Introduction

Nithi Thanon
Department of Computer Science
Faculty of Science, Prince of Songkla University

Contents

History of Java

- Java 1.0 - Sun Microsystems, Inc. (1995)
  - Java 1.1
- J2SE 1.2 - 1.2
  - J2SE - Java 2 Platform Standard Edition
  - J2SE 1.3
  - J2SE 1.4
- J2SE 5 - 1.5 - JDK 5
  - JDK - Java Development Kit
- Java SE 6 - 1.6 - JDK 6
  - Java SE - Java Platform, Standard Edition
- Java SE 7 - 1.7 - JDK 7
  - Sun Microsystems was acquired by Oracle (2010)
- Java SE 8 - 1.8 - JDK 8

- Which Java package do I need?
  - Software Developers

JDK (Java Development Kit)

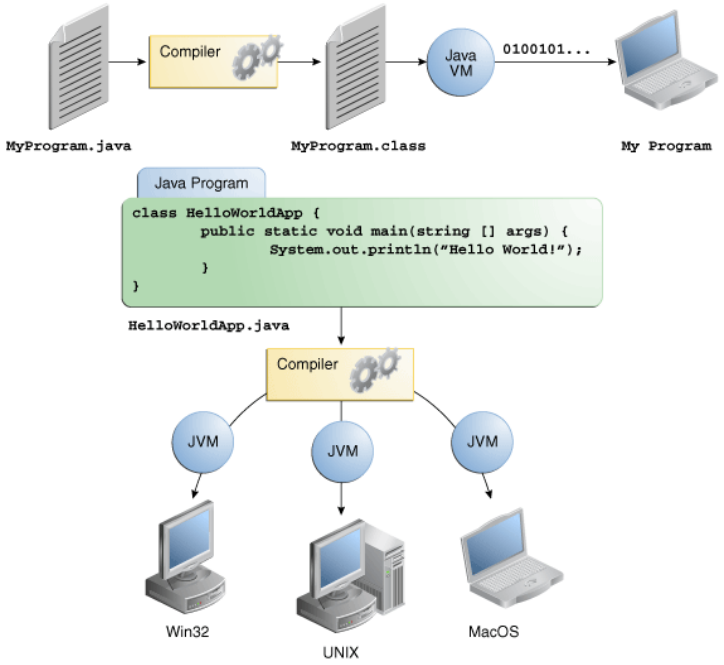
  - End user running Java on a desktop:

JRE (Java Runtime Environment)

- Integrated development environment (IDE)
  - NetBeans
  - Eclipse
  - JCreator
  - DrJava
  - BlueJ
  - IntelliJ

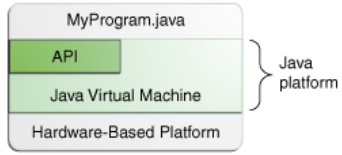
Java Overview

- Programming language
  - syntax from C
  - object model from C++
- Write Once, Run Anywhere
  - platform independent
  - compile program into *bytecode*
  - run by the *JVM (Java Virtual Machine)*
  - JVM on many different operating systems
- Object Oriented Programming (OOP)
- Similar to C++
- Large library



Java Platform

- The Java platform has two components:
  - The Java Virtual Machine
  - The Java Application Programming Interface (API)

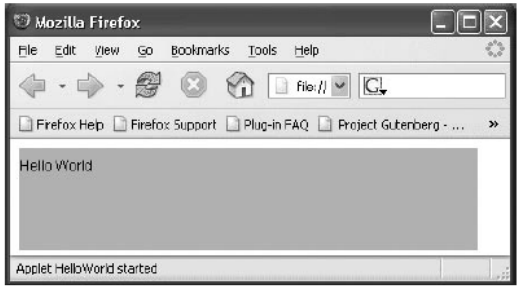


Java Application

- Java Console Application
  - Limited user interaction
  - Applications that run in the background
    - Anti-virus programs
    - Business Data processing/loading large volumes of data
  - *No mouse support*
  - *No graphical/window support*

- Java GUI Application
  - Working directly with the user
  - Display graphical information
  - *More complex to setup*
  - *Less convenient to run in the background*

- Java Applet
  - Run directly with a website
  - Enhances the use of a website
  - Display animation
  - *Cannot save data to the user's disk*



First Program

- Install Java SE Development Kit 8
- Create

```
notepad MyApp.java
public class MyApp {
   public static void main(String[] args) {
         System.out.println("Welcome to Java.");
   }
}
dir
12/28/2015  12:21 PM               115 MyApp.java
```

- Compile

```
javac MyApp.java
dir
12/28/2015  12:22 PM               420 MyApp.class
12/28/2015  12:21 PM               115 MyApp.java
```

- Run

```
java MyApp
Welcome to Java.
```


- Error
  - first error message is wrong
  - look at the second-guess

```
notepad MyApp.java
public class MyApp
   public static void main(String[] args)          // missing {
         System.out.println("Welcome to Java.");
   }
}

javac MyApp.java
MyApp.java:2: error: ';' expected
        public static void main(String[] args)
                                               ^
MyApp.java:5: error: class, interface, or enum expected
}
^
2 errors
```

Java Keywords

- keywords - cannot used as names
- values - true, false, null

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

Java Identifiers

- identifier - a name given to a method, a variable, or any other user-defined item
  - start with alphabet, underscore, dollar sign

```
Test            _top            $up     $_down
MaxLoad         sample23x       y1
```

```
12x
```

  - next may be letter, digit, dollar sign, underscore
  - underscore - readability

```
line_count      my_var
```

  - uppercase and lowercase are different

```
myvar           MyVar
```


Java Class Libraries

- packages - group classes
  - built-in class libraries, built-in methods

```
import java.lang.*;

public class MyApp
   public static void main(String[] args) {
        System.out.println(Math.sqrt(4));
   }
}
```

- Example packages
  - java.lang - basic language functionality and fundamental types
  - java.util - collection data structure classes
  - java.io - file operations
  - java.net - networking operations, sockets, DNS lookups
  - java.awt - basic hierarchy of packages for native GUI components

Comments

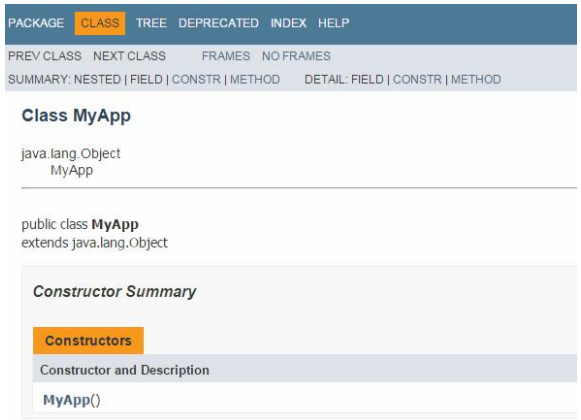
- Comment

```
// single-line comment

/*
multi-line comment
*/
```

- Javadoc Comment

```
/** javadoc
    comment
*/

/**
 * @author Somchai
 * @author CS-PSU
 * @version 2.0
 */
```

```
javadoc MyApp.java
```



Primitive Data types

- eight primitive data types - not objects

Type	*Bits*	*Range*
byte	8	-128 to +127
short	16	-32768 to 32767
int	32	-2147483648 to 2147483647
long	64	-9223372036854775808 to 9223372036854775807
char	16	Unicode 0 to 65536 (ASCII 0 to 127)
float	32	1.4E-45 to 3.4028235E38
double	64	4.9E-324 to 1.7976931348623157E308
boolean	-	true, false

Literals

- int

```
b = 100;
s = 10000;
i = 100000;

i = 128;
i = 0200;        // octal
i = 0X80;        // hexadecimal
i = 0B10000000   // binary
```

- long

```
l = 128L;
```

- double

```
d = 123.4;
d = 1.234E2;     // 1.234 * 10^2
```

- float

```
f = 123.4F;
```


- boolean

```
b = true;
b = false;
b = (rate > 0.5);
```

- char

```
c = 'A';
c = '\t';
c = '\n';
c = '\'';
c = '\"';
c = '\\';
c = '\uFFFF';
```

- string

```
s = "Hello";
s = "Hello \"Somchai\" \n";
```


Variables

- Syntax:

```
type-name variable-name;
```

- Capitalize

```
int numberOfStudents;
double x, y;
boolean isFinished;
char firstInitial, middleInitial, lastInitial;

double principal;               // Amount of money invested.
double interestRate;            // Rate as a decimal, not percentage.

int a = 10, b = 10;
double pi = 3.14159;
```

- Initialization - ตัวแปรต้องกำหนดค่าก่อนใช้งาน

```
int totalPrice;
System.out.println(totalPrice);  // Error

int totalPrice = 0;
System.out.println(totalPrice);
```

Dynamic Initialization

```
double radius = 4, height = 5;
double volume = (22.0/7.0) * radius * radius * height;
```

Assigning value

```
int myInt = 10;

int myInt;
myInt = 10;
```

Using variable

```
int myInt = 10;
System.out.print(myInt);
myInt = myInt + 20;
System.out.print(myInt + 30);
```


Integer type

```
byte  myInt8  = 2;              // -128   to +127
short myInt16 = 1;              // -32768 to +32767
int   myInt32 = 0;              // -2^31  to +2^31-1
long  myInt64 = -1;             // -2^63  to +2^63-1

int   myHex = 0xF;              // hexadecimal (base 16)
int   myOct = 07;               // octal (base 8)
```

Floating-point type

```
double myDouble = 3.14;
double myDouble2 = 3e2;         // 3*10^2 = 300

float myFloat = 3.14;           // 3.14 - double

float myFloat = 3.14F;          // Ok

float myFloat = (float)3.14;    // explicit casting
```


Char type

```
char myChar = 'A';
char myChar = '\u0000';         // \u0000 to \uFFFF

char c;
c = 'X';                        // X    88
c++;                            // Y    89
c = 90;                         // Z    90
```

Boolean type

```
boolean myBool = true;
boolean myBool = false;

boolean b;
b = true;                       // true
if(b) {
   System.out.println("This is executed.");
}
b = false;                      // false
if(b) {
   System.out.println("This is not executed.");
}

System.out.println(10 > 9);     // true
```


Example:

```
/* This class implements a simple program. */
public class Interest
{
 public static void main(String[] args)
 {
   /* Declare the variables. */
   double principal;     // The value of the investment.
   double rate;          // The annual interest rate.
   double interest;      // Interest earned in one year.

   /* Do the computations. */
   principal = 17000;
   rate = 0.027;
   // Compute the interest.
   interest = principal * rate;
   // Compute value of investment after one year, with interest.
   principal = principal + interest;

   /* Output the results. */
   System.out.print("The interest earned is $");
   System.out.println(interest);
   System.out.print("The value of the investment after one year is $");
   System.out.println(principal);
 } // end of main()
} // end of class Interest
```

Variable scope

- Code Block

```
public static void main(String[] args)
{
      int localVar; // local variable
}
```

- Anonymous Code Block

```
public static void main(String[] args)
{
      // Anonymous code block
      {
            int localVar = 10;
      }
      // localVar is unavailable from here
}
```


- Example:

```
// Demonstrate block scope.
public class ScopeDemo {
   public static void main(String args[]) {
      int x; // known to all code within main

      x = 10;
      if (x == 10) { // start new scope

            int y = 20; // known only to this block

            // x and y both known here.

            System.out.println("x and y: " + x + " " + y);
            x = y * 2;
      }
      // y = 100; // Error! y not known here

      // x is still known here.
      System.out.println("x is " + x);
   }
}
```


- Example:

```
// Demonstrate lifetime of a variable.
public class VarInitDemo {
   public static void main(String args[]) {
      int x;

      for (x = 0; x < 3; x++) {

            int y = -1; // y is initialized each time block is entered
            System.out.println("y is: " + y); // this always prints -1

            y = 100;
            System.out.println("y is now: " + y);
      }
   }
}
```

```
y is: -1
y is now: 100
y is: -1
y is now: 100
y is: -1
y is now: 100
```


- Example:

```
// This program will not compile.
public class NestVar {
   public static void main(String args[]) {
      int count;

      for (count = 0; count < 10; count = count + 1) {
            System.out.println("This is count: " + count);

            int count; // illegal!!!

            for (count = 0; count < 2; count++) {
                  System.out.println("This program is in error!");
            }
      }
   }
}
```

Arithmetic operators

- Arithmetic operators

Operator	Meaning
+	Addition (also unary plus)
–	Subtraction (also unary minus)
*	Multiplication
/	Division
%	Modulus
++	Increment
– –	Decrement

```
int x;
x = 3 + 2;                  // 5
x = 3 - 2;                  // 1
x = 3 * 2;                  // 6
x = 3 / 2;                  // 1
x = 3 % 2;                  // 1

double d;
d = 3 / 2;                  // 1
d = (double) 3 / 2;         // 1.5
d = 3.0 / 2;                // 1.5
d = 1 / 2;                  // 0
```


- % operator

```
int x;
double d;
x = 10 / 3;             // 3
x = 10 % 3;             // 1
d = 10 / 3;             //  3.3333333333333335
d = 10 % 3;             // 1.0
```

- ++ and -- operators

```
int x = 0;
++x;                    // x += 1
--x;                    // x -= 1

++x;                    // pre-increment
--x;                    // pre-decrement
x++;                    // post-increment
x--;                    // post-decrement

x = 5;
y = x++;                // y=5, x=6
x = 5;
y = ++x;                // y=6, x=6
```


Assignment operators

- Assignment operators

```
int x = 1;
int y = 2;
x = y;
y = x + 1;

int a, b, c;
a = b = c = 100;
```

- Shorthand Assignment operators

+=	–=	*=	/=
%=	&=	|=	^=

```
int x;
x = 0;
x += 5;                 // x = x + 5;
x -= 5;                 // x = x - 5;
x *= 5;                 // x = x * 5;
x /= 5;                 // x = x / 5;
x %= 5;                 // x = x % 5;
```


Casting Incompatible Types

- automatic type conversion
  - two types are compatible
  - destination type is larger than the source type

```
int i;
long l;
float f;
double d;
i = 10;
l = 10L;
f = i;
f = l;
d = i;
d = l;

// Error
i = f;
i = d;
l = f;
l = d;
```

- Casting Incompatible Types

```
int i;
long l;
float f;
double d;
i = 10;
l = 10L;
f = i;
f = l;
d = i;
d = l;

i = (int)f;
i = (int)d;
l = (long)f;
l = (long)d;
```


- Example:

```
public class CastDemo {
   public static void main(String args[]) {
         double x, y;
         byte b;
         int i;
         char ch;
         x = 10.0;
         y = 3.0;
         i = (int) (x / y); // cast double to int
         System.out.println("Integer outcome of x / y: " + i);
         i = 100;
         b = (byte) i;
         System.out.println("Value of b: " + b);
         i = 257;
         b = (byte) i; // unexpected value
         System.out.println("Value of b: " + b);
         b = 88; // ASCII code for X
         ch = (char) b;
         System.out.println("ch: " + ch);
   }
}
Integer outcome of x / y: 3
Value of b: 100
Value of b: 1
ch: X
```


Relational and Logical operators

- relational operator

Operator	Meaning
= =	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

- logical operator

Operator	Meaning
&	AND
|	OR
^	XOR (exclusive OR)
||	Short-circuit OR
&&	Short-circuit AND
!	NOT

- truth table

p	q	p & q	p | q	p ^ q	!p
False	False	False	False	False	True
True	False	False	True	True	False
False	True	False	True	True	True
True	True	True	True	False	False

```
boolean b;
b = (2 == 3);                  // false
b = (2 != 3);                  // true
b = (2 > 3);                   // false
b = (2 < 3);                   // true
b = (2 >= 3);                  // false
b = (2 <= 3);                  // true

boolean y;
y = (true && false);           // false
y = (true || false);           // true
y = !(true);                   // false
```

Bitwise operators

- Bitwise operators

```
int x;
x = 5 & 4;              // 101 & 100 = 100 (4)          // and
x = 5 | 4;              // 101 | 100 = 101 (5)          // or
x = 5 ^ 4;              // 101 ^ 100 = 001 (1)          // xor
x = 4 << 1;             // 100 << 1 = 1000 (8)          // left shift
x = 4 >> 1;             // 100 >> 1 = 10 (2)            // right shift
x = ~4;                 // ~00000100 = 11111011 (-5)    // invert
```


Operator precedence

- Use parentheses

```
boolean b;

b = 2 + 3 > 1 * 4 && 5 / 5 == 1;                // true

b = ((2 + 3) > (1 * 4)) && ((5 / 5) == 1);      // true
```

Precedence	Operator	Precedence	Operator
1	++ −−! ~	7	&
2	* / %	8	^
3	+ −	9	|
4	<< >> >>>	10	&&
5	< <= > >=	11	||
6	== !=	12	= op=

Short-Circuit operators

- &, | (ทำงานทั้งซ้ายและขวา)

```
public class MyApp {
   public static void main(String args[]) {
        int n, d;
        n = 10;
        d = 2;
        if (d != 0 && (n / d) == 5) {
                System.out.println("d = 2");
        }
        d = 0;
        if (d != 0 && (n / d) == 5) {            // not execute
                System.out.println("d = 0");
        }
        if (d != 0 & (n / d) == 5) {             // execute, error occurred
                System.out.println("d = 0");
        }
   }
}
```

```
d = 2
Exception in thread "main" java.lang.ArithmeticException: / by zero
   at MyApp.main(MyApp.java:13)
```


Expressions

- result of expressions
  - char, byte, short, int -> int
  - long -> long
  - float -> float
  - double -> double

```
byte b;
int i;
b = 10;
i = b * b;                      // OK, no cast needed
b = (byte) (b * b);             // cast needed!!

char ch1 = 'a', ch2 = 'b';
ch1 = (char) (ch1 + ch2);
```

String class

- String is a reference data type

```
String s = new String(" World");

String a = "Hello";
String b = " World";
```

- Combining strings

```
String c = a + b;                       // Hello World
a += b;                                 // Hello World
```

- Escape character

```
String s = "She said \"Hello!\" to me."; // She said "Hello!" to me.
```

- String compare

```
boolean x = a.equals(b);                // false
boolean y = "Hello".equals(a);          // true
boolean z = a.equalsIgnoreCase(b);      // false
```


- String Length

```
String s = "Hello";
int len = s.length();                   // 5
```

- String Methods

```
char c;
c = "Hello World".charAt(6);            // W

int i;
i = "Hello World".indexOf("World");     // 6

String s;
s = "Hello World".toLowerCase();        // hello world
s = "Hello World".toUpperCase();        // HELLO WORLD

s = " Hello World ".trim();             // Hello World

s = "Hello World".substring(6);         // World

s = "Hello World".substring(0, 4);      // Hell

String msg = "Hello World";
s = msg.toUpperCase();                  // HELLO WORLD
```


- ACSII table

	0	1	2	3	4	5	6	7	8	9
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
10	lf	vt	ff	cr	so	si	dle	dc1	dc2	dc3
20	cd4	nak	syn	etb	can	em	sub	esc	fs	gs
30	rs	us	sp	!	"	#	$	%	&	'
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	}	|	~	del		

Math class

```
double d;
int i;
i = Math.abs(-6);        // 6
d = Math.pow(2, 3);      // 8.0

i = Math.min(3, 2);      // 2
d = Math.min(3.0, 2.0);  // 2.0

i = Math.max(3, 2);      // 3
d = Math.max(3.0, 2.0);  // 3.0

d = Math.round(3.2);     // 3.0
d = Math.round(3.6);     // 4.0

d = Math.ceil(3.2);      // 4.0
d = Math.ceil(3.6);      // 4.0

d = Math.floor(3.2);     // 3.0
d = Math.floor(3.6);     // 3.0

d = Math.sqrt(4.0);      // 2.0
d = Math.sqrt(-4.0);     // Nan
```

```java
d = Math.PI * 5 * 5;              // 78.53981633974483
d = Math.PI * Math.pow(5, 2);     // 78.53981633974483

float f;
f = (float) (Math.PI * 5 * 5);    // 78.53982

double d = Math.random();
System.out.println(d);            // 0.2024343181782008

int lower = 1;
int upper = 100;
int rand = (int) (Math.random() * (upper - lower)) + lower;       // 65
```


Screen Output

System.out.println

```java
int x = 1;
int y = 2;
System.out.println("x: " + x);
System.out.println("y: " + y);
System.out.println("x + y: " + (x + y));
```

```
x: 1
y: 2
x + y: 3
```

System.out.printf

```java
int x = 1;
int y = 2;
System.out.printf("x: %d \n", x);
System.out.printf("y: %d \n", y);
System.out.printf("x + y: %d \n", (x + y));
```

```
x: 1
y: 2
x + y: 3
```


รูปแบบ System.out.printf

```
System.out.printf(format, item1, item2, ..., itemk);

    %b          true or false
    %c          'a'
    %d          200
    %f          123.456
    %s          "Hello"
```

```java
double d = 123.456;
System.out.printf("d: %f \n", d);
System.out.printf("d: %.2f \n", d);
System.out.printf("d: %6.2f \n", d);
System.out.printf("d: %4.2f \n", d);
System.out.printf("d: %10.2f \n", d);
System.out.printf("d: %-10.2f \n", d);
```

```
// 1234567890
d: 123.456000
d: 123.46
d: 123.46
d: 123.46
d:     123.46
d: 123.46
```


String.format

```
String s = String.format(format, item1, item2, ..., itemk);
```

```java
double d = 123.456;
String s = String.format("d: %.2f \n", d);
System.out.println(s);
```

```
d: 123.46
```

Keyboard Input

- java.util.Scanner class

```
Scanner kb = new Scanner(System.in);

double d = kb.nextDouble();
```

next()	string, delimited by spaces
nextLine()	line of string
nextByte()	byte
nextShort()	short
nextInt()	int
nextLong()	long
nextFloat()	float
nextDouble()	double


```
import java.util.Scanner;

public class ShowOutput {
 public static void main(String[] args) {
   Scanner kb = new Scanner(System.in);
   System.out.println("Name: ");
   String name = kb.nextLine();
   System.out.println("Age: ");
   int age = kb.nextInt();
   System.out.println("Salary: ");
   double salary = kb.nextDouble();
   System.out.printf("Name: %s Age: %d Salary %.2f", name, age, salary);
 }
}
```

```
Name: Somchai Age: 20 Salary 4500.50
```


Command Line Arguments

- accept parameter from command line

```
java Program arg1 arg2 arg3 ... argk
```

```
public class ShowInput {
 public static void main(String[] args) {
   String name = args[0];
   int age = Integer.parseInt(args[1]);
   double salary = Double.parseDouble(args[2]);
   System.out.printf("Name: %s Age: %d Salary %.2f", name, age, salary);
 }
}
```

- เรียกใช้

```
java ShowInput Somchai 20 4500.50
```

```
Name: Somchai Age: 20 Salary 4500.50
```


Programming Errors

- Syntax Errors

```
public class MyApp {
   public static void main(String[] args) {
       int i = 30
       System.out.println(i + 4);
   }
}
```

- Runtime Errors

```
public class MyApp {
   public static void main(String[] args) {
        int i = 1 / 0;
   }
}
```

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
   at MyApp.main(MyApp.java:3)
```

Logic Errors

```
public class MyApp {
   public static void main(String[] args) {
        int number1 = 3;
        int number2 = 3;
        int number3 = number1 + number2 / 2;      // 4
   }
}
```