

Session 02

Learning the Java Language

(<http://docs.oracle.com/javase/tutorial/java/index.html>)

Objectives

- Study some fundamentals of Java languages: Data types, variables, arrays, operators, logic constructs.
- Pass arguments to the main method
- Input/output variables
- Object-Oriented Programming Concepts: Class, Interface, Package.

Keywords and Identifiers

- Keywords: Almost of them are similar to those in C language
- Naming Convention:

Letter	Letters
\$	Digits, \$
–	–

- Java is a case-sensitive language
- Identifiers must be different to keywords

Primitive Data Types - Variables

- A *primitive* is a simple non-object data type that represents a single value. Java's primitive data types are:

Type	Bytes	Minimum	Maximum
char	2	\u0000	\uFFFF
byte	1	-2 ⁷	2 ⁷ - 1
short	2	-2 ¹⁵	2 ¹⁵ - 1
int	4	-2 ³¹	2 ³¹ - 1
long	8	-2 ⁶³	2 ⁶³ - 1
float	4		
double	8		
boolean	true/false		

Type var [=Initial value] ;

Operators

Category (Descending Precedence)	Operators
Unary	<code>++ -- + - ! ~ (type)</code>
Arithmetic	<code>* / %</code> <code>+ -</code>
Shift	<code><< >> >>></code>
Comparison	<code>< <= > >= instanceof</code> <code>== !=</code>
Bitwise	<code>& ^ </code>
Short-circuit	<code>&& </code>
Conditional	<code>? :</code>
Assignment	<code>= op=</code>

They are the same with
those in C language

Using Operators Demonstration

```

1  public class UseOps {
2      public static void main(String[] args)
3      {
4          int x=-1;
5          System.out.println("-1<<1: " + (x<<1) );
6          System.out.println("-1>>1: " + (x>>1) );
7          System.out.println("-1>>>1: " + (x>>>1));
8          System.out.println("3|4: " + (3|4));
9          System.out.println("3&4: " + (3&4));
10         System.out.println("3^4: " + (3^4));
11         String S="Hello";
12         boolean result= S instanceof String;
13         System.out.println("Hello is an instance of String: " + result);
14     }
15 }

```

Output - Chapter01 (run)

```

run:
-1<<1: -2
-1>>1: -1
-1>>>1: 2147483647
3|4: 7
3&4: 0
3^4: 7
Hello is an instance of String: true
BUILD SUCCESSFUL (total time: 0 seconds)

```

Using Operators Demonstration

Use 2 bytes to store value

```
Output - Chapter01 (run)
run:
-1<<1: -2
-1>>1: -1
-1>>>1: 2147483647
3|4: 7
3&4: 0
3^4: 7
Hello is an instance of String: true
BUILD SUCCESSFUL (total time: 0 seconds)
```

1: → 0000 0000 0000 0001
 1111 1111 1111 1110 (1-complement)
 -1 → 1111 1111 1111 1111 (2-complement)
 -1 <<1 → 1111 1111 1111 1110 (-2)

-1 → 1111 1111 1111 1111
 -1 >>1 → 1111 1111 1111 1111

-1 → 1111 1111 1111 1111
 -1 >>>1 → 0111 1111 1111 1111 (2147483647)

3 → 0000 0000 0000 0011
 4 → 0000 0000 0000 0100
 3|4 → 0000 0000 0000 0111 (7)

3 → 0000 0000 0000 0011
 4 → 0000 0000 0000 0100
 3&4 → 0000 0000 0000 0000 (0)

3 → 0000 0000 0000 0011
 4 → 0000 0000 0000 0100
 3^4 → 0000 0000 0000 0111 (7): XOR BIT

Literals and Value Variables

- Character: 'a'
- String: String S="Hello";
- Escape sequences: see the page 10
- Integral literals:
28, 0x1c, 0X1A (default: int).
123l, 123L (long)
- Floating point:
1.234 (default: double)
1.3f 1.3F
1.3E+21
1.3d 1.3D

Value variable

Stack

n

10

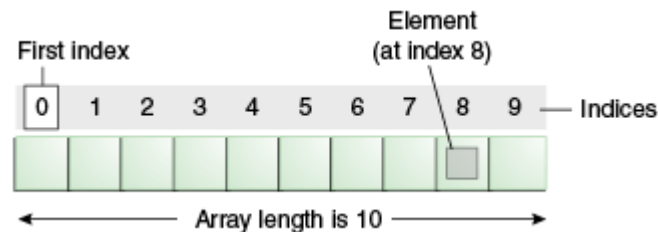
int n=10;

Java Expressions

- Java is an expression-oriented language. A simple expression in Java is either:
 - A constant: 7, false
 - A char - literal enclosed in single quotes: 'A', '3'
 - A String - literal enclosed in double quotes: "foo"
 - The name of any properly declared variables: x
 - Any two|one of the preceding types of expression that are combined with one of the Java binary operators: i++, x + 2, (x + 2)

One Dimensional Arrays (1)

- An *array* is a container object that holds a fixed number of values of a single type.
- The length of an array is established when the array is created.
- Each item in an array is called an *element*, and each element is accessed by its numerical *index*.



One Dimensional Arrays (2)

- Declaring a Variable to Refer to an Array

```
int[] anArray;
```

```
or float anArrayOfFloats[];
```

- Creating, Initializing, and Accessing an Array

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

- Copying Arrays
 - Use arraycopy method from System class.

One Dimensional Arrays (3)

```
int[] ar;
```

```
ar= new int[3];
```

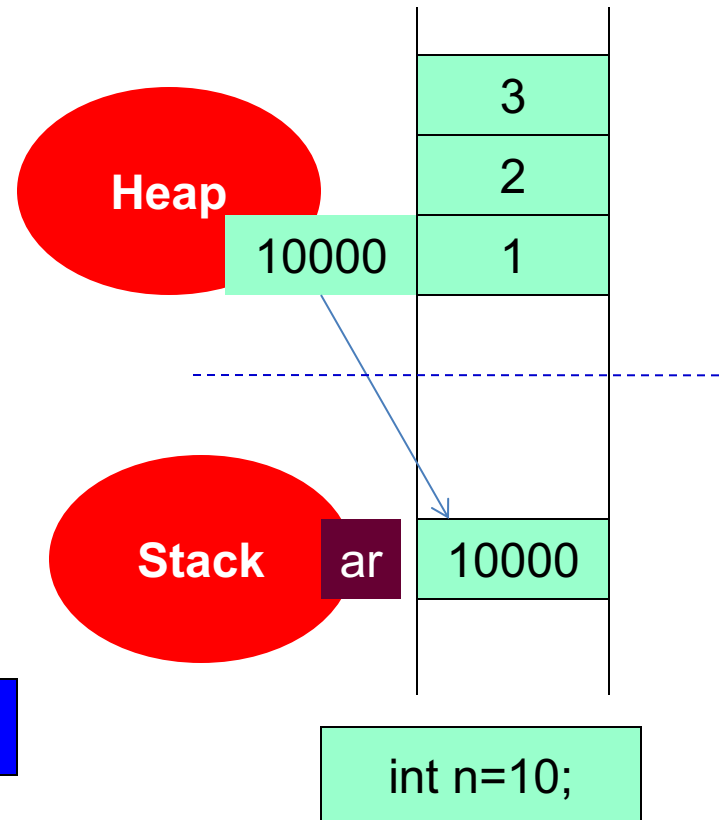
```
ar[0]=1; ar[1]=2; ar[2]=3;
```

```
int a2[];
```

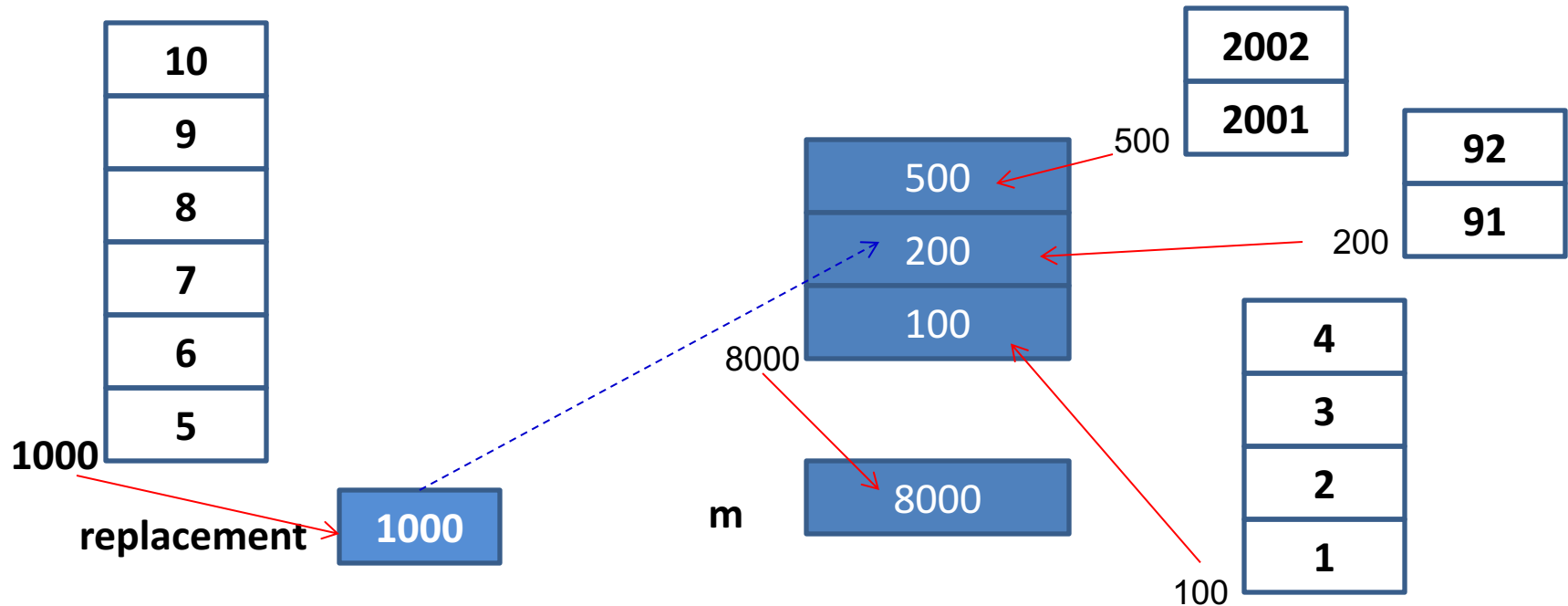
```
int[] a3 = {1,2,3,4,5};
```

```
int a4[] = {1,2,3,4,5};
```

Array is a reference variable



Multiple Dimensional Arrays



```
int m[][]= { {1,2,3,4}, {91,92}, {2001,2002}};
```

```
int[] replacement = {5,6,7,8,9,10};
```

```
m[1]= replacement;
```

`m[i][j]`

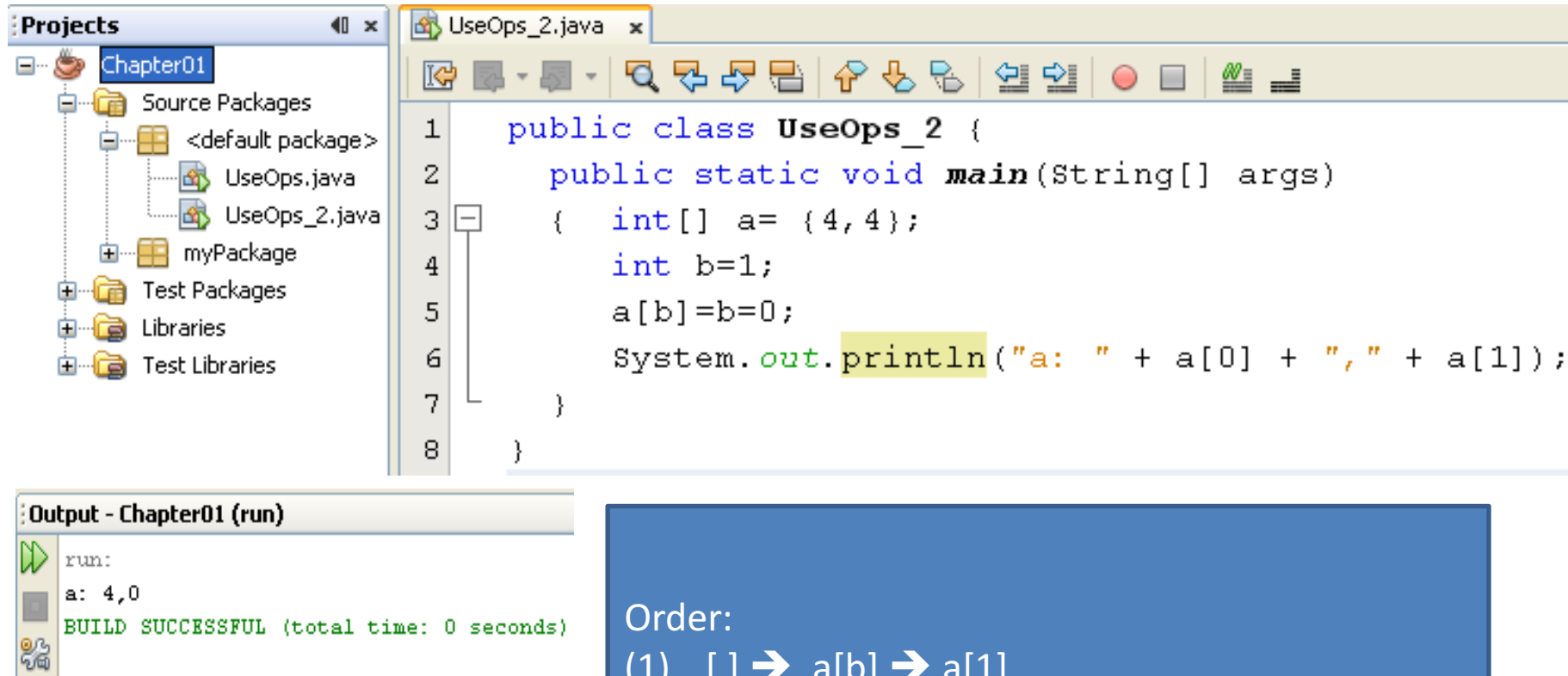
```
int[][] m; // declare a matrix
int r=10, c=5; // number of rows, columns
m= new int[r][c]; // memory allocate
```

Evaluating Expressions and Operator Precedence

- The compiler generally evaluates such expressions from the innermost to outermost parentheses, left to right.

```
int x = 1; int y = 2; int z = 3;  
int answer = ((8 * (y + z)) + y) * x;  
would be evaluated piece by piece as follows:  
((8 * (y + z) ) + y) * x  
((8 * 5) + y) * x  
(40 + y) * x  
42 * x  
42
```

Operator Precedence- Evaluation Order



The screenshot shows an IDE with a project named 'Chapter01'. The source packages include '<default package>' with 'UseOps.java' and 'UseOps_2.java', and 'myPackage'. The 'Test Packages' and 'Test Libraries' are also visible. The 'UseOps_2.java' file is open, showing the following code:

```

1  public class UseOps_2 {
2      public static void main(String[] args)
3      {
4          int[] a= {4,4};
5          int b=1;
6          a[b]=b=0;
7          System.out.println("a: " + a[0] + "," + a[1]);
8      }
9  }

```

The output window shows the following text:

```

run:
a: 4,0
BUILD SUCCESSFUL (total time: 0 seconds)

```

Order:

(1) $[] \rightarrow a[b] \rightarrow a[1]$

(2) $=$ (from the right) $\rightarrow b=0 \rightarrow$ return 0
 $\rightarrow a[1] = 0$

Basic Constructs

- They are taken from C-language
- Selection
 - if, if ... else
 - switch (char/int exp)... case ... default...
- Loops
 - for
 - do... while
 - while

Basic Logic Constructs

- They are the same with those in C-statements

An enhanced for loop

```

2  package com;
3  import java.lang.*;
4  public class Chao {
5      public static void main(String args[]) {
6          System.out.println("Hello");
7          int a[] = { 1,2,3,4,5};
8          for (int i=0;i<a.length;i++) System.out.print(a[i] + ",");
9          System.out.println();
10         for (int x : a) System.out.print(x + ",");
11         System.out.println();
12         for (int x : a) x+=10;
13         for (int i=0;i<a.length;i++) System.out.print(a[i] + ",");
14         System.out.println();
15     }
16 }

```

Read only

a

1	2	3	4	5
---	---	---	---	---

x

1

Output - P1 (run)

```

run:
Hello
1,2,3,4,5,
1,2,3,4,5,
1,2,3,4,5,

```

The String type

- A String represents a sequence of zero or more Unicode characters.
 - `String name = "Steve";`
 - `String s = "";`
 - `String s = null;`
- String concatenation.
 - `String x = "foo" + "bar" + "!";`
- Java is a case-sensitive language.

Type Conversions and Explicit Casting

- * Widening Conversion: OK
- Narrowing conversion: Not allowed. We must use explicit casting.
- A boolean can not be converted to any other type.
- A non-boolean can be converted to another non-boolean type.

```

1 public class Casting_Convert_1 {
2     public static void main (String[] args)
3     {
4         short x, y = 256;
5         byte m, n = 6;
6         x = n ; // Systematic Conversion
7         n = y; // narrow conversion
8         n = (byte) y; // narrow casting, possible loss of precision
9         System.out.println(n);
10    }

```

```

1 public class Casting_Convert_1 {
2     public static void main (String[] args)
3     {
4         short x, y = 256;
5         byte m, n = 6;
6         x = n ; // Systematic Conversion
7         n = (byte) y; // narrow casting, possible loss of p
8         System.out.println(n);
9     }

```

Output - Chapter04 (run)

```

run:
0
BUILD SUCCESSFUL (total time: 0 seconds)

```

0000 0001

0000 0000

y

n

Scope of a Variable

```

ScopeDemo.java x
Source History
2 public static void main (String[] args) {
3     int x=2, k=2;
4     if(x<2) {
5         int y=3;
6         int z=4;
7     }
8     y=6;
9     for (int i=1; i<3; i++) x+=i;
10    k+=i;
11 }
12 }

```

Scope of the variable y

Scope of the variable i

Input/Output Data

- Class `java.lang.System`
- Class `java.util.Scanner`

Refer to Java documentation:
`java.lang.String` class,
- the **`format`** method,
- format string
for more details

```

1  /* Write a program that will accept an array of integers then
2     print out entered value and the sum of values
3  */
4  import java.util.Scanner;
5  public class InputOutputDemo {
6      public static void main (String args[])
7      {
8          int a[]; // array of integers
9          int n; // number of elements of the array
10         int i; // variable for traversing the array
11         Scanner sc= new Scanner(System.in); // object for the keyboard
12         System.out.print("Enter number of elements: ");
13         n = Integer.parseInt(sc.nextLine());
14         a = new int[n]; // mem. allocating for elements of the array
15         for (i=0;i<n;i++)
16         {
17             System.out.print("Enter the " + (i+1) + "/" + n + " element: ");
18             a[i]=Integer.parseInt(sc.nextLine());
19         }
20         System.out.print("Entered values: ");
21         for (i=0;i<n;i++) System.out.format("%5d", a[i]);
22         int S=0;
23         for (int x: a) S+=x;
24         System.out.println("\nSum of values: " + S);
25     }
26 }
    
```

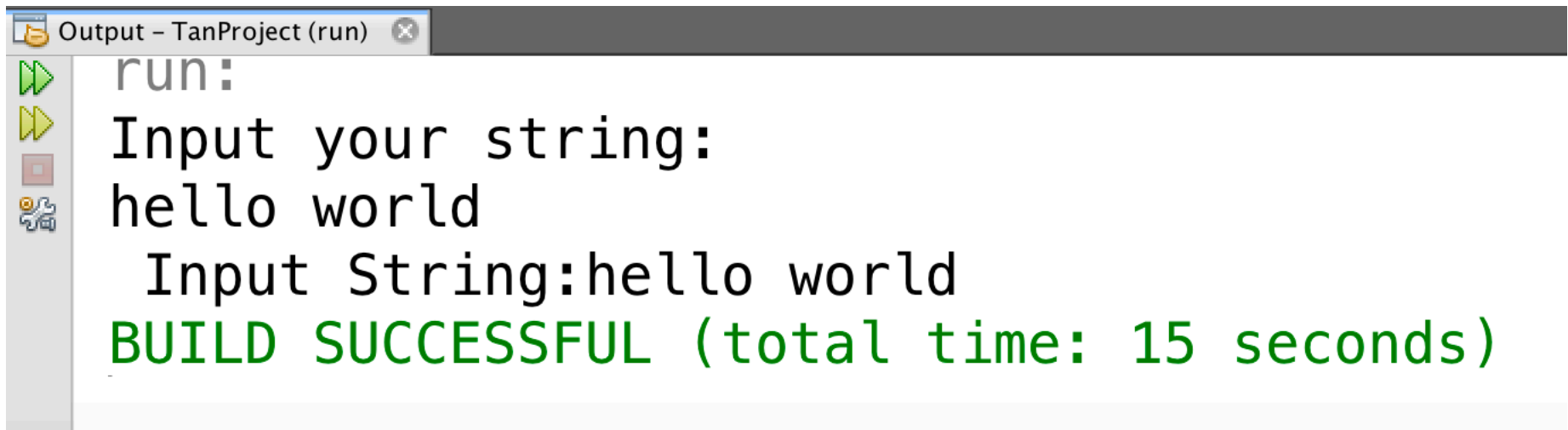
`n= sc.nextInt();`

```

Output - Chapter01 (run) #2
run:
Enter number of elements: 5
Enter the 1/5 element: 1
Enter the 2/5 element: 4
Enter the 3/5 element: 2
Enter the 4/5 element: 0
Enter the 5/5 element: 7
Entered values:      1      4      2      0      7
Sum of values: 14
BUILD SUCCESSFUL (total time: 11 seconds)
    
```

Input/Output Data

```
Scanner sc= new Scanner(System.in);  
System.out.println("Input your string: ");  
String str= sc.nextLine();  
System.out.println(" Input String:"+ str);
```



The screenshot shows an IDE output window titled "Output - TanProject (run)". The output text is as follows:

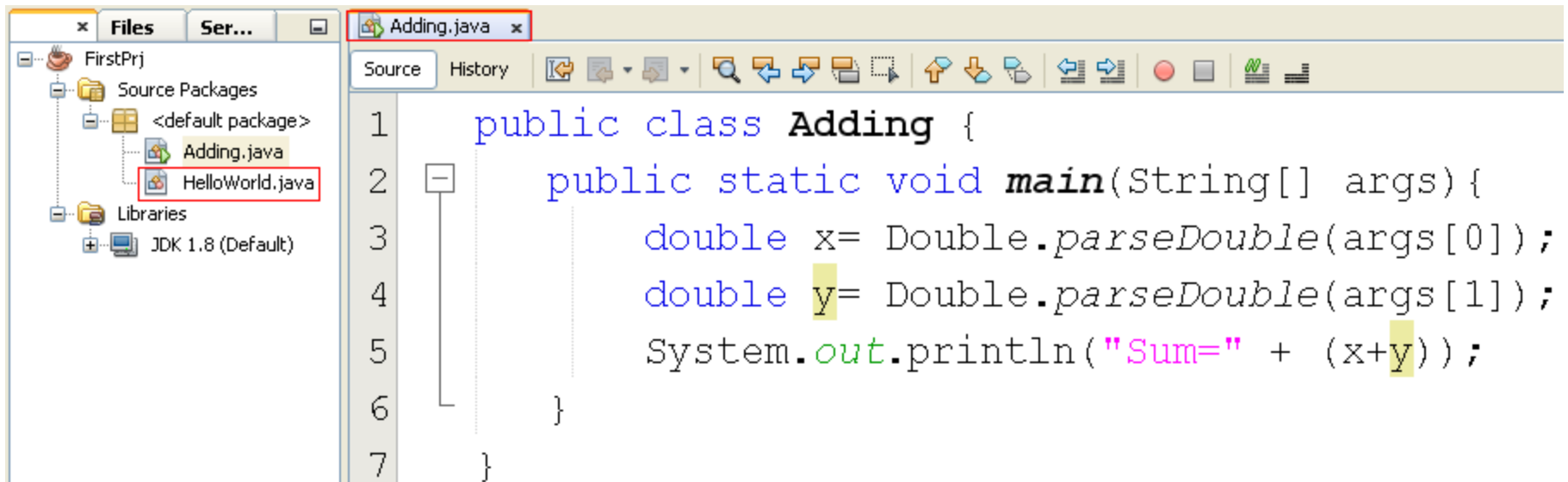
```
run:  
Input your string:  
hello world  
Input String:hello world  
BUILD SUCCESSFUL (total time: 15 seconds)
```

The output window includes a vertical toolbar on the left with icons for running, stepping through, and debugging the code.

Elements of Java Style

- Proper Use of Indentation
 - Statements within a block of code should be indented relative to the starting/ending line of the enclosing block.
- Use Comments Wisely
- Placement of Braces
 - Opening brace at the end of the line of code that starts a given block. Each closing brace goes on its own line, aligned with the first character of the line con.
- Descriptive Variable Names

Pass Arguments to the method main

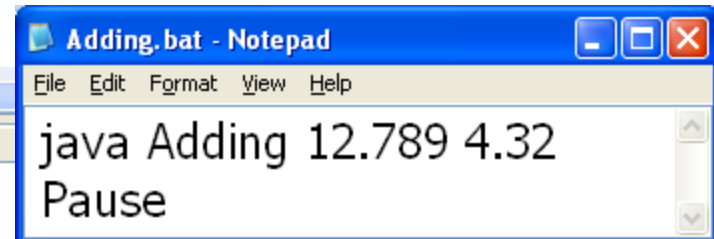
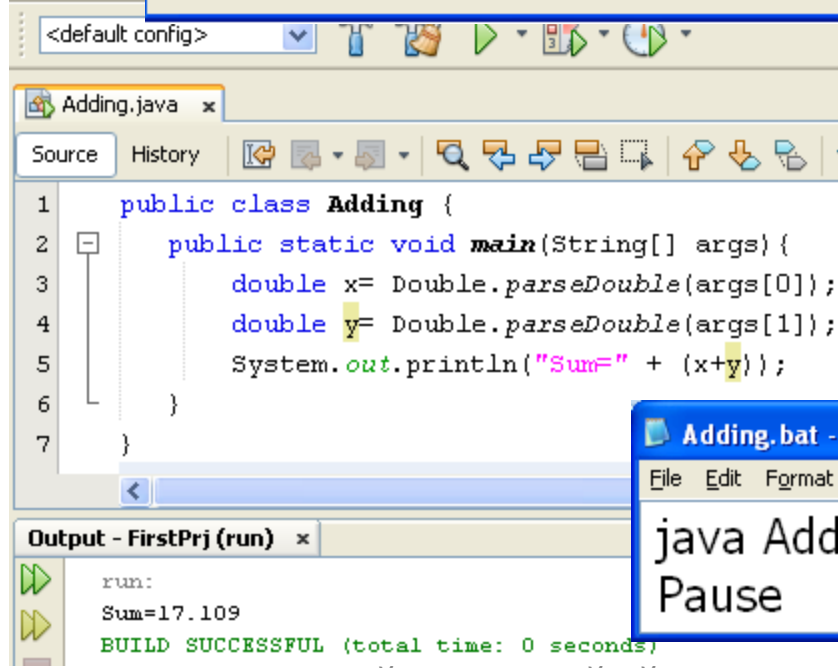
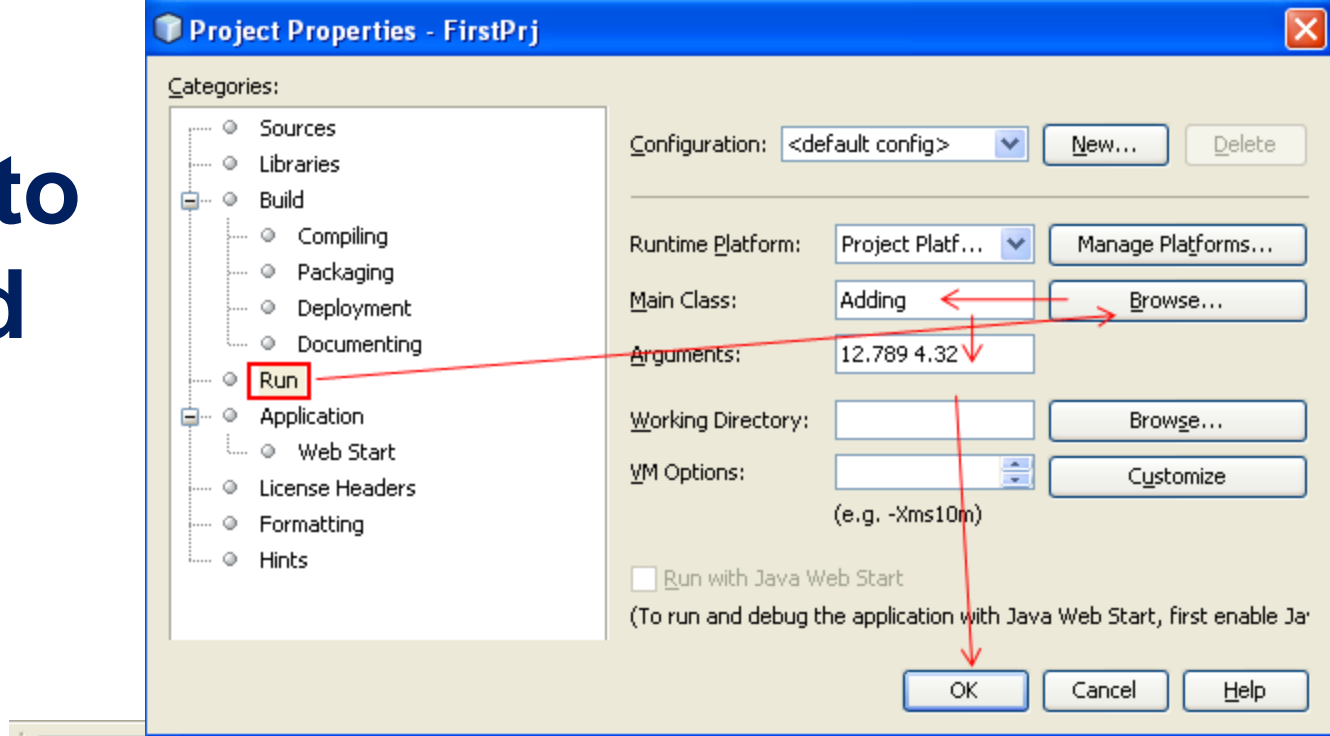
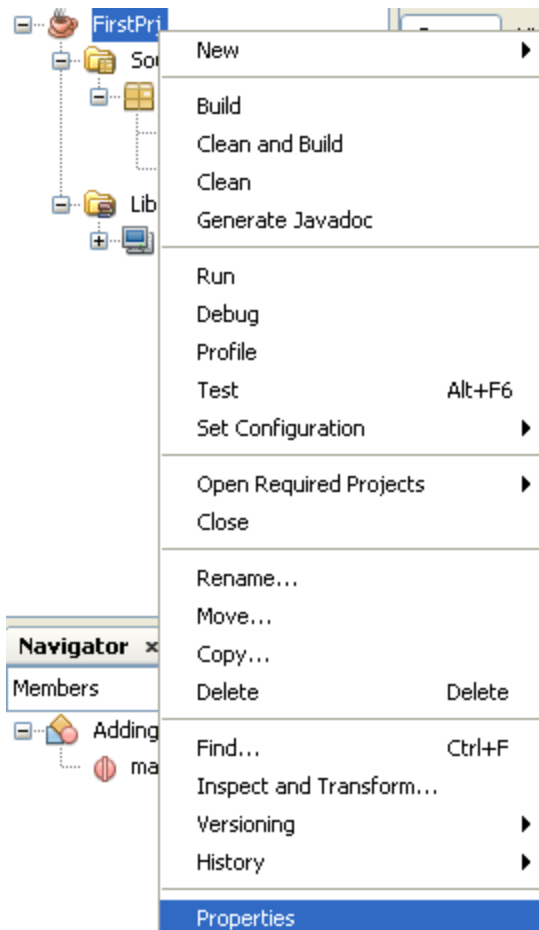


```

1  public class Adding {
2      public static void main(String[] args){
3          double x= Double.parseDouble(args[0]);
4          double y= Double.parseDouble(args[1]);
5          System.out.println("Sum=" + (x+y));
6      }
7  }
    
```


Pass

Arguments to the method main



Home Work

- Change base number:
 - Input *decimal* number
 - Output binary number
- Bubble sort:
 - Input: interger array
 - Output: sorted array.

What Is a Package?

- A package is a namespace that organizes a set of related classes and interfaces.
- The Java platform provides an enormous class library (a set of packages) suitable for use in your own applications called API.
 - For example, a String object contains state and behavior for character strings.

User-Defined Package

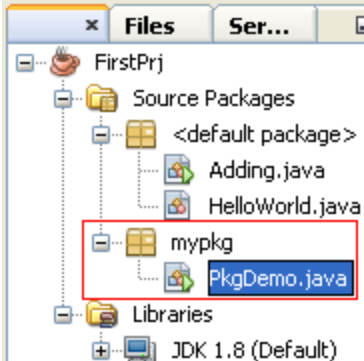
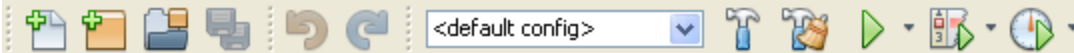
- Add a Java class

New Java Class

Steps	Name and Location
1. Choose File Type	Class Name: <input type="text" value="PkgDemo"/>
2. Name and Location	Project: <input type="text" value="FirstPrj"/>
	Location: <input type="text" value="Source Packages"/>
	Package: <input type="text" value="mypkg"/>
	Created File: <input type="text" value="K:\GiangDay\FU\Java-OOP\Demos\FirstPrj\src\mypkg\PkgDemo.java"/>

FirstPrj - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run



Source History

```

1 package mypkg;
2 public class PkgDemo {
3     public static void main(String[] args){
4         System.out.println("Package Demo.");
5     }
6 }

```

If package is used, it must be the first line in Java code

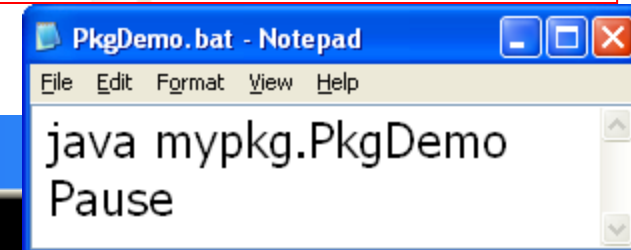
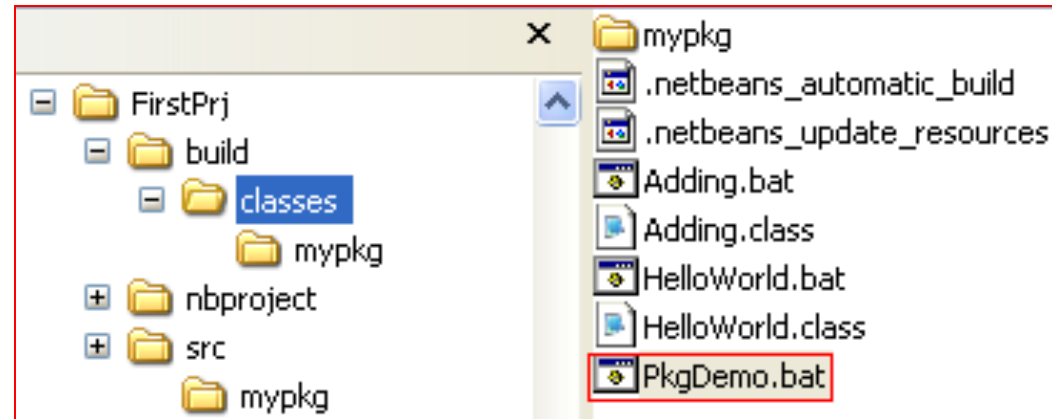
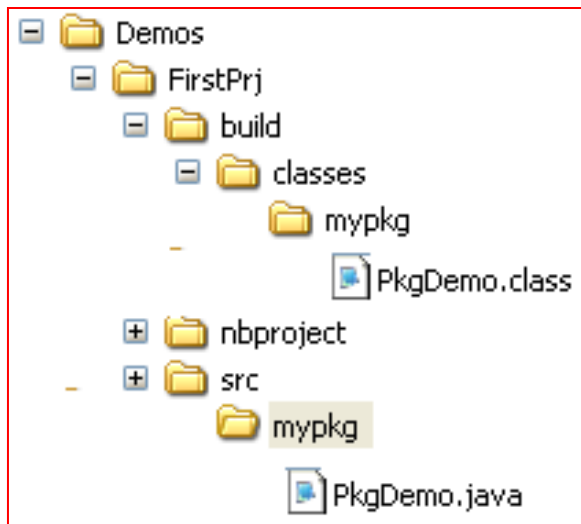
Output - FirstPrj (run)

```

run:
Package Demo.
BUILD SUCCESSFUL (total time: 0 seconds)

```

User-Defined Package



```
C:\ Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\USER>k:

K:\>cd K:\GiangDay\FU\Java-OOP\Demos\FirstPrj\build\classes\mypkg

K:\GiangDay\FU\Java-OOP\Demos\FirstPrj\build\classes\mypkg>java PkgDemo
Error: Could not find or load main class PkgDemo

K:\GiangDay\FU\Java-OOP\Demos\FirstPrj\build\classes\mypkg>cd..

K:\GiangDay\FU\Java-OOP\Demos\FirstPrj\build\classes>java mypkg.PkgDemo
Package Demo.

K:\GiangDay\FU\Java-OOP\Demos\FirstPrj\build\classes>
```

Summary

- The core concepts behind object-oriented programming: objects, interfaces, classes, and inheritance.
- The traditional features of the language, including variables, arrays, data types, operators, and control flow.