



Java Foundations

4-5

The Math Class



Objectives

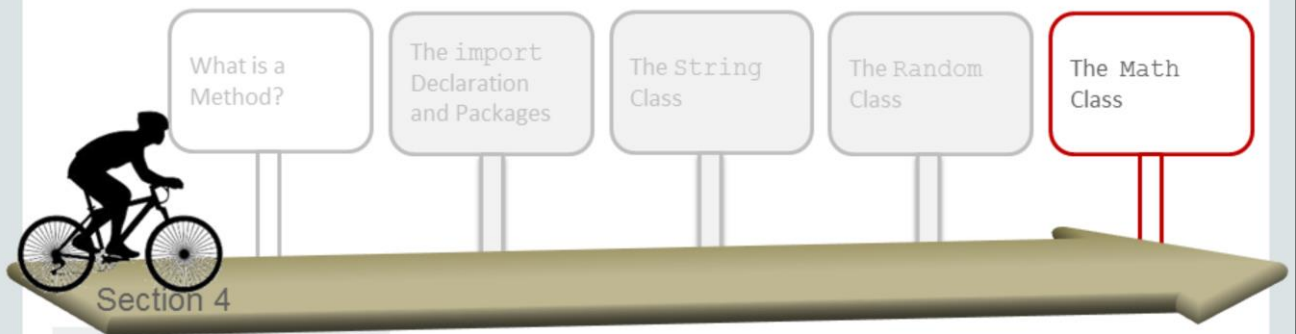
This lesson covers the following objective:

- Understand the methods of the `Math` class
- Use methods of the `Math` class to perform mathematical calculations
- Use fields of the `Math` Class



Topics

- Getting Started with Math Class
- Using methods of Math Class
- Using fields of Math Class



Performing Mathematical Calculations

- While developing programs, you may need more advanced mathematical calculations than what the basic Java math operators provide.
- For example:
 - Finding the maximum or minimum of two values
 - Rounding values
 - Logarithmic functions
 - Square root
 - Trigonometric functions
- The Java `Math` class contains methods for performing mathematical calculations.

The Math Class

- Is one of the many classes included in the Java class libraries.
- Contains methods that perform various mathematical functions.
- Is part of the `java.lang` package.

Documentation for the Math Class

You can access the documentation from here:

<http://docs.oracle.com/javase/8/docs/api/index.html>

The screenshot shows the Oracle Java Platform documentation for the `Math` class. The left sidebar lists various Java packages and classes, with `java.lang.Math` selected. The main panel displays the class documentation, including the class name, package, and a list of methods. A callout box points to the 'Field Summary' section, which lists the fields of the class.

Overview Packages **Class** Use Tree Deprecated Index Help

java.lang
Class Math
java.lang.Object
java.lang.Math

public final class Math
extends java.lang.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 `BigDecimal`, all implementations of the equivalent functions of class `Math` are not defined to return the bit-for-bit same result. 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 *ulp*, units in the last place. For a given floating-point format, an *ulp* of a specific real number value is the distance between the two floating-point values bracketing that numerical value. When discussing the accuracy of a method as a whole rather than at a specific argument, the number of *ulps* cited is for the worst-case error at any argument. If a method always has an error less than 0.5 *ulps*, the method always returns the floating-point number nearest the exact result, such a method is correctly rounded. A correctly rounded method is generally the best floating-point approximation can be, however, it is impractical for many floating-point methods to be correctly rounded. Instead, for the `Math` class, a larger error bound of 1 or 2 *ulps* is allowed for certain methods. Informally, with a 1 *ulp* error bound, when the exact result is a representable number, the exact result should be returned as the computed result; otherwise, either of the two floating-point values which bracket the exact result may be returned. For exact results large in magnitude, one of the endpoints of the bracket may be infinite. Besides accuracy at individual arguments, maintaining proper relations between the method at different arguments is also important. Therefore, most methods with more than 0.5 *ulp* errors are required to be *monotonic*, whenever the mathematical function is non-decreasing, so is the floating-point approximation; likewise, whenever the mathematical function is non-increasing, so is the floating-point approximation. Not all approximations that have 1 *ulp* accuracy will automatically meet the monotonicity requirement.

Since:
JDK1.0

Field Summary

Field	Field and Description
public static double	E

Scroll to see a list of fields and methods available in this class.

Scroll down the lower-left panel and click the Math link to display the documentation in the main panel on the right.



Exercise 1

- Examine the `Math` class documentation:
 - Standard Edition for Java SE 8:
<http://docs.oracle.com/javase/8/docs/api/>
 - See if you can find a value for `PI` and a method for computing the square root of a number.

Topics

- Getting Started with Math Class
- Using methods of Math Class
- Using fields of Math Class

What is a
Method?

The import
Declaration
and Packages

The String
Class

The Random
Class

The Math
Class



Section 4

ORACLE ACADEMY

JFo 4-5
The Math Class

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

9

Some of the Methods Available in Math Class

Method Name	Description
<code>abs(value)</code>	absolute value
<code>ceil(value)</code>	rounds up
<code>cos(value)</code>	cosine, in radians
<code>floor(value)</code>	rounds down
<code>log(value)</code>	logarithm base e
<code>log10(value)</code>	logarithm base 10
<code>max(value1, value2)</code>	larger of two values
<code>min(value1, value2)</code>	smaller of two values
<code>pow(base, exponent)</code>	<i>base</i> to the <i>exponent</i> power
<code>random()</code>	random double between 0 and 1
<code>round(value)</code>	nearest whole number
<code>sin(value)</code>	sine, in radians
<code>sqrt(value)</code>	square root

What's Different About the `Math` Class?

- The methods of the `Math` class are static methods.
- Static methods can be invoked through the class name.
- That means you don't have to create an object of the `Math` class to call the methods.
- For example, to invoke the methods of the `Random` class, you have to create an object of the `Random` class like this:

```
Random rndNum = new Random();  
int randomNum = rndNum.nextInt();
```

How Do You Call the Methods of the Math Class?

You can call methods of the `Math` class without creating an instance of the `Math` class, like this:

- Syntax:

- `Math.methodName (parameters)`

- Example:

- `Math.sqrt (121.0) ;`



Call methods by prefacing them with Math dot operator.

Calling a Method and Observing Its Result

- Let's see an example of calling a method and observing its result:

```
public static void main(String[] args) {  
  
    Math.sqrt(121.0);  
  
}
```

- Observe the output:
 - No output is displayed.
 - Simply calling these methods produces no visible result.

How Do the Methods of the `Math` Class Work?

- The `Math` methods don't print the results to the console.
- Each method returns a numerical result.
- The returning value is more flexible than printing.
- You can store, print, or combine it with a larger expression.

Storing and Printing the Results

- To see the result, you must print it or store it in a variable. For example:
- Print the result:

```
public static void main(String[] args) {  
    System.out.println("Square root: " + Math.sqrt(121.0)); //11.0  
}
```

- Store the value:

```
public static void main(String[] args) {  
    double sqroot= Math.sqrt(121.0);  
    System.out.println("Square root: "+sqroot); //11.0  
}
```

Combining the Results

You can combine the results and use it in a larger expression, like this:

```
public static void main(String[] args) {  
    double result = Math.min(3, 7) + Math.abs(-50);  
    System.out.println("Result is " + result); //53  
}
```




Exercise 2

- On paper, evaluate the following Java statements and record the results:
 - `Math.abs(-1.23)`
 - `Math.pow(3, 2)`
 - `Math.sqrt(121.0) - Math.sqrt(256.0)`
 - `Math.abs(Math.min(-3, -5))`



Exercise 3

- Consider an integer variable named `age`.
- Use `Math.max` and `Math.min` methods to answer the following questions:
 - What expression would replace negative ages with 0?
 - What expression would limit the maximum age to 40?

Answer:

What expression would replace negative ages with 0?

```
Math.max(age, 0)
```

What expression would limit the maximum age to 40?

```
Math.min(age, 40)
```

Topics

- Getting Started with Math Class
- Using methods of Math Class
- Using fields of Math Class

What is a
Method?

The import
Declaration
and Packages

The String
Class

The Random
Class

The Math
Class



Section 4

ORACLE ACADEMY

JFo 4-5
The Math Class

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

19

Fields in the Math Class

The `Math` class contains two constant fields: `PI` and `E`

Field	Description
<code>Math.E</code>	2.7182818...
<code>Math.PI</code>	3.1415926...

PI Field

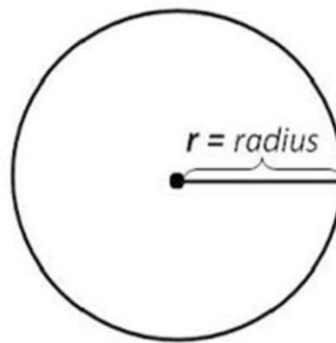


- The `Math` class contains a constant, `PI`.
- It contains a double value: 3.14159265358979323846.
- Remember, `Math` class methods are static methods and are accessed by using the `Math` class name.
- Similarly, `PI` is a static variable in the `Math` class, and it is accessed by using the `Math` class name.
- To use `PI` in a program, specify the class name (`Math`) and `PI`, separated by the dot operator:

–`Math.PI`

Calculating the Area of a Circle

- Suppose that you have to write a Java program to compute the area of a circle.
- Here's the formula to compute the area of a circle:
 - $\text{Area} = \text{PI} * \text{radius} * \text{radius}$
 - Where PI is a constant (approximately 3.1416)



Computing the Area of a Circle

Using the `Math.PI` field for calculating the area yields a more accurate result than using a constant value for pi like 3.14.

```
public class AreaOfCircle {  
    public static void main(String args[]) {  
        Scanner sc = new Scanner(System.in);  
        System.out.print("Enter the radius: ");  
        double radius = sc.nextDouble();  
        double area = Math.PI * radius * radius;  
        System.out.println("The area of circle is: " + area);  
    }  
}
```

Output:

Enter the radius: 7.5

The area of circle is: 176.71458676442586



Exercise 4

- A person's body mass index (BMI) is computed like this:

$$BMI = \frac{weight}{height^2} \times 703$$

- Import and open the `MathEx` project.
- Examine `ComputeBMI.java`.
- Write a program that computes the BMI and rounds off the BMI.



Exercise 4



- Use the methods of the `Math` class and display the output as:
 - Enter the weight in pounds: 132.5
 - Enter the height in inches: 62.5
 - Your Body Mass Index is 24



Summary

In this lesson, you should have learned how to:

- Use methods of the `Math` class to perform mathematical calculations
- Use fields of the `Math Class`



