



Jurusan Teknik Komputer dan Informatika

Politeknik Negeri Bandung

Pertemuan 2 Java Fundamental 1

3.1 – 3.6

D4 Kelas 1A/1B

Dosen Pengampu :
Zulkifli Arsyad, Wendy Wirasta

- Comment
- Data Type
- Variable
- Operators
- String

Comment

Sr.No.	Comment & Description
1	<code>/* text */</code> The compiler ignores everything from <code>/*</code> to <code>*/</code> . When longer comments are needed
2	<code>//text</code> The compiler ignores everything from <code>//</code> to the end of the line.
3	<code>/** documentation */</code> This is a documentation comment and in general its called doc comment . The JDK javadoc tool uses <i>doc comments</i> when preparing automatically generated documentation.

Listing 3.1 FirstSample/FirstSample.java

```
1  /**
2   * This is the first sample program in Core Java Chapter 3
3   * @version 1.01 1997-03-22
4   * @author Gary Cornell
5   */
6  public class FirstSample
7  {
8      public static void main(String[] args)
9      {
10         System.out.println("We will not use 'Hello, World!'");
11     }
12 }
```

Comment documentation

```
Source History
7 import java.io.*;
8
9 /**
10  * <h1>Add Two Numbers!</h1>
11  * The AddNum program implements an application that
12  * simply adds two given integer numbers and Prints
13  * the output on the screen.
14  * <p>
15  * <b>Note:</b> Giving proper comments in your program makes it more
16  * user friendly and it is assumed as a high quality code.
17  *
18  * @author Zulkifli
19  * @version 1.0
20  * @since 2014-03-31
21  */
22
23 public class AddNum {
24
25     /**
26      * This method is used to add two integers. This is
27      * a the simplest form of a class method, just to
28      * show the usage of various javadoc Tags.
29      * @param numA This is the first paramter to addNum method
30      * @param numB This is the second parameter to addNum method
31      * @return int This returns sum of numA and numB.
32      */
33     public int addNum(int numA, int numB) {
34         return numA + numB;
35     }
36
37     /**
38      * This is the main method which makes use of addNum method.
39      * @param args Unused.
```

```
/**
 * This is the main method which makes use of addNum method.
 * @param args Unused.
 * @exception IOException On input error.
 * @see IOException
 */
}

public static void main(String args[]) throws IOException {
    AddNum obj = new AddNum();
    int sum = obj.addNum(10, 20);

    System.out.println("Sum of 10 and 20 is :" + sum);
}
```

Tag	Description	Syntax
@author	Adds the author of a class.	@author name-text
{@code}	Displays text in code font without interpreting the text as HTML markup or nested javadoc tags.	{@code text}
{@docRoot}	Represents the relative path to the generated document's root directory from any generated page.	{@docRoot}
@deprecated	Adds a comment indicating that this API should no longer be used.	@deprecated deprecatedtext
@exception	Adds a Throws subheading to the generated documentation, with the classname and description text.	@exception class-name description
{@inheritDoc}	Inherits a comment from the nearest inheritable class or implementable interface.	Inherits a comment from the immediate superclass.
{@link}	Inserts an in-line link with the visible text label that points to the documentation for the specified package, class, or member name of a referenced	{@link package.class#member label}

Generate Javadoc

javadoc -d [path to javadoc destination directory] [package name]

```
D:\ZULKIFLI\POLBAN\PENGAJARAN2022\GENAP20212022\TeknikPemrograman\Pertemuan2\AddNum\src\addnum>javadoc -d D:\ZULKIFLI\POLBAN\PENGAJARAN2022\GENAP20212022\TeknikPemrograman\Pertemuan2\AddNum\javadoc AddNum.java
Loading source file AddNum.java...
Constructing Javadoc information...
Standard Doclet version 11.0.10
Building tree for all the packages and classes...
Generating D:\ZULKIFLI\POLBAN\PENGAJARAN2022\GENAP20212022\TeknikPemrograman\Pertemuan2\AddNum\javadoc\addnum\AddNum.html...
Generating D:\ZULKIFLI\POLBAN\PENGAJARAN2022\GENAP20212022\TeknikPemrograman\Pertemuan2\AddNum\javadoc\addnum\package-summary.html...
Generating D:\ZULKIFLI\POLBAN\PENGAJARAN2022\GENAP20212022\TeknikPemrograman\Pertemuan2\AddNum\javadoc\addnum\package-tree.html...
Generating D:\ZULKIFLI\POLBAN\PENGAJARAN2022\GENAP20212022\TeknikPemrograman\Pertemuan2\AddNum\javadoc\constant-values.html...
Building index for all the packages and classes...
Generating D:\ZULKIFLI\POLBAN\PENGAJARAN2022\GENAP20212022\TeknikPemrograman\Pertemuan2\AddNum\javadoc\overview-tree.html...
Generating D:\ZULKIFLI\POLBAN\PENGAJARAN2022\GENAP20212022\TeknikPemrograman\Pertemuan2\AddNum\javadoc\index-all.html...
Building index for all classes...
Generating D:\ZULKIFLI\POLBAN\PENGAJARAN2022\GENAP20212022\TeknikPemrograman\Pertemuan2\AddNum\javadoc\allclasses-index.html...
Generating D:\ZULKIFLI\POLBAN\PENGAJARAN2022\GENAP20212022\TeknikPemrograman\Pertemuan2\AddNum\javadoc\allpackages-index.html...
Generating D:\ZULKIFLI\POLBAN\PENGAJARAN2022\GENAP20212022\TeknikPemrograman\Pertemuan2\AddNum\javadoc\deprecated-list.html...
Building index for all classes...
Generating D:\ZULKIFLI\POLBAN\PENGAJARAN2022\GENAP20212022\TeknikPemrograman\Pertemuan2\AddNum\javadoc\allclasses.html...
```

PACKAGE CLASS TREE DEPRECATED INDEX HELP

ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Package addnum

Class AddNum

java.lang.Object
addnum.AddNum

public class **AddNum**
extends java.lang.Object

Add Two Numbers!

The AddNum program implements an application that simply adds two given integer numbers and Prints the output on the screen.

Note: Giving proper comments in your program makes it more user friendly and it is assumed as a high quality code.

Since:
2022-02-09

Constructor Summary

Constructors	
Constructor	Description
AddNum()	

Method Summary

There are eight primitive types in Java, Four of them are integer types; two are floating-point number types; one is the character type char

Type	Storage Requirement	Range (Inclusive)
int	4 bytes	−2,147,483,648 to 2,147,483, 647 (just over 2 billion)
short	2 bytes	−32,768 to 32,767
long	8 bytes	−9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
byte	1 byte	−128 to 127

Table 3.2 Floating-Point Types

Type	Storage Requirement	Range
float	4 bytes	Approximately $\pm 3.40282347\text{E}+38\text{F}$ (6–7 significant decimal digits)
double	8 bytes	Approximately $\pm 1.79769313486231570\text{E}+308$ (15 significant decimal digits)

```
public static void main(String[] args) {  
    byte angka1 = 125;  
    byte angka2 = 6;  
    byte hasil = (byte) ( angka1+angka2);  
    System.out.println(hasil);  
}
```

Angka 1	Angka 2	Hasil
125	2	127
125	3	-128
125	4	-127
125	5	-126
125	6	-125

- In Java, every variable has a type

```
double salary;  
int vacationDays;  
long earthPopulation;  
boolean done;
```

- Initializing Variable

- ```
int vacationDays;
vacationDays = 12;
```

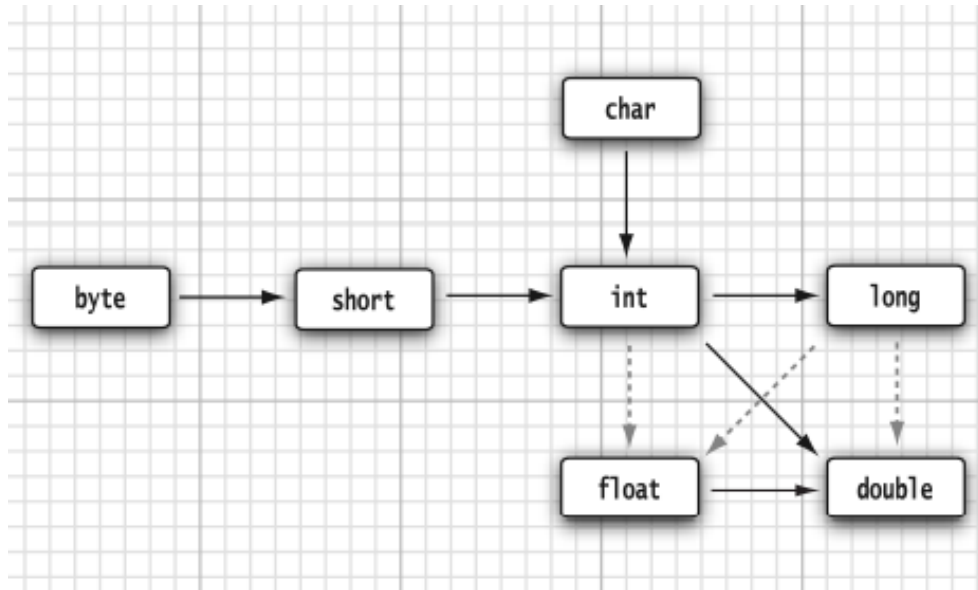
```
double salary = 65000.0;
System.out.println(salary);
int vacationDays = 12; // OK to declare a variable here
```

- The usual arithmetic operators `+`, `-`, `*`, `/` are used in Java for addition, subtraction, multiplication, and division
- Mathematical Function
  - `Math.sqrt(x)`;
  - `Math.pow(x,a)`;
  - Trigonometry function
    - `Math.sin`, `Math.cos`, `Math.tan`
  - Exponential function
    - `Math.exp`, `math.log`
- Cast



# Conversion between numeric types

- Conversion between numeric types



- Cast

```
double x = 9.997;
int nx = (int) x;
```

- Substring (extract sub string from large string)

```
String greeting = "Hello";
String s = greeting.substring(0, 3);
```

- creates a string consisting of the characters "Hel".
- Concatenation ( join 2 string or more)

```
String expletive = "Expletive";
String PG13 = "deleted";
String message = expletive + PG13;
```

- Testing String for Equality

```
s.equals(t)
```

```
"Hello".equals(greeting)
```

- Empty and Null String

```
if (str.length() == 0)
```

```
if (str == null)
```

or

```
if (str != null && str.length() != 0)
```

```
if (str.equals(""))
```

- String API
  - String class in java contain more than 50 method

- `char charAt(int index)`  
returns the code unit at the specified location. You probably don't want to call this method unless you are interested in low-level code units.
- `int codePointAt(int index) 5.0`  
returns the code point that starts at the specified location.
- `int offsetByCodePoints(int startIndex, int cpCount) 5.0`  
returns the index of the code point that is `cpCount` code points away from the code point at `startIndex`.
- `int compareTo(String other)`  
returns a negative value if the string comes before `other` in dictionary order, a positive value if the string comes after `other` in dictionary order, or 0 if the strings are equal.
- `IntStream codePoints() 8`  
returns the code points of this string as a stream. Call `toArray` to put them in an array.
- `new String(int[] codePoints, int offset, int count) 5.0`  
constructs a string with the `count` code points in the array starting at `offset`.
- `boolean equals(Object other)`  
returns `true` if the string equals `other`.
- `boolean equalsIgnoreCase(String other)`  
returns `true` if the string equals `other`, except for upper/lowercase distinction.
- `boolean startsWith(String prefix)`
- `boolean endsWith(String suffix)`  
returns `true` if the string starts or ends with `suffix`.