PREDEFINED METHODS

From “Java Programming...” D.S. Malik – Ch7

Outline

1. Introduction
2. Mathematical Methods
3. Character Manipulation Methods
4. import static Statements
5. Parsing Methods

Pre-defined methods

- A predefined method is an existing method that is written and provided by Java.
- In Java, predefined methods are organized as a collection of classes, called class libraries.
- Class libraries are stored in packages.
- The programmer can use a predefined method immediately after importing the relevant package.

- Example:
  - `nextInt()`, `nextDouble()`, etc… with `Scanners`
  - `equals(…)`, `toUpperCase()`, etc… with `Strings`

Pre-defined methods

- We have seen methods like these
  - `nextInt()`, `nextDouble()`, etc… with `Scanners`
  - `equals(…)`, `toUpperCase()`, etc… with `Strings`

- They were used with objects of those classes
- Syntax of a call:

```
objectName.methodName(parameters)
```

Pre-defined methods

- We will see methods like these
  - `pow(3,2), cos(theta),` etc… for class `Math`
  - `isDigit(ch), toUpperCase(ch),` etc… for class `Character`
- These do NOT need an object of those classes
- Syntax of a call:

```
className.methodName(parameters)
```

`import` statements

- All methods in this lecture belong to the package `java.lang`
- You have seen import statements like this:

```
import java.util.*;
```

- We can have import statements like this:

```
import java.lang.Math.*;
```

- But we will also see statements like this:

```
import static java.lang.Math.* ;
```

The `Math` Class

- Provides many standard mathematical constants and methods:

```
double E = 2.7182818283590455;
double PI = 3.141592653689793;
```

Name	Description	Argument Type	Return Type	Example	Value Returned
pow	Power	double	double	Math.pow(2.0,3.0)	8.0
abs	Absolute value	int, long, float, or double	Same as the type of the argument	Math.abs(-7) Math.abs(7) Math.abs(-3.5)	7 7 3.5
max	Maximum	int, long, float, or double	Same as the type of the arguments	Math.max(5, 6) Math.max(5.5, 5.3)	6 5.5

The `Math` Class

Name	Description	Argument Type	Return Type	Example	Value Returned
`min`	Minimum	`int`, `long`, `float`, or `double`	Same as the type of the arguments	`Math.min(5, 6)` `Math.min(5.5, 5.3)`	5 5.3
`round`	Rounding	`float` or `double`	`int` or `long`, respectively	`Math.round(6.2)` `Math.round(6.8)`	6 7
`ceil`	Ceiling	`double`	`double`	`Math.ceil(3.2)` `Math.ceil(3.9)`	4.0 4.0
`floor`	Floor	`double`	`double`	`Math.floor(3.2)` `Math.floor(3.9)`	3.0 3.0
`sqrt`	Square root	`double`	`double`	`sqrt(4.0)`	2.0

The `Math` Class

Name	Description	Argument Type	Return Type	Example	Value returned
`cos(x)`	Returns the cosine of x (x is in radians)	`double`	`double`	`Math.cos(0.0)`	1.0
`sin(x)`	Returns the sine of x (x is in radians)	`double`	`double`	`Math.sin(0.0)`	0.0
`tan(x)`	Returns the tangent of x (x is in radians)	`double`	`double`	`Math.tan(0.0)`	0.0
`exp(x)`	Returns the value of e^x; where e is approximately 2.718.	`double`	`double`	`Math.exp(3.0);`	20.086
`log(x)`	Returns the natural logarithm (base E) of x	`double`	`double`	`Math.log( 2.00)`	0.693
`log10(x)`	Returns the base 10 logarithm of x	`double`	`double`	`Math.log10 (1000.00)`	3.00

The `Math.random()` method

- `Math.random()` returns a random `double` that is greater than or equal to zero and less than 1, i.e., `[0,1[`
- You can scale the result using addition and multiplication
- Example:
  - the following simulates rolling a six sided die

```
int die = (int) (6.0 * Math.random())+1;
```

The Character Class

Name	Description	Argument Type	Return Type	Examples	Return Value
toUpperCase	Convert to uppercase	char	char	Character.toUpperCase('a') Character.toUpperCase('A')	'A' 'A'
toLowerCase	Convert to lowercase	char	char	Character.toLowerCase('a') Character.toLowerCase('A')	'a' 'a'
isUpperCase	Test for uppercase	char	boolean	Character.isUpperCase('A') Character.isUpperCase('a')	true false
isLowerCase	Test for lowercase	char	boolean	Character.isLowerCase('A') Character.isLowerCase('a')	false true
isLetter	Test for a letter	char	boolean	Character.isLetter('A') Character.isLetter('%')	true false
isDigit	Test for a digit	char	boolean	Character.isDigit('5') Character.isDigit('A')	true false
isWhitespace	Test for whitespace	char	boolean	Character.isWhitespace(' ') Character.isWhitespace('A')	true false

Whitespace characters are those that print as white space, such as the blank, the tab character ('\t'), and the line-break character ('\n').

Example 1

```
// assume you did: import java.lang.Character.*;
String str = "I got\t30 cats.\n";
char oneChar;
int i, ws=0, digits=0;

for(i=0; i < str.length(); i++)
{   oneChar = str.charAt(i);
    if (Character.isWhitespace(oneChar))
        ws++;
    else if (Character.isDigit(oneChar))
        digits++;
}
System.out.println("the string: \"" + str +"\" has ws= " +
                    ws + " digits= " + digits);
System.out.println(Character.toUpperCase(str.charAt(2)));
System.out.println(str.toUpperCase());
```

Example 2

- Write a complete program that computes the length of a side of a triangle using the following formula: $a^2 = b^2 + c^2 - 2bc\ cos\theta$
- The lengths of the sides b and c are given in cm.
- The angle θ is given in degrees.

`import static` Statement

- For simplification, Java 5.0 introduces the following import statement:

```
import static java.packageName.ClassName.*;
```

- After you include this statement in the program, you can omit the name of the class and the dot operator when calling a method.

`import static` Statement

- For example:
  - If you include:

```
import static java.lang.Math.*;
```

  - you can simply call `abs(x)`
  - instead of `Math.abs(x)`

  - If you include:

```
import static java.lang.Character.*;
```

  - you can simply call `toLowerCase(ch)`
  - instead of `Character.toLowerCase(ch)`

import VS import static

```
1  import java.lang.Math.*;          //java.lang is the package;
2                                    //Math is the class
3  public class MathTest1
4  { public static void main (String[] args) {
5    int rand = (int)(Math.random()*10);
6    double c60 = Math.cos(Math.PI/3);
7    System.out.printf ("Random = %d and cos(60)= %f", rand, c60);
8  }}
```

0 - 9

What is the range produced?

```
1  import static java.lang.Math.*; //java.la        package;
2                                   // Math
3  public class MathTest2
4  { public static void main (String[] args) {
5    double rand = (int)(random()*10);
6    double c60 = cos(PI/3);
7    System.out.printf ("Random = %d and cos(60)= %f", rand, c60);
8  }}
```

import static means you may omit class Name and dot

3. PARSING NUMERIC STRINGS

- A string that consists of only an integer or a floating-point number is called a numeric string.
- A numeric string can contain a minus sign.
- Examples of numeric strings include:
  - "2015"
  - "-20"
  - "144.45"
  - "-53.99"
- In order to process numeric strings as numbers (addition, multiplication, etc…), they must be converted first into numbers.

3. PARSING NUMERIC STRINGS

- Assume we have:
  String strExp1 = "2015", strExp2 = "-20", strExp3 = "144.45";

- Integer.parseInt (strExpression) converts a numeric string to an int.
  - Integer.parseInt (strExp1) ➔ 2015
  - Integer.parseInt (strExp2) ➔ -20

- Float.parseFloat (strExpression) converts a numeric string to a float.
  - Float.parseFloat (strExp2) ➔ -20.0 (as a float)
  - Float.parseFloat (strExp3) ➔ 144.45 (as a float)

- Double.parseDouble (strExpression) converts a numeric string to a double.
  - Double.parseDouble (strExp2) ➔ -20.0 (as a double)
  - Double.parseDouble (strExp3) ➔ 144.45 (as a double)

Parsing Example

In this example we will read a line from the user in the format: name age height and then divide the whole string into three variables:

name of type String,

age of type Integer,

and height of type double.

Parsing Example

```
import java.util.*;
public class HelloWorld {
public static void main(String []args){
  Scanner input = new Scanner(System.in);
  System.out.println("Enter your name, age, and height: ");
  String str = input.nextLine();
  int first_space = str.indexOf(" ");
  int second_space = str.indexOf(" ",first_space+1);
  String name = str.substring(0,first_space);
  int year =
      Integer.parseInt(str.substring(first_space+1,second_space));
  double height =
        Double.parseDouble(str.substring(second_space+1));
  System.out.println(name + " will be " + (height+2.0)+
                     " tall when he turns " +(year+1));
} // end main
} // end class
```

Self-Check Exercises

- Use the Java built-in methods to compute the roots of a quadratic equation in x of the form: $ax^2 + bx + c = 0$.

- The two roots are defined as:

$$\text{root1} = \frac{-b+\sqrt{b^2-4ac}}{2a}$$

$$\text{root2} = \frac{-b-\sqrt{b^2-4ac}}{2a}$$