Chapter 3: Introduction to Objects and Input/Output

Chapter Objectives

- Learn about **objects** and **reference variables**.
- Explore how to use **predefined methods** in a program.
- Become familiar with the `class` **`String`**.
- Learn how to use **input and output dialog boxes** in a program.
- Explore how to **format the output of decimal numbers** with the `String` method `format`.

Java Variables

```
1. int x;
2. x=45;
```

Figure 3-1 Variable x and its data

```
3. String str;
4. str = "java programming";
```

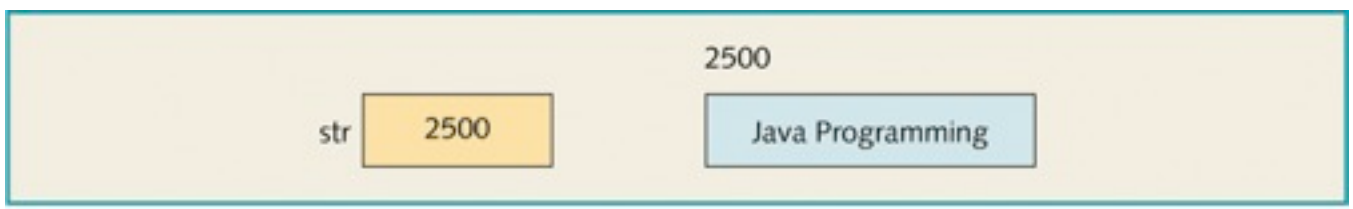


Figure 3-2 Variable `str` and the data it points to

Java Variables

- There are two types of variables in Java:
  - primitive type variables
  - reference variables.
- **Primitive type variables** directly store data into their memory space.

```
int x;
x = 45;
```

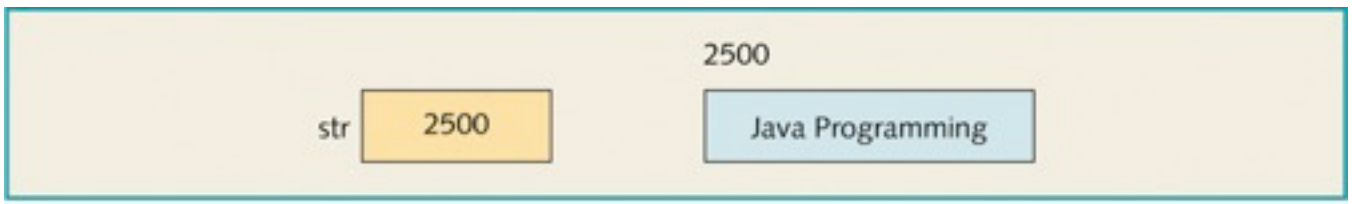
Figure 3-1 Variable x and its data

- The variable x can store DIRECTLY an `int` value in its memory space.
- The second statement DIRECTLY stores 45 in x.

Object and Reference Variables

- In reality,
- `str = "Java Progrmming"` =

```
String str = new String("Java Programming");
```



- The variable `str` ***cannot*** **directly** store data in its memory space.
- The variable `str` stores the memory location, that is, the **address of the memory space** where the actual data is stored.

So… what is `new`?

- In java `new` is an operator that causes the system to:
  1. Allocate memory space of a specific type,
  2. Store data in that space,
  3. Return the address of that space.

- Remember:
  - `String` → calss type
  - `str` → object of that class

- So, **Reference variables** are variables that store the address of the **object** (`str`). containing the data `("Java Programming")`.
- An **object** is an instance of a class and the operator new is used to instantiate an object.

Object and Reference Variables

- `str` is a **reference variable** and the memory space (2500) where the string data is stored is called **string object** or **instance of class String** .
- In java, any variable declared using a class is a ***reference variable***.
- String objects are **immutable ;** that is , once they are created ,they cannot be changed.

Object and Reference Variables

```
String str;
str = "Hello there!";
```

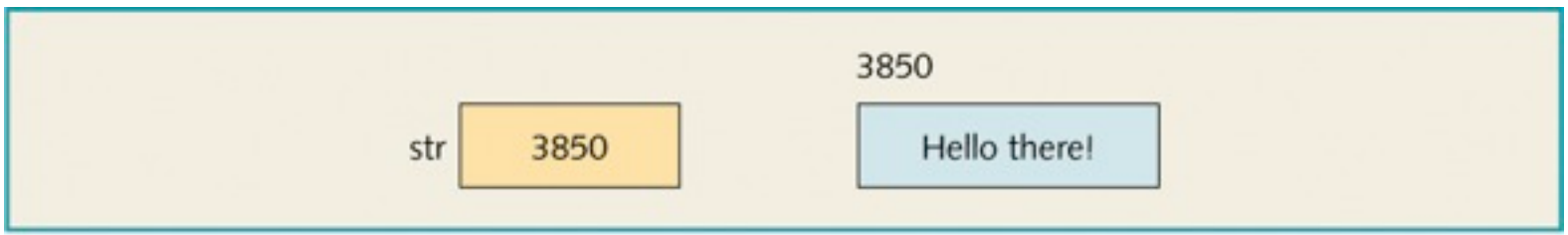


Figure 3-4 Variable `str`, its value, and the object `str`

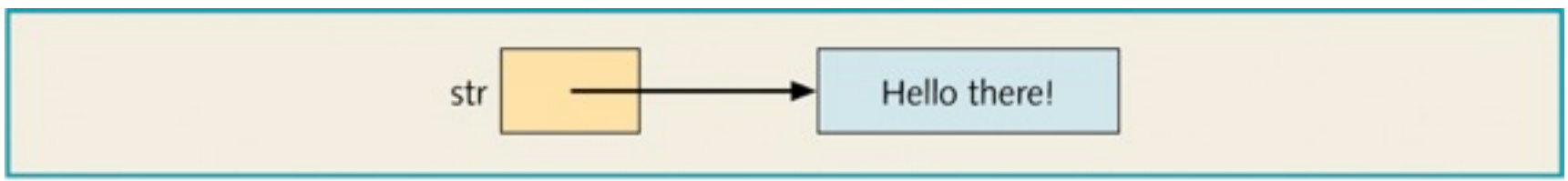


Figure 3-5 Variable `str` and the object `str`

Object and Reference Variables

- Java system reclaims unused memory spaces for later use → **Garbage collection.**
- As we saw before, the assignment operator(=) stores the address of that memory space into the variable `str`. So, `String` objects can be instantiated without using the **`new`** operator.
- This is because the `class String` is so important in Java that it has defined the assignment operator for the `class String`.
- We typically use the assignment operator to instantiate a `String` object.

Using Predefined Classes and Methods in a Program

- There are many predefined packages, classes, and methods in Java.
- Library: A collection of packages.
- Package: Contains several classes.
- Class: Contains several methods.
- Method: A set of instructions.
  - `main` method executes automatically when you run the program.
  - Other methods executes only when you activate them (call them).

Using Predefined Classes and Methods in a Program

To use a method (`pow`) you must know:

- Name of the class containing the method. **(`Math`).**
- Name of the package containing the class **(`java.lang`).**
- Name of the method - **(`pow`),** what the method does , number of its parameters (its has two parameters), type of each parameter and the return type.
- **`Math.pow(x, y)`** $= x^y$

Using Predefined Classes and Methods in a Program

- Example method call:

```
import java.lang; //imports package

Math.pow(2, 3);   //calls power method
                      // in class Math ,executes
it and                // returns the value of 2^3.
```

- **Dot (.) operator**: Used to access the method in the class.

The API documentation

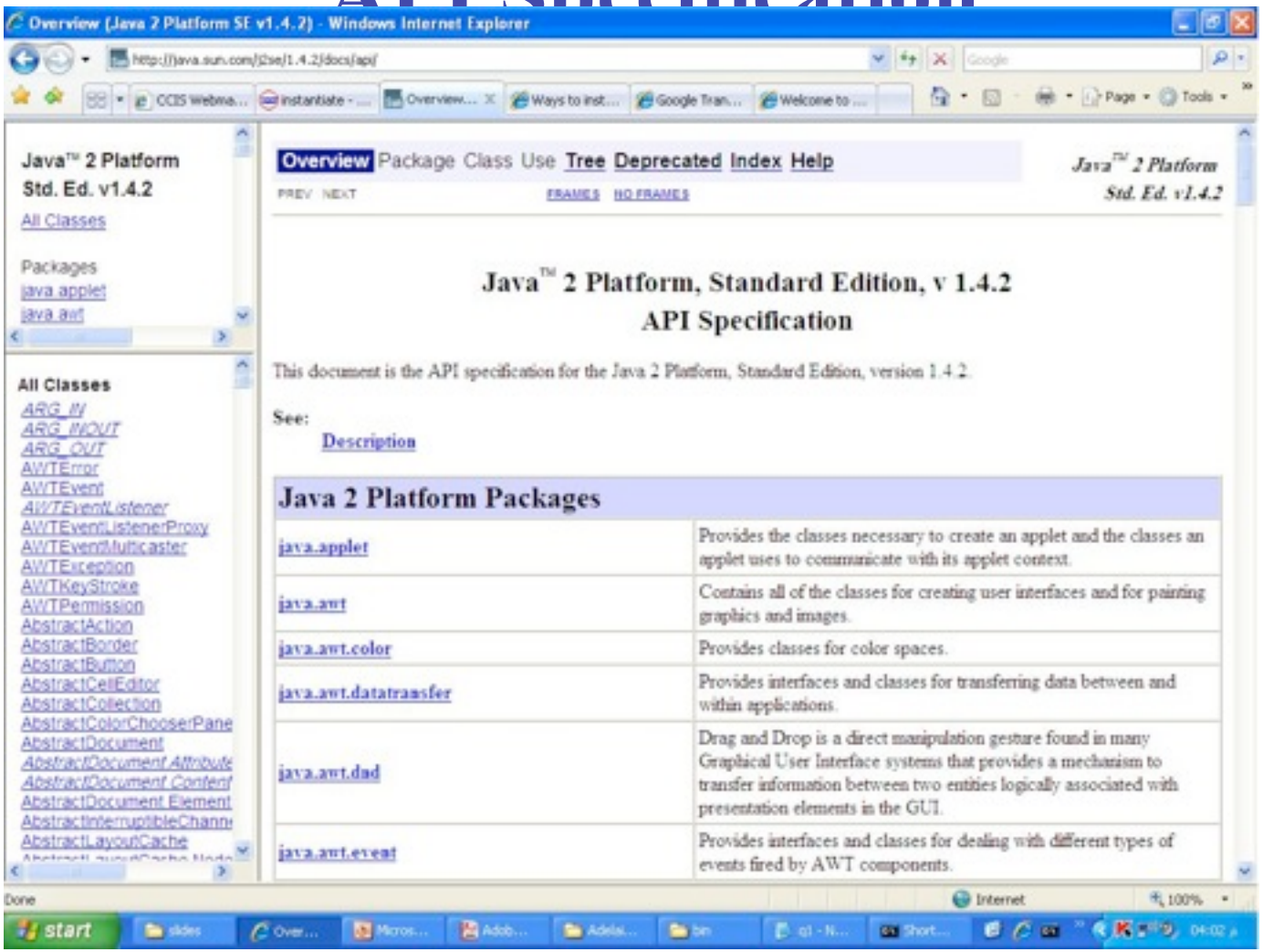
How do we find out what methods are available in the Math class?

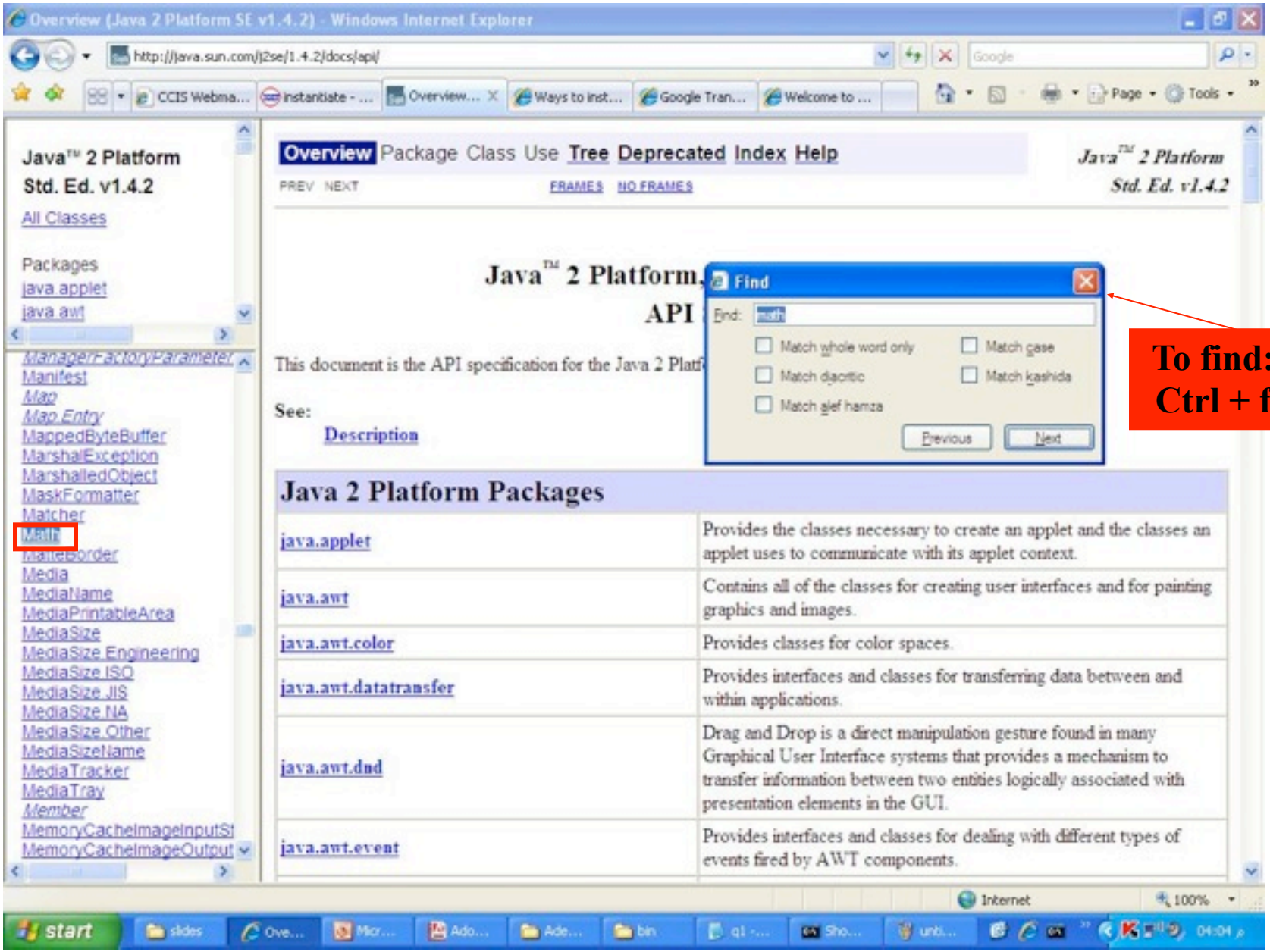
- There is on-line (web based) documentation for the Java libraries.

  http://java.sun.com/j2se/1.5.0/docs/api/

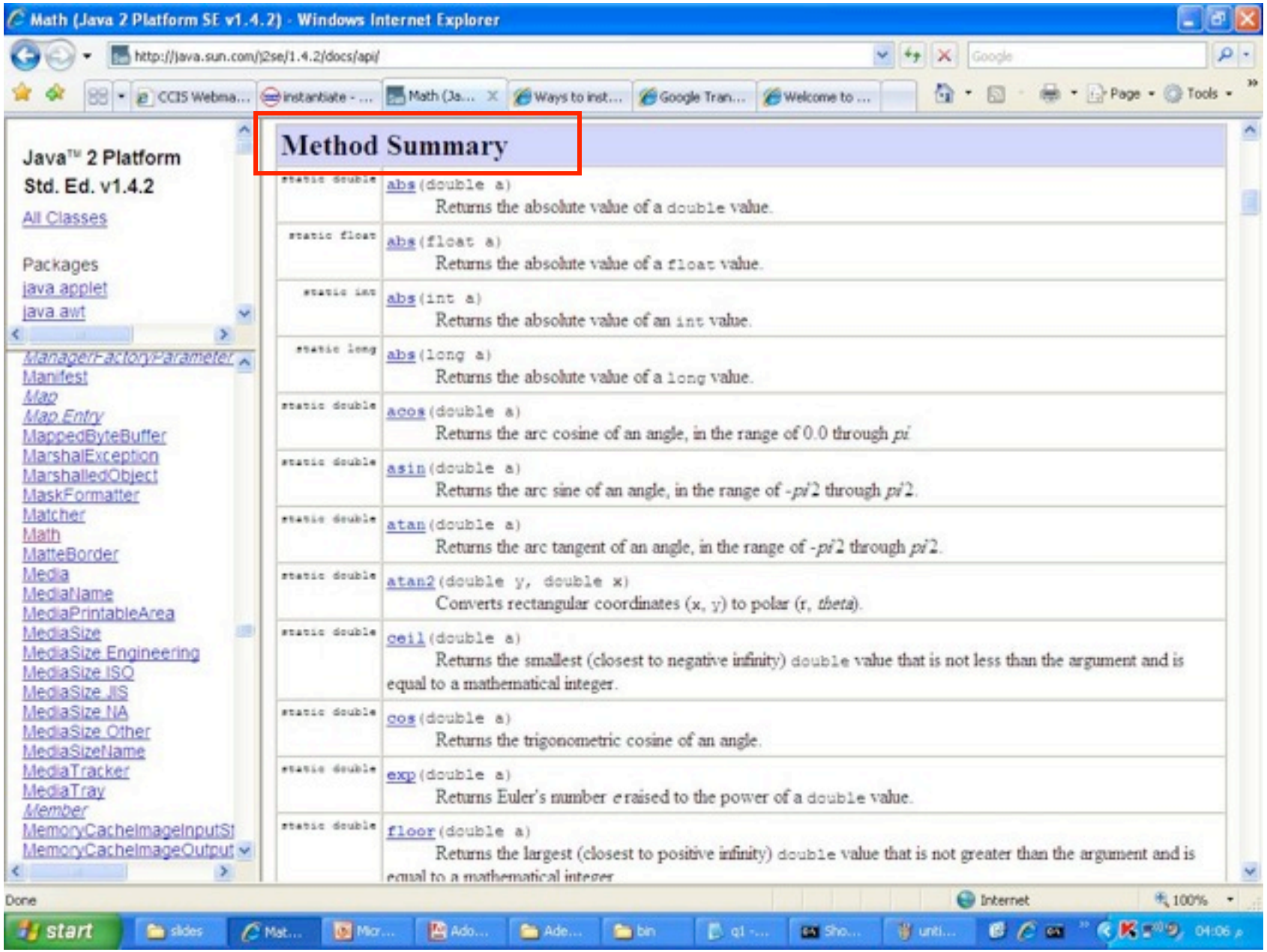
- You don't need to study the entire set of classes in the API.
- What it is useful for is:
  - Looking up the format for a method you already know exists
  - Finding a class that you think probably should exist. This will come with experience as you get used to the types of classes that are defined in libraries.

Java 2 Platform Standard Edition 5.0 API Specification




Overview (Java 2 Platform SE v1.4.2) - Windows Internet Explorer
http://java.sun.com/j2se/1.4.2/docs/api/
Java™ 2 Platform
Std. Ed. v1.4.2
All Classes
Packages
java.applet
java.awt
Overview Package Class Use Tree Deprecated Index Help
PREV NEXT
FRAMES NO FRAMES
Java™ 2 Platform Std. Ed. v1.4.2
Java™ 2 Platform,
API
This document is the API specification for the Java 2 Platf
See:
Description
Find
Find: math
Match whole word only
Match case
Match diacritic
Match kashida
Match alef hamza
Previous
Next
To find:
Ctrl + f
Java 2 Platform Packages
java.applet
Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt
Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color
Provides classes for color spaces.
java.awt.datatransfer
Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd
Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event
Provides interfaces and classes for dealing with different types of events fired by AWT components.
Manifest
Map
Map.Entry
MappedByteBuffer
MarshalException
MarshalledObject
MaskFormatter
Matcher
Math
MatteBorder
Media
MediaName
MediaPrintableArea
MediaSize
MediaSize.Engineering
MediaSize.ISO
MediaSize.JIS
MediaSize.NA
MediaSize.Other
MediaSizeName
MediaTracker
MediaTray
Member
Internet
100%
start

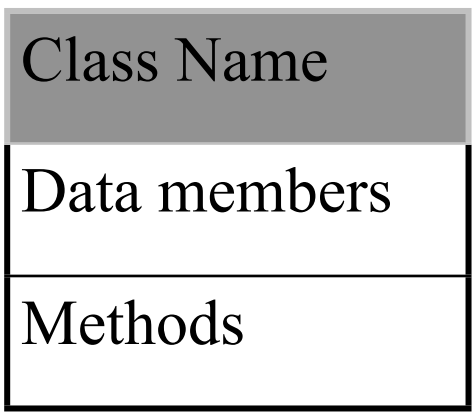

Math (Java 2 Platform SE v1.4.2) - Windows Internet Explorer
http://java.sun.com/j2se/1.4.2/docs/api/
Java™ 2 Platform
Std. Ed. v1.4.2
All Classes
Packages
java applet
java awt
Overview Package Class Use Tree Deprecated Index Help
PREV CLASS NEXT CLASS
FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD
DETAIL: FIELD | CONSTR | METHOD
Java™ 2 Platform
Std. Ed. v1.4.2
java.lang
Class Math
java.lang.Object
java.lang.Math
public final class Math
extends Object
The class Math contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.
Unlike some of the numeric methods of class StrictMath, all implementations of the equivalent functions of class Math are not defined to return the bit-for-bit same results. This relaxation permits better-performing implementations where strict reproducibility is not required.
By default many of the Math methods simply call the equivalent method in StrictMath for their implementation. Code generators are encouraged to use platform-specific native libraries or microprocessor instructions, where available, to provide higher-performance implementations of Math methods. Such higher-performance implementations still must conform to the specification for Math.
The quality of implementation specifications concern two properties, accuracy of the returned result and monotonicity of the method. Accuracy of the floating-point Math methods is measured in terms of ulps, units in the last place. For a given floating-point format, an ulp of a specific real number value is the difference between the two floating-point values closest to that
Done
Internet
100%
start


Math (Java 2 Platform SE v1.4.2) - Windows Internet Explorer
http://java.sun.com/j2se/1.4.2/docs/api/
Java™ 2 Platform
Std. Ed. v1.4.2
All Classes
Packages
java.applet
java.awt
Method Summary
abs(double a)
Returns the absolute value of a double value.
abs(float a)
Returns the absolute value of a float value.
abs(int a)
Returns the absolute value of an int value.
abs(long a)
Returns the absolute value of a long value.
acos(double a)
Returns the arc cosine of an angle, in the range of 0.0 through pi.
asin(double a)
Returns the arc sine of an angle, in the range of -pi/2 through pi/2.
atan(double a)
Returns the arc tangent of an angle, in the range of -pi/2 through pi/2.
atan2(double y, double x)
Converts rectangular coordinates (x, y) to polar (r, theta).
ceil(double a)
Returns the smallest (closest to negative infinity) double value that is not less than the argument and is equal to a mathematical integer.
cos(double a)
Returns the trigonometric cosine of an angle.
exp(double a)
Returns Euler's number e raised to the power of a double value.
floor(double a)
Returns the largest (closest to positive infinity) double value that is not greater than the argument and is
Done
Internet
100%
start


Math (Java 2 Platform SE v1.4.2) - Windows Internet Explorer
http://java.sun.com/j2se/1.4.2/docs/api/
Java™ 2 Platform
Std. Ed. v1.4.2
All Classes
Packages
java.applet
java.awt
ManagerFactoryParameter
Manifest
Map
Map.Entry
MappedByteBuffer
MarshalException
MarshalledObject
MaskFormatter
Matcher
Math
MatteBorder
Media
MediaName
MediaPrintableArea
MediaSize
MediaSize.Engineering
MediaSize.ISO
MediaSize.JIS
MediaSize.NA
MediaSize.Other
MediaSizeName
MediaTracker
MediaTray
Member
MemoryCacheImageInputSt
MemoryCacheImageOutput
static double IEEEremainder(double f1, double f2)
Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard.
static double log(double a)
Returns the natural logarithm (base e) of a double value.
static double max(double a, double b)
Returns the greater of two double values.
static float max(float a, float b)
Returns the greater of two float values.
static int max(int a, int b)
Returns the greater of two int values.
static long max(long a, long b)
Returns the greater of two long values.
static double min(double a, double b)
Returns the smaller of two double values.
static float min(float a, float b)
Returns the smaller of two float values.
static int min(int a, int b)
Returns the smaller of two int values.
static long min(long a, long b)
Returns the smaller of two long values.
static double pow(double a, double b)
Returns the value of the first argument raised to the power of the second argument.
static double random()
Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.
static double rint(double a)
Returns the double value that is closest in value to the argument and is equal to a mathematical integer.
Done
Internet
100%
start

Class scope

- **A class has :**
- A name
- variables or data members
- methods
- Members are accessible to all class methods
- example:

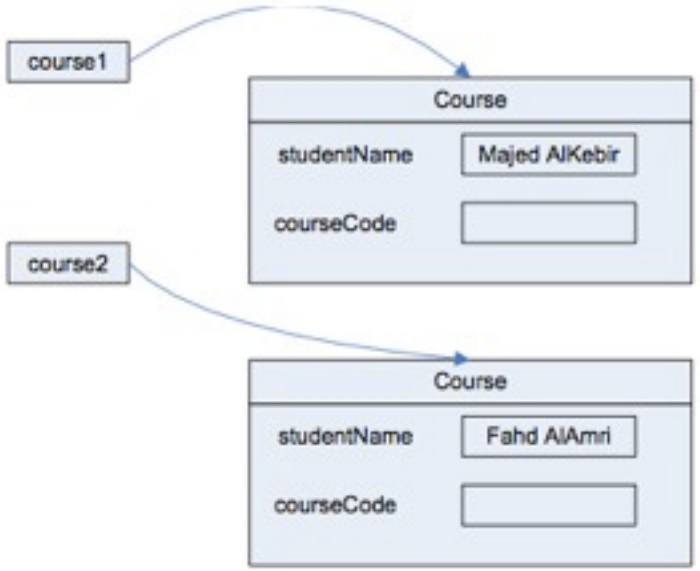


class examplr

```
class Course {
// Data Member
public String studentName;
public String courseCode ;
}
////////////////////////////////////
public class CourseRegistration {
public static void main(String[] args) {
Course course1, course2;

//Create and assign values to course1
course1 = new Course( );
course1.courseCode= new String("CSC112");
course1.studentName= new String("Majed AlKebir"); //Create and assign values to
course2
course2 = new Course( );
course2.courseCode= new String("CSC107");
course2.studentName= new String("Fahd AlAmri");

System.out.println(course1.studentName + " has the course "+
course1.courseCode);
System.out.println(course1.studentName + " has the course "+
course1.courseCode);
}
}
```



The `class String`

- String variables are reference variables.
- Given:

```
String name;
```

  - Equivalent statements:

```
name = new String("Lisa Johnson");
```

```
name = "Lisa Johnson";
```

The `class String`

- A `String` object is an instance of `class String`.
- A `String` object with the value `"Lisa Johnson"` is instantiated.
- The address of the object is stored in `name`.
- The `new` operator is unnecessary when instantiating Java strings.
- `String` methods are called using the dot operator.

The `class String`

- Java system automatically makes the class **String** available (i.e no need to import this class)
- **Example :**

  Consider the following declaration :

  ```
  String  sentence ;
  sentence = “programming with Java”
  ```

Some Commonly Used String Methods

Table 3-1 Some Commonly Used String Methods

```
String(String str)
  //Constructor: Creates a string object and initializes the string
  //object with characters specified by str.
  //Example:
  //String myStr = new String(sentence);
  //       myStr is a String variable initialized using sentence
```

```
char charAt(int index)
  //Returns the character at the position specified by index.
  //Example: sentence.charAt(3) returns 'g'
```

```
int indexOf(char ch)
  //Returns the index of the first occurrence of the character
  //specified by ch; if the character specified by ch does not
  //appear in the string, it returns -1.
  //Example: sentence.indexOf('J') returns 17
  //         sentence.indexOf('a') returns 5
```

Some Commonly Used String Methods

```
int indexOf(char ch, int pos)
  //Returns the index of the first occurrence of the character
  //specified by ch. The parameter pos specifies from where to begin
  //the search; if the character specified by ch does not
  //appear in the string, it returns -1.
  //Example: sentence.indexOf('a', 10) returns 18
```

```
int indexOf(String str)
  //Returns the index of the first occurrence of the string
  //specified by str; if the string specified by str does not
  //appear in the string, it returns -1.
  //Example: sentence.indexOf("with") returns 12
  //         sentence.indexOf("ing") returns 8
```

```
int indexOf(String str, int pos)
  //Returns the index of the first occurrence of the string
  //specified by str. The parameter pos specifies from where to begin
  //the search; if the string specified by str does not appear
  //in the string, it returns -1.
```

Some Commonly Used String Methods

Table 3-1 Commonly Used String Methods (continued)

```
String concat(String str)
  //Returns the string that is this string concatenated with str
  //Example: The expression
  //         sentence.concat(" is fun.")
  //         returns the string "Programming with Java is fun."
```

```
int length()
  //Returns the length of the string
  //Example: sentence.length() returns 21, the number of characters in
  //         "Programming with Java"
```

```
String replace(char charToBeReplaced, char charReplacedWith)
  //Returns the string in which every occurrence of
  //charToBeReplaced is replaced with charReplacedWith
  //Example: sentence.replace('a', '*') returns the string
  //         "Progr*mming with J*v*"
  //         Each occurrence of a is replaced with *
```

Some Commonly Used String Methods

```
String substring(int startIndex, int endIndex)
  //Returns the string which is a substring of this string
  //starting at startIndex until endIndex - 1.
```

```
String toLowerCase()
  //Returns the string that is same as this string except that
  //all uppercase letters of this string are replaced with
  //their equivalent lowercase letters.
  //Example: sentence.toLowerCase() returns "programming with java"
```

```
String toUpperCase()
  //Returns the string that is same as this string except
  //that all lowercase letters of this string are replaced with
  //their equivalent uppercase letters.
  //Example: sentence.toUpperCase() returns "PROGRAMMING WITH JAVA"
```

Examples on String methods

```
String s1;
s1 = "abcdefeg" ;

System.out.println( s1.length() );     // 8
System.out.println(s1.charAt(3));      // d
System.out.println(s1.indexOf('e'));   // 4
System.out.println(s1.indexOf("cd"));  // 2
System.out.println(s1.toUpperCase());  // ABCDEFEG
System.out.println(s1.indexOf('z'));   //-1
System.out.println(s1.charAt(20));     //Exception
                                       //out of range
```

More examples on String methods

```
String s1 ;
s1 = "abcdefeg" ;

System.out.println(s1.substring(1 , 4));   //bcd
System.out.println(s1.substring(7 , 8));  //g
System.out.println(s1 + "xyz");       //abcdefegxyz
System.out.println(s1.replace('d' ,'D')); //abcDefeg
System.out.println(s1.charAt(4) );        // e
System.out.println(s1.indexOf('b'));       // 1
System.out.println(s1.indexOf('e',5));    // 6
```

Go through Example 3-4 from the text book.

Input/Output

- Other ways to input data.
- Other ways to output results.
- Format output using method **`printf()`**
- Format output of decimal numbers to a specific numbers of decimal places .

Formatting Output with `printf`

- `System.out` → output object
  - `print`
  - `println`
- Both cannot format the output in a specific manner.
- For example: align the output in certain columns.
- `printf` → does that.

Formatting Output with `printf`

- The syntax to use the method `printf` to produce output on the standard output device is:

  ```
  System.out.printf(formatString);
  ```

  or

  ```
  System.out.printf(formatString,argumentList);
  ```

- `formatString` is a string specifying the format of the output.
- `argumentList` is a list of arguments that consists of constant values, variables, or expressions.
- If there is more than one argument in `argumentList`, the arguments are separated with commas.

Formatting Output with `printf`

- For example:
  - The statement: `System.out.printf("Hello there!");`

    Consists of only the `format string`
  - The statement:

```
System.out.printf("There are %.2f inches in %d centimeters.%n",
                  centimeters / 2.54, centimeters);
```

    Consists of both the `format string` and `argumentList`.

- `%.2f` and `%d` are called **format specifiers**.

- By default, there is a one-to-one correspondence between format specifiers and the arguments in `argumentList`.

Formatting Output with `printf`

```
System.out.printf("There are %.2f inches in %d centimeters.%n",
                  centimeters / 2.54, centimeters);
```

- The first format specifier, `%.2f`, is matched with the first argument, which is the expression `centimeters / 2.54`.
- The second format specifier, `%d`, is matched with the second argument, which is `centimeters`.
- The format specifier **`%n`** positions the insertion point at the beginning of the next line.
- If centimeters = 150 ➔ 150/2.54 =59.05511811023
- The o/p would be :

  There are **59.06** inches in **150** centimeters
- Note that the value of the expression `centimeters / 2.54` is rounded.

Formatting Output with `printf`

- A format specifier for general, character, and numeric types has the following syntax:

```
%[argument_index$][flags][width][.precision]conversion
```

- The expressions in square brackets are optional. That is, they may or may not appear in a format specifier.
- The optional *argument_index* is a (decimal) integer that indicates the position of the argument in the argument list. The first argument is referenced by "`1$`," the second by "`2$`," etc.
- The optional *flags* is a set of characters that modify the output format.
- The optional *width* is a (decimal) integer that indicates the minimum number of characters to be written to the output.
- The optional *precision* is a (decimal) integer that is usually used to restrict the number of characters.
- The required *conversion* is a character that indicates how the argument should be formatted.

Formatting Output with `printf`

Table 3-2 Some of the Supported Conversions

	Conversion	The result is
`'s'`	general	a string
`'c'`	character	a Unicode character
`'d'`	integral	formatted as a (decimal) integer
`'e'`	floating point	formatted as a decimal number in computerized scientific notation
`'f'`	floating point	formatted as a decimal number
`'%'`	percent	`'%'`
`'n'`	line separator	the platform-specific line separator

Formatting Output with `printf`

Example3_6

```
1.  public class Example3_6
2.  {
3.     public static void main (String[] args)
4.     {
5.        int num = 763;
6.        double x = 658.75;
7.        String str = "Java Program.";
8.
9.        System.out.println("12345678901234567890123456789O");
♦         System.out.printf("%5d%7.2f%15s%n",   num, x, str);
♦         System.out.printf("%15s%6d%9.2f %n",    str, num, x);
♦         System.out.printf("%8.2f%7d%15s %n", x, num, str);
♦         System.out.printf("num = %5d %n", num);
♦         System.out.printf("x = %10.2f %n", x);
♦         System.out.printf("str = %15s %n", str);
♦         System.out.printf("%10s%7d %n","Program No.", 4);
♦      }
♦   }
```

Formatting Output with `printf`

Example3_6

```
1234567890123456789012345678901234567890
  763 658.75  Java Program.
  Java Program.    763    658.75
  658.75    763  Java Program.
num =    763
x =      658.75
str =    Java Program.
Program No.      4
```

```
1.  public class Example3_6
2.  {
3.     public static void main (String[] arg
4.     {
5.        int num = 763;
6.        double x = 658.75;
7.        String str = "Java Program.";
8.
9.        System.out.println("12345678901234567890123456789012345678901234567890");
♦         System.out.printf("%5d%7.2f%15s%n",   num, x, str);
♦         System.out.printf("%15s%6d%9.2f %n",    str, num, x);
♦         System.out.printf("%8.2f%7d%15s %n", x, num, str);
♦         System.out.printf("num = %5d %n", num);
♦         System.out.printf("x = %10.2f %n", x);
♦         System.out.printf("str = %15s %n", str);
♦         System.out.printf("%10s%7d %n","Program No.", 4);
♦      }
♦   }
```

Formatting Output with `printf`

Example3_6

```
1234567890123456789012345678901234567890
  763 658.75  Java Program.
  Java Program.    763    658.75
  658.75     763  Java Program.
num =    763
x =      658.75
str =    Java Program.
Program No.      4
```

```
1.  public class Example3_6
2.  {
3.     public static void main (String[] arg
4.     {
5.        int num = 763;
6.        double x = 658.75;
7.        String str = "Java Program.";
8.
9.        System.out.println("1234567890123456789012345678901234567890");
♦         System.out.printf("%5d%7.2f%15s%n",   num, x, str);
♦         System.out.printf("%15s%6d%9.2f %n",    str, num, x);
♦         System.out.printf("%8.2f%7d%15s %n", x, num, str);
♦         System.out.printf("num = %5d %n", num);
♦         System.out.printf("x = %10.2f %n", x);
♦         System.out.printf("str = %15s %n", str);
♦         System.out.printf("%10s%7d %n","Program No.", 4);
♦      }
♦   }
```

7 → width
.2→ precision
f → conversion

Formatting Output with `printf`

- The output of a `printf` statement is right-justified by default.

- To force the output to be left-justified, you can use the format specifier `flag`. If `flag` is set to `'-'` (negative), then the output of the result is left justified.
- The following example clarifies this:

Formatting Output with `printf`

Example3_7

```
1.  public class Example3_7
2.  {
3.     public static void main (String[] args)
4.     {
5.        int num = 763;
6.       double x = 658.75;
7.        String str = "Java Program.";
8.
9.      System.out.println("12345678901234567890123456789O");
10.     System.out.printf("%-5d%-7.2f%-15s ***%n",    num, x, str);
11.     System.out.printf("%-15s%-6d%-    9.2f ***%n", str, num, x);
12.     System.out.printf("%-8.2f%-7d%-15s ***%n", x, num, str);
13.     System.out.printf("num = %-5d ***%n", num);
```

Formatting Output with `printf`

Example3_7

```
1234567890123456789012345678901234567890
763   658.75 Java Program.    ***
Java Program.   763    658.75     ***
658.75   763    Java Program.    ***
num = 763    ***
x = 658.75      ***
str = Java Program.    ***
Program No.4        ***
```

```
1. public class Example3_7
2. {
3.    public static void main (String[] a
4.    {
5.       int num = 763;
6.      double x = 658.75;
7.       String str = "Java Program.";
8.
9.      System.out.println("12345678901234567890123456789012345678901234567890");
10.     System.out.printf("%-5d%-7.2f%-15s ***%n",    num, x, str);
11.     System.out.printf("%-15s%-6d%-    9.2f ***%n", str, num, x);
12.     System.out.printf("%-8.2f%-7d%-15s ***%n", x, num, str);
13.     System.out.printf("num = %-5d ***%n", num);
```

Formatting Output with `printf`

Example3_7

```
1. public class Example3_7
2. {
3.    public static void main (String[] a
4.    {
5.       int num = 763;
6.      double x = 658.75;
7.       String str = "Java Program.";
8.
9.      System.out.println("12345678901234567890123456789O");
10.     System.out.printf("%-5d%-7.2f%-15s ***%n",    num, x, str);
11.     System.out.printf("%-15s%-6d%-    9.2f ***%n", str, num, x);
12.     System.out.printf("%-8.2f%-7d%-15s ***%n", x, num, str);
13.     System.out.printf("num = %-5d ***%n", num);
```

```
12345678901234567890123456789O
763   658.75 Java Program.    ***
Java Program.  763    658.75     ***
658.75  763    Java Program.    ***
num = 763    ***
x = 658.75       ***
str = Java Program.    ***
Program No.4         ***
```

- → flag
7 → width
.2→ precision
f → conversion

Parsing Numeric Strings

- A string consisting of only integers or decimal numbers is called a **numeric string**.

- To convert a string consisting of an integer to a value of the type `int`, we use the following expression:

```
Integer.parseInt(strExpression)
```

- **Example:**

```
Integer.parseInt("6723") = 6723
Integer.parseInt("-823") = -823
```

Parsing Numeric Strings

- To convert a string consisting of a decimal number to a value of the type `float`, we use the following expression:

```
Float.parseFloat(strExpression)
```

- **Example:**

```
Float.parseFloat("34.56") = 34.56
Float.parseFloat("-542.97") = -542.97
```

- To convert a string consisting of a decimal number to a value of the type `double`, we use the following expression:

```
Double.parseDouble(strExpression)
```

- **Example:**

```
Double.parseDouble("345.78") = 345.78
Double.parseDouble("-782.873") = -782.873
```

Formatting the Output Using the `String` Method `format`

- `printf` cannot be used with output dialog boxes.

- Two other ways:
  1. Use the `String` method `format`. ➔ our interest
  2. Use the **`class`** `DecimalFormat`. ➔ Appendix D

Formatting the Output Using the `String` Method `format`

Example 3-13

```
double x = 15.674;
double y = 235.73;
double z = 9525.9864;
int num = 83;
String str;
```

Expression	Value
`String.format("%.2f", x)`	`"15.67"`
`String.format("%.3f", y)`	`"235.730"`
`String.format("%.2f", z)`	`"9525.99"`
`String.format("%7s", "Hello")`	`"  Hello"`
`String.format("%5d%7.2f", num, x)`	`"   83  15.67"`
`String.format("The value of num = %5d", num)`	`"The value of num =    83"`
`str = String.format("%.2f", z)`	`str = "9525.99"`

Example3_15

```
The value of x with two decimal places = 15.67
The value of y with two decimal places =235.73
The value of z with two decimal places = 9525.99
```

```
import java.util.*;

public class Example3_15{
 public static void main(String[] args) {
   double x = 15.674;
   double y = 235.73;
   double z = 9525.9864;
   String str;

  str = String.format("The value of x with two decimal places = %.2f%n", x)
       + String.format("The value of y with two decimal places = %.2f%n", y)
       + String.format("The value of z with two decimal places = %.2f%n", z);


   System.out.println( str );

  }
}
```

Chapter Summary

- Primitive type variables store data into their memory space.
- Reference variables store the address of the object containing the data.
- An object is an instance of a class.
- Operator new is used to instantiate an object.
- Garbage collection reclaims memory that is not being used.

Chapter Summary

- To use a predefined method, you must know its name and the class and package it belongs to.
- The dot (.) operator is used to access a certain method in a class.
- Methods of the `class String` are used to manipulate input and output data.
- Dialog boxes can be used to input data and output results.
- Data can be formatted using the `String` method `format`.