive types : sama seperti C++, tetapi sizes

  byte (8 bits) char (16 bits) short (16 bits) int (32 bits)

  long (64 bits) float (32 bits) double (64 bits) boolean
- variables, assignments, arithmetic & relational operators : sama seperti C++
- control structures : sama seperti C++, kecuali goto
- Functions : mirip dengan C++, tetapi harus class & harus ditentukan public/private

in Java, every variable & method belongs to a class

- Seperti di C++, dengan default setiap object mempunyai salinan data fields sendiri sehingga dikenal sebagai *instance variables*
- Seperti di C++, sebuah variables dideklarasikan static bersama dengan seluruh class objects sehingga dikenal sebagai *class variables*
- Hal yang sama, terjadi pada sebuah static method (*class method*)

  Hanya dapat dioperasikan pada class variables, diakses dari class itu sendiri

```
class Math
{  public static final double PI = 3.14159;                    // access as Math.PI
   public static double sqrt(double num) { . . . }    // access as in Math.sqrt(9.0)
  . . . }
```

Primitive vs. Reference Types

primitive types are handled exactly as in C++

- space untuk sebuah primitive object secara implisit dialokasikan
  → variable mengacu pada actual data (disimpan pada stack)

reference types (classes) are handled differently

- space untuk sebuah reference object secara eksplisit dialokasikan menggunakan `new`
  → variable mengacu pada sebuh pointer ke data (dimana disimpan pada heap)

Note: tidak seperti C++, programmer tidak bertanggung jawab untuk menghapus dynamic objects
JVM melaksanakan automatic garbage collection untuk mereklamasi memory yang tidak digunakan

Java only provides by-value parameter passing

- Tetapi mengacu pada object yang diimplementasikan sebagai pointers ke dynamic memory
- Menghasilkan behavior mimics by-reference

```
public void Init(int[] nums)
{    for (int i = 0; i < nums.length; i++)  {
     nums[i] = 0;                                    }
}
```

```
int nums[] = new int[10];
Init(nums);
```

Java Libraries

- **String class** (secara otomatis di-load dari `java.lang`)

```
int length()
char charAt(index)
int indexOf(substring)
String substring(start, end)
String toUpperCase()
boolean equals(Object)
...
```

```
String str = "foo"

String str = new String("foo");
```

- **Array class** (secara otomatis di-load dari `java.lang`)

```
int length          instance variable
Type [](index)      operator
String toString()
...
```

```
int[] nums = {1,2,3,4,5};

int[] nums = new int[10];
```

- **Java provides extensive libraries of data structures & algorithms**

```
java.util  →   Date          Random        Timer
               ArrayList     LinkedList    Stack
               TreeSet HashSet
               TreeMap HashMap
```

Applet Behavior

- recall
  - init method dipanggil saat applet diload pertama
  - Berguna untuk initializing variables & objects
  - paint method dipanggil segera setelah init, dan kapanpun applet perlu menggambar (contoh : setelah window resized)

- when paint is called, it is given the default Graphics object
  - Graphics methods termasuk :

```
void drawString(String msg, int x, int y)

void setColor(Color color)
```

`Color` class is predefined, constants include:

```
Color.red, Color.blue, Color.black, ...
```

Hello Again

```
import java.awt.*;
import java.applet.*;
import java.util.Random;

/**
 * This class displays lots of "Hello world!"s on the applet window.
 */
public class HelloWorld2 extends Applet
{
    private static final int NUM_WORDS=100;
   private static final Color[] colors =
    {Color.black,Color.red,Color.blue,Color.green,
    Color.yellow};
    private static Random randy;

    private int RandomInRange(int low, int high)
    {
          return (Math.abs(randy.nextInt()) %
        (high-low+1)) + low;
    }

    public void init()
    {
          randy = new Random();
    }

    public void paint(Graphics g)
    {
          for (int i = 0; i < NUM_WORDS; i++) {
                int x = RandomInRange(1, 140);
                int y = RandomInRange(10, 200);
                g.setColor(colors[RandomInRange(0,4)]);
                g.drawString("Hello world!", x, y);
          }
     }
}
```

store colors in an array

- pick random index and change color using `setColor`

Random class provides methods for generating random values

override init method to allocate & initialize (similar to a constructor)

```
<applet code="HelloWorld2.class" height=200 width=200>
You must use a Java-enabled browser to view this applet.
</applet>
```

view page

Parameters & Applet Dimensions

recall:

- Dapat menentukan parameters di HTML document menggunakan `<PARAM>` tags
- Akses nilai parameter values (berdasar pada nama) menggunakan `getParameter` method

can also access the dimensions of an applet using a Dimension object

```
Dimension dim = getSize();     // stores applet dimensions
```

dapat mengakses applet height melalui `dim.height`

dapat mengakses applet width melalui `dim.width`

Adaptive Hello

```
import java.awt.*;
import java.applet.*;
import java.util.Random;

/**
 * This class displays lots of "Hello world!"s on the applet window.
 */
public class HelloWorld3 extends Applet
{
    private static final Color[] colors =
            {Color.black,Color.red,Color.blue,Color.green,Color.yellow};
    private static Random randy;
    private Dimension dim;
    private int numReps;

    private int RandomInRange(int low, int high)
    {
            return (Math.abs(randy.nextInt()) % (high-low+1)) + low;
    }

    public void init()
    {
            randy = new Random();
            dim = getSize();
            numReps = Integer.parseInt(getParameter("reps"));
    }

   public void paint(Graphics g)
   {
       for (int i = 0; i < numReps; i++) {
      int x = RandomInRange(1, dim.width-60);
      int y = RandomInRange(10, dim.height);
       g.setColor(colors[RandomInRange(0,4)]);
      g.drawString("Hello world!", x, y);
            }
    }
}
```

`getParameter` accesses the values of the parameters

here, specify number of reps in Web page

uses `getSize` to get dimensions, pick random coords for text within the applet

```
<applet code="HelloWorld3.class" height=300 width=400>
  <param name="reps" value=200>
  You must use a Java-enabled browser to view this applet.
</applet>
```

view page

Applet Graphics

- in addition to displaying text
  - Dapat juga menggambar memperhitungkan Object Graphics

```
void drawLine(int x1, int y1, int x2, int y2)

void drawRect(int x, int y, int width, int height)
void fillRect(int x, int y, int width, int height)

void drawOval(int x, int y, int width, int height)
void fillOval(int x, int y, int width, int height)
```

- EXAMPLE: draw a red circle inscribed in a square, then draw random dots (dart pricks)
  - Dengan menghitung banyaknya jumlah titik di dalam vs. di luar lingkaran, dapat memperkirakan nilai dari π

    π = 4 * (area of circle/area of square)

Graphical Applet

```
public class Monte1 extends Applet
{
    private static Random randy;
    private int NUM_POINTS;
    private int SIZE;

    private int RandomInRange(int low, int high) { CODE OMITTED }
    private double distance(int x1, int y1, int x2, int y2) { CODE OMITTED }

    public void init()
    {
        randy = new Random();
        NUM_POINTS = Integer.parseInt(getParameter("points"));
        Dimension dim = getSize();
        SIZE = Math.min(dim.width, dim.height);
    }

    public void paint(Graphics g)
    {
        g.setColor(Color.red);
        g.fillOval(0, 0, SIZE, SIZE);
        for (int i = 0; i < NUM_POINTS; i++) {
            int x = RandomInRange(0, SIZE);
            int y = RandomInRange(0, SIZE);
            if (distance(x, y, SIZE/2, SIZE/2) < SIZE/2) {
                g.setColor(Color.white);
            }
            else {
                g.setColor(Color.black);
            }
            g.drawLine(x, y, x, y);
        }
    }
}
```

`init` method creates random number generator & gets parameters

`paint` method draws a circle and a bunch of random points

```
<applet code="Monte1.class" height=300 width=300>
  <param name="points" value=20000>
  You must use a Java-enabled browser...
</applet>
```


view page

Double Buffering

- note: paint is called every time the page is brought to the front
  - Pada versi saat ini dari Monte, hal ini berarti titik-titik baru digambar setiap saat setiap kali page tak dikenali dan kemudian membawanya dari belakang ke depan

    → *Membuang waktu menggambar ulang*

    → *Titik-titik berbeda setiap kali applet digambar*

- the double buffering approach works by keeping an off-screen image
  - Pada init method (yang dipanggil saat page di-load):

    draw the figures on a separate, off-screen Graphics object
  - Pada paint method (yang dipanggil kapanpun page di tampilkan ke depan):

    simply display the off-screen image on the screen

Buffered Applet

```
public class Monte2 extends Applet
{
    . . .
    private Image offScreenImage;
    private Graphics offScreenGraphics;
    . . .
    public void init()
    {
        randy = new Random();
        NUM_POINTS = Integer.parseInt(getParameter("points"));
        Dimension dim = getSize();
        SIZE = Math.min(dim.width, dim.height);

        offScreenImage = createImage(SIZE, SIZE);
        offScreenGraphics = offScreenImage.getGraphics();

        offScreenGraphics.setColor(Color.red);
        offScreenGraphics.fillOval(0, 0, SIZE, SIZE);
        for (int i = 0; i < NUM_POINTS; i++) {
            int x = RandomInRange(0, SIZE);
            int y = RandomInRange(0, SIZE);
            if (distance(x, y, SIZE/2, SIZE/2) < SIZE/2) {
                offScreenGraphics.setColor(Color.white);
            }
            else {
                offScreenGraphics.setColor(Color.black);
            }
            offScreenGraphics.drawLine(x, y, x, y);
        }
    }

    public void paint(Graphics g)
    {
        g.drawImage(offScreenImage, 0, 0, null);
    }
}
```

`init` method is called when page is loaded

does drawing to a separate, off-screen Graphics object

`paint` is called after `init` and whenever the applet is revisited

Note: don't see image in progress

```
<applet code="Monte2.class" height=300 width=300>
  <param name="points" value=20000>
  You must use a Java-enabled browser...
</applet>
```

view page

```
public class Monte3 extends Applet
{
    . . .

  public void init() {
     randy = new Random();
     NUM_POINTS = Integer.parseInt(getParameter("points"));
     Dimension dim = getSize();
     SIZE = Math.min(dim.width, dim.height);
  }

  public void paint(Graphics g) {
     if (offScreenImage == null) {
         offScreenImage = createImage(SIZE, SIZE);
         offScreenGraphics = offScreenImage.getGraphics();

         offScreenGraphics.setColor(Color.red);
         g.setColor(Color.red);
         offScreenGraphics.fillOval(0, 0, SIZE, SIZE);
         g.fillOval(0, 0, SIZE, SIZE);
         for (int i = 0; i < NUM_POINTS; i++) {
             int x = randomInRange(0, SIZE);
             int y = randomInRange(0, SIZE);
             if (distance(x, y, SIZE/2, SIZE/2) < SIZE/2) {
                 offScreenGraphics.setColor(Color.white);
                 g.setColor(Color.white);
             }
             else {
                 offScreenGraphics.setColor(Color.black);
                 g.setColor(Color.black);
             }
             offScreenGraphics.drawLine(x, y, x, y);
             g.drawLine(x, y, x, y);
         }
     }
     else {
         g.drawImage(offScreenImage, 0, 0, null);
     }
  }
}
```

Better buffering

if want to see image as it is drawn, must be done in `paint`

when first loaded, have `paint` draw on the graphics screen and also to an off-screen buffer

on subsequent repaints, simply redraw the contents of the off-screen buffer

```
<applet code="Monte3.class" height=300 width=300>
  <param name="points" value=20000>
</applet>
```


view page

GUI Elements pada Applets

- Java mempunyai extensive library yang mendukung GUIs (Graphical User Interfaces)
  - Mempunyai element yang sesuai dengan HTML buttons, text boxes, text areas, …

- Setiap element harus dibuat dan secara eksplisit ditambahkan ke applet

```
nameLabel = new Label("User's name");
add(nameLabel);

nameField = new TextField(20);
nameField.setValue("Dave");
add(nameField);
```

- Java menyediakan beberapa class untuk mengontrol layout
  - `FlowLayout` adalah default
  - `BorderLayout` membolehkan penempatan element di sekitar borders applet
  - Sebuah `Panel` dapat berisi banyak element

Text Boxes

```
public class Monte4 extends Applet
{
  . . .

  private Label insideLabel;
  private TextField insideField;
  private Label outsideLabel;
  private TextField outsideField;

  public void init()
  {
    randy = new Random();
    NUM_POINTS =
     Integer.parseInt(getParameter("points"));
    Dimension dim = getSize();
    SIZE = Math.min(dim.width, dim.height);

    setLayout(new BorderLayout());

     Panel p = new Panel();
     insideLabel = new Label("Inside:");
     p.add(insideLabel);
     insideField = new TextField(5);
     p.add(insideField);
     outsideLabel = new Label("Outside:");
     p.add(outsideLabel);
     outsideField = new TextField(5);
     p.add(outsideField);

     add(p, BorderLayout.SOUTH);
  }
```

```
public void paint(Graphics g)
{
 . . .
   insideField.setText("0");
   outsideField.setText("0");
      . . .
      if (distance(x, y, SIZE/2, SIZE/2) < SIZE/2) {
        g.setColor(Color.white);
        int value =
          Integer.parseInt(insideField.getText())+1;
        insideField.setText(""+value);
      }
      else {
        g.setColor(Color.black);
        int value =
          Integer.parseInt(outsideField.getText())+1;
        outsideField.setText(""+value);
      }

    . . .
  }
}
```

view page

```
<applet code="Monte4.class" height=335 width=300>
  <param name="points" value=20000>
</applet>
```

Event Handling

- in order to handle events (e.g., text changes, button clicks), can use the *event delegation model*
  - Harus menentukan bahwa class meng-implementasikan the `ActionListener` interface

```
public class Monte5 extends Applet implements ActionListener
```

  - Setiap source dari events harus didaftarkan dalam applet

```
dotButton = new Button("Click to generate dots");
dotButton.addActionListener();
```

  - Harus mempunyai sebuah `actionPerformed` method untuk menangani events

```
public void actionPerformed(ActionEvent e)
{
  if (e.getSource() == dotButton) {
    drawDots();
  }
}
```

ActionListener

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import java.util.Random;

public class Monte5 extends Applet
                    implements ActionListener
{
  . . .
  private Button dotButton;

  public void init()
  {
    randy = new Random();
    NUM_POINTS =
      Integer.parseInt(getParameter("points"));
    Dimension dim = getSize();
    SIZE = dim.width;

    setLayout(new BorderLayout());
    dotButton =
      new Button("Click to generate dots");
    dotButton.addActionListener(this);
    add(dotButton, BorderLayout.SOUTH);

    drawCircle();
  }
```

```
  public void drawCircle()
  {
      CODE FOR DRAWING CIRCLE
  }

  public void drawDots()
  {
    drawCircle();

    Graphics g = getGraphics();
    for (int i = 0; i < NUM_POINTS; i++) {
        CODE FOR DRAWING DOTS
    }
  }

  public void paint(Graphics g)
  {
    g.drawImage(offScreenImage, 0, 0, null);
  }

  public void actionPerformed(ActionEvent e)
  {
    if (e.getSource() == dotButton) {
      drawDots();
    }
  }
}
```

view page

```
<applet code="Monte5.class" height=325 width=300>
  <param name="points" value=20000>
</applet>
```