3. Numerical Data

20 Aug 2015

Objectives

- Select proper types for numerical data.
- Write arithmetic expressions in Java.
- Evaluate arithmetic expressions, following the precedence rules.
- Describe how the memory allocation works for objects and primitive data values.
- Write mathematical expressions, using methods in the Math class.

Objectives

- Use the GregorianCalendar class in manipulating date information such as year, month, and day.
- Use the DecimalFormat class to format numerical data.
- Convert input string values to numerical data.
- Input numerical data by using System.in and output numerical data by using System.out.
- Apply the incremental development technique in writing programs.

Variables

- There are six numerical data types in Java (order by precision):
  1. byte (1 byte): integer
  2. short (2 bytes): integer
  3. int (4 bytes): integer
  4. long (8 bytes): integer
  5. float (4 bytes): floating-point
  6. double (8 bytes): floating-point

Variables

Data Type	Content	Default Value†	Minimum Value	Maximum Value
`byte`	Integer	0	−128	127
`short`	Integer	0	−32768	32767
`int`	Integer	0	−2147483648	2147483647
`long`	Integer	0	−9223372036854775808	9223372036854775807
`float`	Real	0.0	−3.40282347E+38‡	3.40282347E+38
`double`	Real	0.0	−1.79769313486231570E+308	1.79769313486231570E+308

Variables

- Declaring Variables: Declaring variable name and its data type using this syntax

  <data type> <variables>;

Here is an example of declaring variables of different data types:

```
int       i, j, k;
float     numberOne, numberTwo;
long      bigInteger;
double    bigNumber;
```

```
int count = 10, height = 34;
```

Variables

- Assignment Statement: We assign a value to a variable by using this syntax

  <variable> = <expression>;

```
firstNumber = 234;
```

```
sum       = firstNumber + secondNumber;
solution  = x * x - 2 * x + 1;
average   = (x + y + z) / 3.0;
```

```
4 + 5 = x;
x + y = y + x;
```

Bad Version

Variables

- Integer Assignment:
  - An integer literal is of type long if it ends with the letter L or l; otherwise it is of type int.
    - long: 100l, 100L
    - int: 100

    - valid: int x = 100 long x = 100 long x = 100l
      long x = 100L short x = 100 byte x = 100

    - invalid: int x = 100l int x = 100L short x = 100000
      byte x = 1000

Variables

- Floating-Point Assignment:
  - A floating-point literal is of type float if it ends with the letter F or f; otherwise its type is double and it can optionally end with the letter D or d.
  - We also can express float and double literal constants in scientific notation as Number * 10^exponent (eg. 4 * 10^6) which in Java is expressed as

    <number> E <exponent> or

    <number> e <exponent>

Variables

- double: 2.45, 2.45d, 2.45D, 4e4 (4 * 10^4)
- float: 4.5f, 4.5F, 4e4f, 4E4F

- valid:

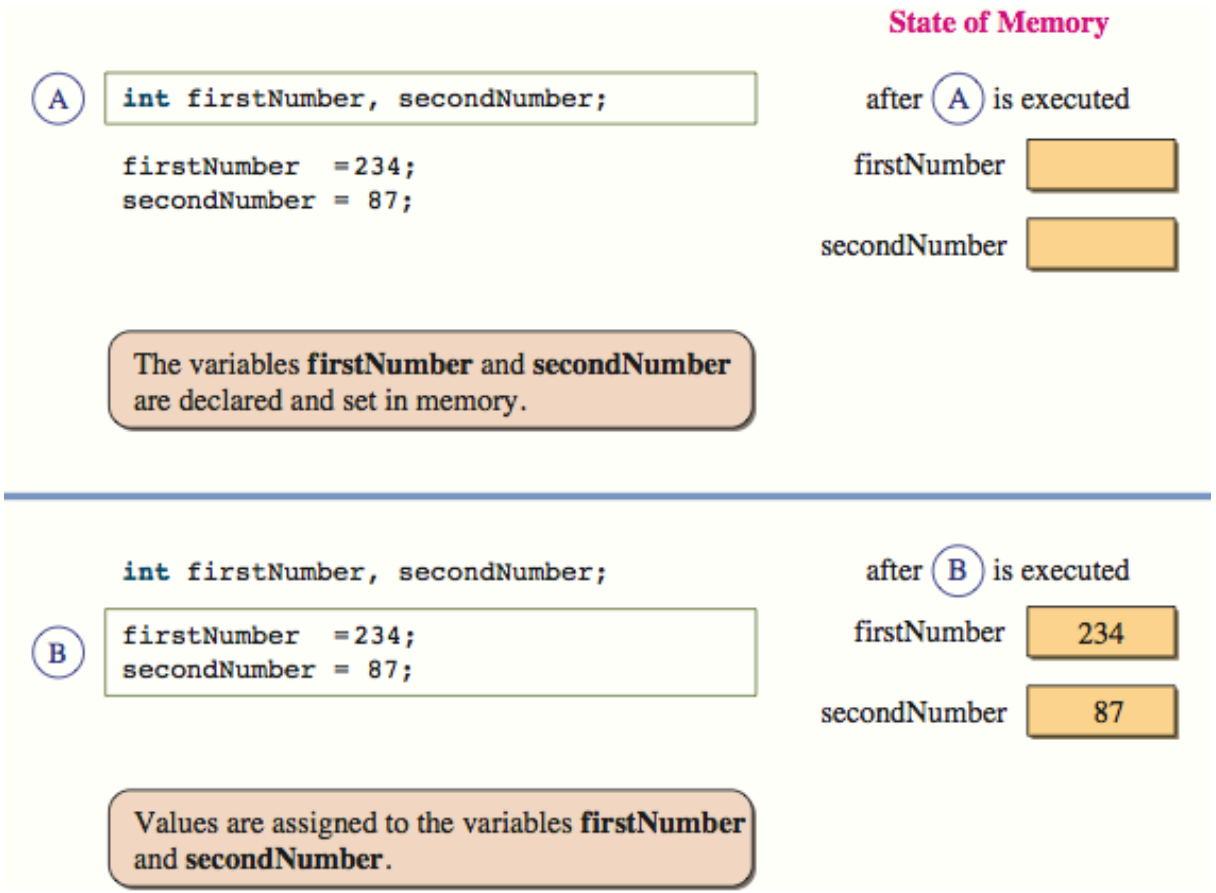
```
float f = 1.0f      float f = 1.0F      float f = 10
float f = 10L       double d = 1.0f     double d = 1.0
double d = 1.0d     double d = 1.0D     float f = 4e4f
float f = 4E4F      double d = 4e4      double d = 4E4D
```

- invalid:

```
float f = 1.0       float f = 1.0d      float f = 1.0D
float f = 4e4
```

Variables

- Variable Declaration and Assignment



Variables

- Difference between object and numerical data declaration

Numerical Data

```
int number;

number = 237;
number = 35;
```

number

Object

```
Customer customer;

customer = new Customer();
customer = new Customer();
```

customer

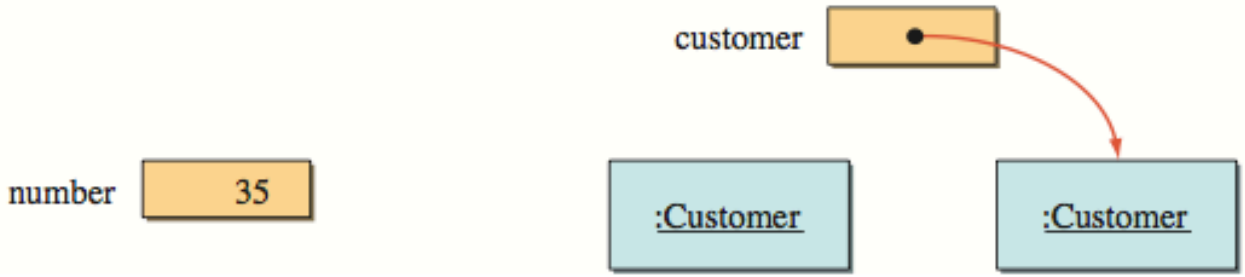
Variables

```
int number;
number = 237;
number = 35;
```

```
Customer customer;
customer = new Customer();
customer = new Customer();
```


```
int number;
number = 237;
number = 35;
```

```
Customer customer;
customer = new Customer();
customer = new Customer();
```



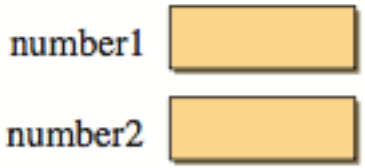
Variables

- An effect of assigning the content of one variable to another

Numerical Data

```
int number1, number2;
```

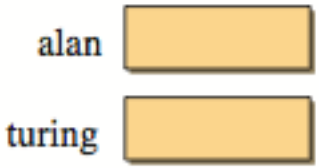
```
number1 = 237;
number2 = number1;
```



Object

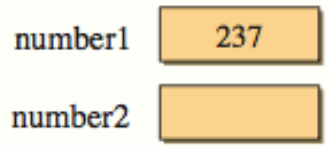
```
Professor alan, turing;
```

```
alan   = new Professor();
turing = alan;
```

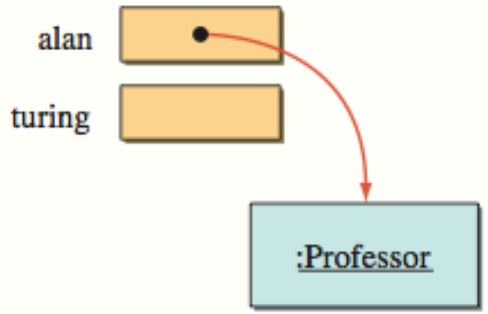


Variables

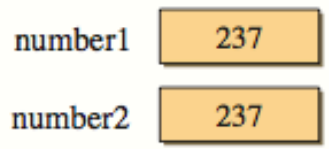
```
int number1, number2;
number1 = 237;
number2 = number1;
```



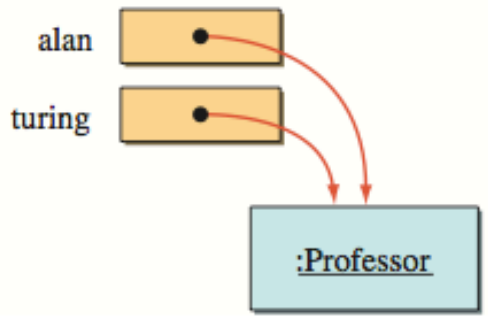

```
Professor alan, turing;
alan   = new Professor();
turing = alan;
```




```
int number1, number2;
number1 = 237;
number2 = number1;
```




```
Professor alan, turing;
alan   = new Professor();
turing = alan;
```



Arithmetic Expressions

- Arithmetic Expression: An expression involving operands and arithmetic operator

Operation	Java Operator	Example	Value (x = 10, y = 7, z = 2.5)
Addition	+	x + y	17
Subtraction	–	x – y	3
Multiplication	*	x * y	70
Division	/	x / y	1
		x / z	4.0
Modulo division (remainder)	%	x % y	3

Arithmetic Expressions

- Operand in arithmetic expressions can be:
  - a constant
  - a variable
  - a method call
  - another arithmetic expression

  - Ex.

```
newSalary = (salary * (1.07)) + getBaseRate(year);
```

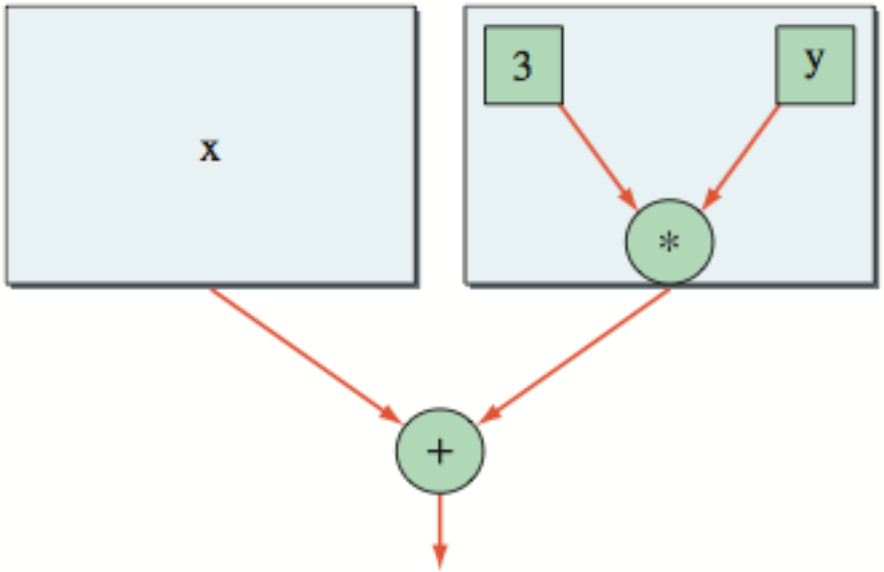
Arithmetic Expressions

- ประเภทของ Arithmethic Operator แยกตามจำนวนของ operand
  1. Unary Operator: คือ operator ที่มี operand 1 ตัว
     - Ex. -10, -x, +10
  2. Binary Operator: คือ operator ที่มี operand 2 ตัว
     - Ex. x + 1, x + y, x / y, 10 % 3

Arithmetic Expressions

- Precedence Rules: Order of evaluation of subexpressions

```
x + 3 * y
```

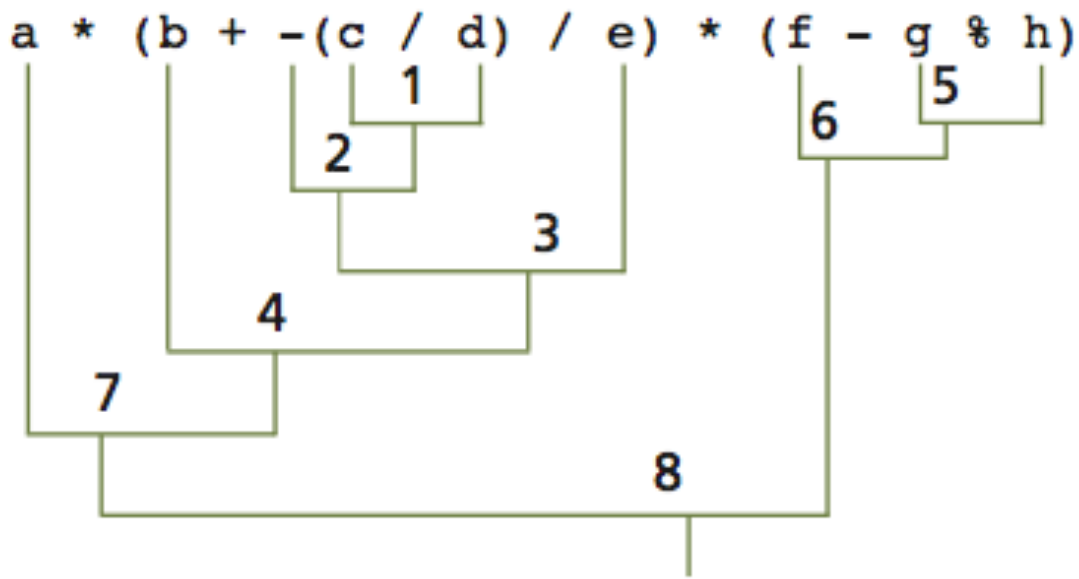


Arithmetic Expressions

Order	Group	Operator	Rule
High	Subexpression	( )	Subexpressions are evaluated first. If parentheses are nested, the innermost subexpression is evaluated first. If two or more pairs of parentheses are on the same level, then they are evaluated from left to right.
↕	Unary operator	-, +	Unary minuses and pluses are evaluated second.
	Multiplicative operator	*, /, %	Multiplicative operators are evaluated third. If two or more multiplicative operators are in an expression, then they are evaluated from left to right.
Low	Additive operator	+, -	Additive operators are evaluated last. If two or more additive operators are in an expression, then they are evaluated from left to right.

Arithmetic Expressions

- Ex. a * (b + -(c / d) / e) * (f – g % h)



Arithmetic Expressions

- Typecasting: คือการเปลี่ยนค่าจาก data type หนึ่งไปเป็น data type อื่น มี 2 แบบ
  1. Implicit Typecasting (or Numeric Promotion)
  2. Explicit Typecasting

Arithmetic Expressions

1. Implicit Typecasting (Numeric Promotion): คือการเปลี่ยนจาก data type เดิมไปเป็น data type ที่มี precision สูงกว่าเมื่อทำ operator ใดๆ

Operator Type	Promotion Rule
Unary	**1.** If the operand is of type `byte` or `short`, then it is converted to `int`. **2.** Otherwise, the operand remains the same type.
Binary	**1.** If either operand is of type `double`, then the other operand is converted to `double`. **2.** Otherwise, if either operand is of type `float`, then the other operand is converted to `float`. **3.** Otherwise, if either operand is of type `long`, then the other operand is converted to `long`. **4.** Otherwise, both operands are converted to `int`.

Arithmetic Expressions

- Ex. Implicit Typecasting (Numeric Promotion)

Division Operation	Result
23 / 5	4
23 / 5.0	4.6
25.0 / 5.0	5.0

Modulo Operation	Result
23 % 5	3
23 % 25	23
16 % 2	0

Arithmetic Expressions

2. Explicit Typecasting: คือการใช้คำสั่งเพื่อเปลี่ยนจาก data type เดิมไปเป็น data type ใหม่ โดยมีรูปแบบเป็น

(<data type>) <expression>

– Ex.

```
int x = (int) 1.5 * 3;          //x = 3
int x = (int) (1.5 * 3);        //x = 4
float x = (float) (1.5 * 3)     //x = 4.5f
```

Arithmetic Expressions

- Shorthand Assignment Operator: Some time we need to modify same variable value and reassigned it to same reference variable using a shorthand operator.

Operator	Usage	Meaning
+=	a += b;	a = a + b;
−=	a −= b;	a = a − b;
*=	a *= b;	a = a * b;
/=	a /= b;	a = a / b;
%=	a %= b;	a = a % b;

Constants

- Constant: is a variable whose value cannot change once it has been assigned.
  - A constant is declared in a manner similar to a variable but with the additional reserved word 'final'.

```
final double  PI = 3.14159;
final short   FARADAY_CONSTANT = 23060;
final double  CM_PER_INCH = 2.54;
final int     MONTHS_IN_YEAR = 12;


final double SPEED_OF_LIGHT = 3.0E+10D;
final short  MAX_WGT_ALLOWED = 400;
```

Displaying Numerical Values

- Using System.out
  - print() method

```
int num = 15;

System.out.print(num);
System.out.print(" ");
System.out.print(10);
```

```
15 10
```

  - println() method

```
int num = 15;
System.out.println(num);
System.out.println(10);
```

```
15
10
```

Displaying Numerical Values

- The '+' symbol could mean two different things: string concatenation or numerical addition.

```
int x = 1;
int y = 2;
```

```
System.out.println("x + y = " + x + y);
```

```
x + y = 12
```

```
System.out.println("x + y = " + (x + y));
```

```
x + y = 3
```

Displaying Numerical Values

- Using DecimalFormat Class to format the result
  - Without DecimalFormat

```
final double PI = 3.14159;

double radius, area, circumference;

radius = 2.35;

//compute the area and circumference
area          = PI * radius * radius;
circumference = 2.0 * PI * radius;

System.out.println("Given Radius: " + radius);
System.out.println("Area: " + area);
System.out.println("Circumference: " + circumference);
```

```
Given Radius: 2.35
Area: 17.349430775000002
Circumference: 14.765473
```

Displaying Numerical Values

– With DecimalFormat

```
DecimalFormat df = new DecimalFormat("0.000");

System.out.println("Given Radius: " + df.format(radius));
System.out.println("Area: " + df.format(area));
System.out.println("Circumference: "
                        + df.format(circumference));
```

```
Given Radius:  2.350
Area: 17.349
Circumference: 14.765
```

Displaying Numerical Values

- Using control characters to format output
  - “\t” (tab): to make tab between output
  - “\n” (new line): to make new line

```
final String TAB = "\t";
final String NEWLINE = "\n";
...
System.out.println(
          "Given Radius:  " + TAB + radius + NEWLINE +
          "Area:          " + TAB + area   + NEWLINE +
          "Circumference: " + TAB + circumference);
```

```
Given Radius:  2.35
Area:          17.349430775000002
Circumference: 14.765473
```

Getting Numerical Input

- Using Scanner Class to get input from console ทำได้ 2 วิธีคือ
  1. Using nextXxx() method in Scanner Class to get numerical value
  2. Using next() method in Scanner Class to get input String then using method from type wrapper class to convert to numerical value eg. Integer.parseInt(String) to convert String to int value

Getting Numerical Input

1. Using nextXxx() method in Scanner Class to get numerical value

Method	Example
`nextByte( )`	`byte b = scanner.nextByte( );`
`nextDouble( )`	`double d = scanner.nextDouble( );`
`nextFloat( )`	`float f = scanner.nextFloat( );`
`nextInt( )`	`int i = scanner.nextInt( );`
`nextLong( )`	`long l = scanner.nextLong( );`
`nextShort( )`	`short s = scanner.nextShort( );`

Getting Numerical Input

- Ex. nextInt()

```
Scanner scanner = new Scanner(System.in);
int age;
System.out.print("Enter your age: ");
age = scanner.nextInt( );
```

Getting Numerical Input

- Ex. nextInt() and nextFloat()

```
Scanner scanner = new Scanner(System.in);
int height;
float gpa;
System.out.print("Enter your height in inches: ");
height = scanner.nextInt( );
System.out.print("Enter your gpa: ");
gpa = scanner.nextFloat( );
```

Getting Numerical Input

- Ex. nextInt() for multiple input

```
Scanner scanner = new Scanner(System.in);

int num1, num2;

System.out.print("Enter two integers: ");

num1 = scanner.nextInt( );
num2 = scanner.nextInt( );

System.out.print("num1 = " + num1 + " num2 = " + num2);
```

```
Enter two integers:  12 ENTER
87 ENTER
num1 = 12 and num2 = 87
```

Space separates the two input values.

```
Enter two integers: 12 8 ENTER
num1 = 12 and num2 = 87
```

Getting Numerical Input

- Ex. nextDouble()

```
Scanner scanner = new Scanner(System.in);

double num;

System.out.print("Enter a double: ");
num = scanner.nextDouble( );

System.out.print("You entered " + num);
```

```
Enter a double:  35 ENTER
You entered 35.0
```

Getting Numerical Input

- Ex. Invalid – Type mismatch

```
Scanner scanner = new Scanner(System.in);

int num;

System.out.print("Enter an integer: ");
num = scanner.nextDouble();

System.out.print("You entered " + num);
```

Type mismatch

Bad Version

Getting Numerical Input

2. Using next() method in Scanner Class to get input String then using method from type wrapper class to convert to numerical value

```
Scanner sc = new Scanner(System.in);

System.out.print("Enter integer: ");
int i = Integer.parseInt(sc.next());

System.out.print("Enter double: ");
double d = Double.parseDouble(sc.next());

System.out.println("Input is: " + i + " and " + d);
```

```
Enter integer: 5
Enter double: 10
Input is: 5 and 10.0
```

Getting Numerical Input

- Type wrapper classes for each primitive data type

Primitive	Wrapper Class
boolean	Boolean
byte	Byte
char	Character
int	Integer
float	Float
double	Double
long	Long
short	Short

The Math Class

- The Math class in the java.lang package contains class methods for commonly used mathematical functions.

Class Method	Argument Type	Result Type	Description	Example
`abs( a )`	`int`	`int`	Returns the absolute `int` value of `a`.	abs(10) → 10 abs(−5) → 5
	`long`	`long`	Returns the absolute `long` value of `a`.	
	`float`	`float`	Returns the absolute `float` value of `a`.	
	`double`	`double`	Returns the absolute `double` value of `a`.	
`acos( a )`†	`double`	`double`	Returns the arccosine of `a`.	acos(−1) → 3.14159

The Math Class

`asin( a )`†	`double`	`double`	Returns the arcsine of `a`.	asin(1) → 1.57079
`atan( a )`†	`double`	`double`	Returns the arctangent of `a`.	atan(1) → 0.785398
`ceil( a )`	`double`	`double`	Returns the smallest whole number greater than or equal to `a`.	ceil(5.6) → 6.0 ceil(5.0) → 5.0 ceil(−5.6) → −5.0
`cos( a )`†	`double`	`double`	Returns the trigonometric cosine of `a`.	cos(π/2) → 0.0
`exp( a )`	`double`	`double`	Returns the natural number *e* (2.718 . . .) raised to the power of `a`.	exp(2) → 7.389056099

The Math Class

Class Method	Argument Type	Result Type	Description	Example
`floor( a )`	`double`	`double`	Returns the largest whole number less than or equal to `a`.	floor(5.6) → 5.0 floor(5.0) → 5.0 floor(−5.6) → −6.0
`log( a )`	`double`	`double`	Returns the natural logarithm (base *e*) of `a`.	log(2.7183) → 1.0
`max( a, b )`	`int`	`int`	Returns the larger of `a` and `b`.	max(10, 20) → 20
	`long`	`long`	Same as above.	
	`float`	`float`	Same as above.	
`min(a, b)`	`int`	`int`	Returns the smaller of `a` and `b`.	min(10, 20) → 10
	`long`	`long`	Same as above.	
	`float`	`float`	Same as above.	
`pow(a, b)`	`double`	`double`	Returns the number `a` raised to the power of `b`.	pow(2.0, 3.0) → 8.0

The Math Class

random( )	<none>	double	Generates a random number greater than or equal to 0.0 and less than 1.0.	
round(a)	float	int	Returns the int value of a rounded to the nearest whole number.	round(5.6) → 6 round(5.4) → 5 round(−5.6) → −6
	double	long	Returns the float value of a rounded to the nearest whole number.	
sin(a)†	double	double	Returns the trigonometric sine of a.	sin(π/2) → 1.0
sqrt(a)	double	double	Returns the square root of a.	sqrt(9.0) → 3.0
tan(a)†	double	double	Returns the trigonometric tangent of a.	tan(π/4) → 1.0
toDegrees	double	double	Converts the given angle in radians to degrees.	toDegrees(π/4) → 45.0
toRadians	double	double	Reverse of toDegrees.	toRadians(90.0) → 1.5707963

The Math Class

- All methods and constants in Math class are class methods and class constants, so using these syntax
  - Using methods:
    Math.<method name>(<args>)
  - Using constants:
    Math.<constant name>

The Math Class

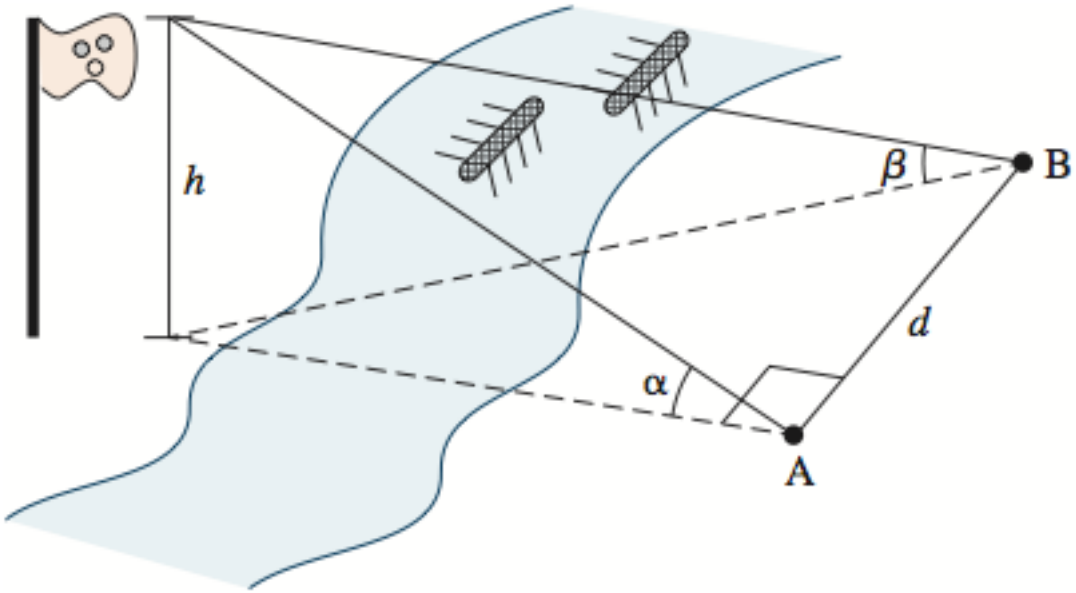
- Ex.

$$\frac{1}{2}\sin\left(x - \frac{\pi}{\sqrt{y}}\right)$$

```
(1.0 /2.0) * Math.sin( x - Math.PI / Math.sqrt(y) )
```

The Math Class

- Ex.



The Math Class

- Ex. (cont.)

$$h = \frac{d \sin \alpha \sin \beta}{\sqrt{\sin(\alpha + \beta) \sin(\alpha - \beta)}}$$

```
height = ( distance * Math.sin(alphaRad) * Math.sin(betaRad) )
       /
     Math.sqrt( Math.sin(alphaRad + betaRad) *
                Math.sin(alphaRad - betaRad) );
```

Random Number Generation

- Random class from the java.util package used to generate random numbers using these methods:
  1. nextInt(): returns an int value, that is any value between -2147483648 and 2147483647

  2. nextInt(int bound): returns an int value between 0 (inclusive) and the specified value (exclusive)

Random Number Generation

- Ex. To generate a random integer between -2147483648 and 2147483647

```
Random random = new Random( );

int num = random.nextInt( );
```

Random Number Generation

- Ex. To generate a random integer between 0 and 10

```
Random random = new Random( );
int num = random.nextInt(11);
```

Random Number Generation

- Ex. To generate a random integer between 1 and 6 for dice game

This generates an integer between 0 and 5, inclusive.

```
int num = random.nextInt(6) + 1;
```

Random Number Generation

- To generate a random integer in the range of [min, max] where 0 <= min < max, we write

```
int num = random.nextInt(max-min+1) + min;
```

This generates an integer between 0 and (max-min), inclusive.

The GregorianCalendar Class

- We have a very useful class named java.util.GregorianCalendar in manipulating calendar information such as year, month, and day.
  - Create a new GregorianCalendar object that represents today as

```
GregorianCalendar today = new GregorianCalendar( );
```

The GregorianCalendar Class

– Create a new GregorianCalendar object that represents specific day (July 4, 1776) as

The value of 6 means July.

```
GregorianCalendar independenceDay =
            new GregorianCalendar(1776, 6, 4);
```

```
GregorianCalendar independenceDay =
          new GregorianCalendar(1776, Calendar.JULY, 4);
```

The GregorianCalendar Class

- Constants defined in the Calendar class for retrieved different pieces of calendar/time information

Constant	Description
YEAR	The year portion of the calendar date
MONTH	The month portion of the calendar date
DATE	The day of the month
DAY_OF_MONTH	Same as DATE
DAY_OF_YEAR	The day number within the year
DAY_OF_MONTH	The day number within the month
DAY_OF_WEEK	The day of the week (Sun—1, Mon—2, etc.)
WEEK_OF_YEAR	The week number within the year
WEEK_OF_MONTH	The week number within the month
AM_PM	The indicator for AM or PM (AM—0 and PM—1)
HOUR	The hour in 12-hour notation
HOUR_OF_DAY	The hour in 24-hour notation
MINUTE	The minute within the hour

The GregorianCalendar Class

- Ex. Running the program at February 13, 2008,13:30 p.m.

```
GregorianCalendar cal = new GregorianCalendar();

System.out.println(cal.getTime());
System.out.println("");

System.out.println("YEAR:          " + cal.get(Calendar.YEAR));
System.out.println("MONTH:         " + cal.get(Calendar.MONTH));
System.out.println("DATE:          " + cal.get(Calendar.DATE));
```

```
Wed Feb 13:30:51 PST 2008

YEAR:           2008
MONTH:          1
DATE:           13
```

The GregorianCalendar Class

- Ex. (cont.)

```
System.out.println("DAY_OF_YEAR:    "
                                + cal.get(Calendar.DAY_OF_YEAR));
System.out.println("DAY_OF_MONTH:   "
                                + cal.get(Calendar.DAY_OF_MONTH));
System.out.println("DAY_OF_WEEK:    "
                                + cal.get(Calendar.DAY_OF_WEEK));
System.out.println("WEEK_OF_YEAR:   "
                                + cal.get(Calendar.WEEK_OF_YEAR));
System.out.println("WEEK_OF_MONTH:  "
                                + cal.get(Calendar.WEEK_OF_MONTH));
```

```
DAY_OF_YEAR:      44
DAY_OF_MONTH:     13
DAY_OF_WEEK:      4
WEEK_OF_YEAR:     7
WEEK_OF_MONTH:    3
```

The GregorianCalendar Class

- Ex. (cont.)

```
System.out.println("AM_PM:          " + cal.get(Calendar.AM_PM));
System.out.println("HOUR:           " + cal.get(Calendar.HOUR));
System.out.println("HOUR_OF_DAY:    "
                              + cal.get(Calendar.HOUR_OF_DAY));
System.out.println("MINUTE:         " + cal.get(Calendar.MINUTE));
```

```
AM_PM:          1
HOUR:           1
HOUR_OF_DAY:    13
MINUTE:         30
```

The GregorianCalendar Class

- Ex. Display day name of week of 4th July 1776

```
GregorianCalendar independenceDay
        = new GregorianCalendar(1776, Calendar.JULY, 4);

SimpleDateFormat sdf = new SimpleDateFormat("EEEE");

System.out.println("Output: " + sdf.format(independenceDay.getTime()));
```

```
run:
Output: Thursday
```

Summary

- A variable must be declared before we can assign a value to it.
- There are six numerical data types in Java: byte, short, int, long, float, and double.
- A primitive variable is a memory location in which to store a value.
- Object names are synonymous with variables whose contents are memory addresses.

Summary

- Precedence rules determine the order of evaluating arithmetic expressions.
- Constants hold values just as variables do, but we cannot change their values, using ‘final’ keyword.
- The standard classes introduced in this chapter are Math, GregorianCalendar and DecimalFormat

Summary

- The Math class contains many class methods for mathematical functions.
- The GregorianCalendar class is used in the manipulation of calendar information.
- The DecimalFormat class is used to format numerical data.

Reference

- C. Thomas Wu, An Introduction to Object-Oriented Programming with Java, 5th Edition
  - Chapter 3: Numerical Data

Question?