Java Lec 4 Final Math

Monday, August 23, 2021 9:01 PM


Java_Fina...

Programming using Java

Java: final, static

Math class

Constants: `final`

- A `final` variable is a constant
- Once its value has been set, it cannot be changed
- Named constants make programs easier to read and maintain

dont allow changes.

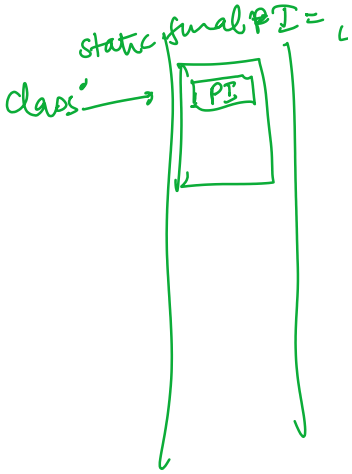
- Convention: use all-uppercase names for constants

Constants: `final`

```
final double QUARTER_VALUE = 0.25;
final double DIME_VALUE = 0.1;
final double NICKEL_VALUE = 0.05;
final double PENNY_VALUE = 0.01;
payment = dollars + quarters * QUARTER_VALUE
   + dimes * DIME_VALUE
   + nickels * NICKEL_VALUE
   + pennies * PENNY_VALUE;
```

Constants: `static final`

- If constant values are needed in several methods, declare them together with the instance fields of a class and tag them as `static` and `final`
- Give `static final` constants public



- Give double final constants public access to enable other classes to use them

Constants: `static final`

```
public class Math
{
   . . .
   public static final double E =
              2.7182818284590452354;
   public static final double PI =
              3.14159265358979323846;
}


double circumference = Math.PI * diameter;
```

Class

always use this

→ Math useful value methods.

Constant Definition

In a method:

```
final typeName variableName = expression ;
```

In a class:

```
accessSpecifier static final
          typeName variableName = expression;
```

private.

Access spe

Example:

```
final double NICKEL_VALUE = 0.05;
public static final double
       LITERS_PER_GALLON = 3.785;
```

Purpose:

To define a constant in a method or a class

Self Check

- What is the difference between the following two statements?

```
final double CM_PER_INCH = 2.54;
```

and

```
public static final double CM_PER_INCH = 2.54;
```

- What could go wrong with the following statement?

```
double circumference = 3.14 * diameter;
```

Answers

- The first definition is used inside a method, the second inside a class
- (1) You should use a named constant, not the "magic number" 3.14

the "magic number" 3.14
(2) 3.14 is not an accurate representation of π

default

The `Math` class

- Math class: contains methods like `sqrt` and `pow`
- To compute x^n, you write `Math.pow(x, n)`
- However, to compute x^2 it is significantly more efficient simply to compute `x * x`
- To take the square root of a number, use the `Math.sqrt`; for example, `Math.sqrt(x)`

The Math class

- In Java,

$$\frac{-b + \sqrt{b^2 - 4ac}}{}$$

$$2a$$

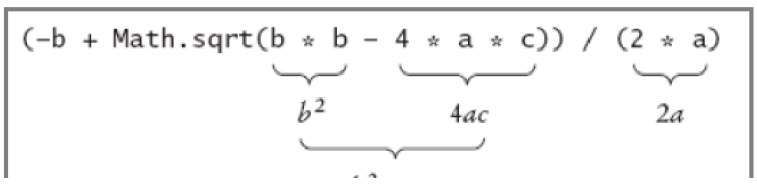
can be represented as

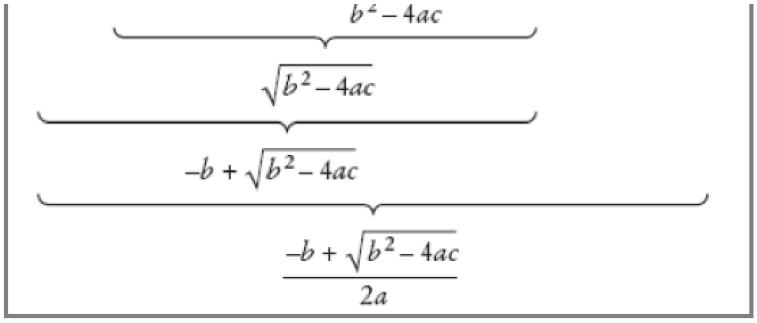
```
(-b + Math.sqrt(b * b - 4 * a * c)) / (2 * a)
```

Mathematical Methods in Java

`Math.sqrt(x)`	square root
`Math.pow(x, y)`	power x^y
`Math.exp(x)`	e^x
`Math.log(x)`	natural log
`Math.sin(x)`, `Math.cos(x)`, `Math.tan(x)`	sine, cosine, tangent (*x* in radian)
`Math.round(x)`	closest integer to *x*
`Math.min(x, y)`, `Math.max(x, y)`	minimum, maximum

Analyzing an Expression





Analyzing an Expression

Self Check

- What is the value of `1729 / 100`?
  Of `1729 % 100`?
- Why doesn't the following statement compute the average of `s1`, `s2`, and `s3`?

```
double average = s1 + s2 + s3 / 3; // Erro
```

- What is the value of

```
Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2))
```

in mathematical notation?

Answers

- 17 and 29
- Only `s3` is divided by 3. To get the correct

correct result, use parentheses. Moreover, if `s1`, `s2`, and `s3` are integers, you must divide by 3.0 to avoid integer division:

```
(s1 + s2 + s3) / 3.0
```

$$\sqrt{x^2 + y^2}$$

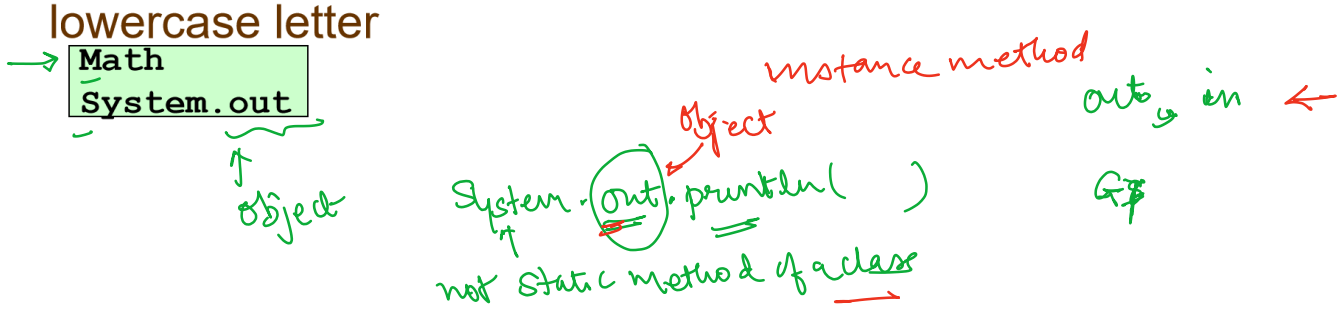
Calling `Static` Methods

- A `static` method does not operate on an object

```
double x = 4;
double root = x.sqrt(); // Error
```

- Static methods are defined inside classes
- Naming convention: Classes start with an uppercase letter; objects start with a lowercase letter

```
Math
System.out
```



Static Method Call

```
ClassName. methodName(parameters)
```

Example:

```
Math.sqrt(4)
```

Purpose:

To invoke a static method (a method that does not operate on an object)
and supply its parameters

Self Check

- Why can't you call `x.pow(y)` to compute x^y?
- Is the call `System.out.println(4)` a static method call?

Answers

- `x` is a number, not an object, and you cannot invoke methods on numbers

- No–the `println` method is called on the object `System.out`