Assignment Operators

Dr. Rekha P
Amrita Center for Wireless Networks & Applications
Amrita School of Engineering
Amrita Vishwa Vidyapeetham
rekhap@am.amrita.edu

Assignment Operator

- An operator to assign a value/expression to a variable
- Use ‘=‘ as assignment operator
- lvalue = rvalue
- lvalue should be a variable
- rvalue is an expression that evaluates to a value
- rvalue is evaluated and the result is stored in lvalue
- lvalue can be used in rvalue

```
int x=100+300;
```

Assignment Operator…

- Can assign multiple variables in the same statement
- This is evaluated from right to left
- Step1: y=100+300
- Step2: x=y
- x and y are equal to 400

```
int y
int x=y=100+300;
```

Assignment Statement

- A statement that evaluates an expression and assigns the result to a variable

```
int y = 10;
double d = 100.25;
int x = 10 ;
x = x +30 ;
```

- x is 40
- Each Assignment statement is also called Assignment expression

Augmented Assignment Operator

```
int i = 10;  ...
i=i+10 ;
```

- This can be written in an alternate way

```
i += 10 ;
```

- Augmented Assignment operator can be combined with any arithmetic operator

```
int i = 2;
i *= 5+1;
```

- Equivalent to i= i*(5+1) => 12
- *NOT i= i*5 +1 => 11*

Concluding Thought…

"In times of tragedies, our duty is to lend a helping hand to those in grief and thus light lamps of kindness and compassion."

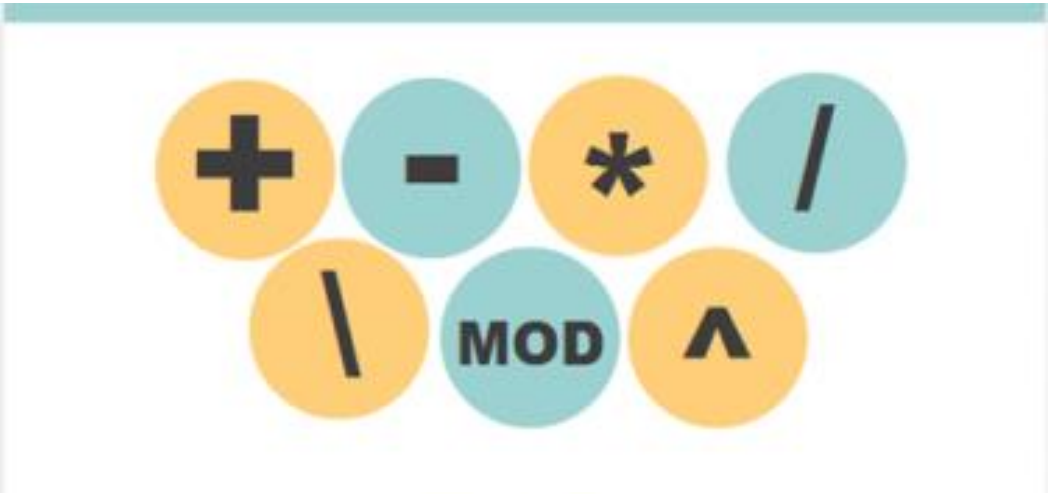
— **Amma**

Arithmetic Operators

Dr. Rekha P
Amrita Center for Wireless Networks & Applications
Amrita School of Engineering
Amrita Vishwa Vidyapeetham
rekhap@am.amrita.edu

Arithmetic Operators

- Addition
- Subtraction
- Multiplication
- Division
- Modulo



Addition

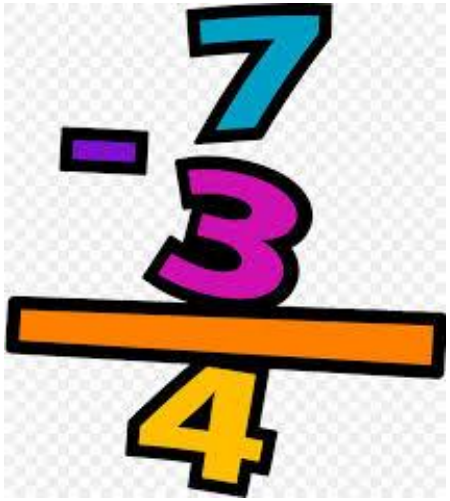
- Operator to add two or more values
- Use ‘+’ as the addition operator
- int x ;
- x = 5+6 ;
- System.out.println(“The sum is “ + x) ;
- We can also print the sum directly
- System.out.println(“The sum is “ + (5+6)) ;
- ‘+’ also used as the string concatenation operator

Addition…

- System.out.println(“The sum is “ + (5+6)) ;
  - Output: The sum is 11
- Parenthesis are VERY Important
- Without brackets it will be considered as concatenation operator
- System.out.println(“The sum is “ + 5+6) ;
  - Output: The sum is 56

Subtraction

- Operator to subtract two or more values
- Using ‘-’
- int x = 3-1 ;
- System.out.println(“The difference is “+x) ;
- We can display the difference directly
- System.out.println(“The difference is “+(3-1)) ;



Subtraction…

- System.out.println("The difference is "+(3-1)) ;
  - Output: The difference is 2
- Parenthesis is IMPORTANT
- System.out.println("The difference is "+ 3-1) ;
  - ERROR
  - "The difference is + 3 is evaluated first and get the string
  - Now you are trying to subtract a constant from a string and results in an error

Multiplication

- Operator to multiply two or more values
- Use ‘*’ as the multiplication operator
- int x ;
- x = 5*6 ;
- System.out.println(“The product is “ + x) ;
- System.out.println(“The product is “ + (5*6)) ;
- System.out.println(“The product is “ + 5*6) ;
- Works with and Without Parenthesis
  - ‘*’ is evaluated first
  - Operator precedence

Division

- Operator to divide two or more values
- Use ‘/’ as the division operator
- int x ;
- x = 6/2 ;
- System.out.println(“The result is “ + x) ;
- System.out.println(“The result is “ + (6/2)) ;
- System.out.println(“The result is “ + 6/2) ;
- Works with and Without Parenthesis
  - ‘/’ is evaluated first
  - Operator precedence

Division...

- The result of the division of two integers is always an integer – ½ = 0 NOT 0.5
- Use casting to get double result
- The result of the division of two doubles is double
- The result of the division of a double and an integer is double(implicit casting)

Concluding Thought…

"In times of tragedies, our duty is to lend a helping hand to those in grief and thus light lamps of kindness and compassion."

— **Amma**

Arithmetic Operators-Casting

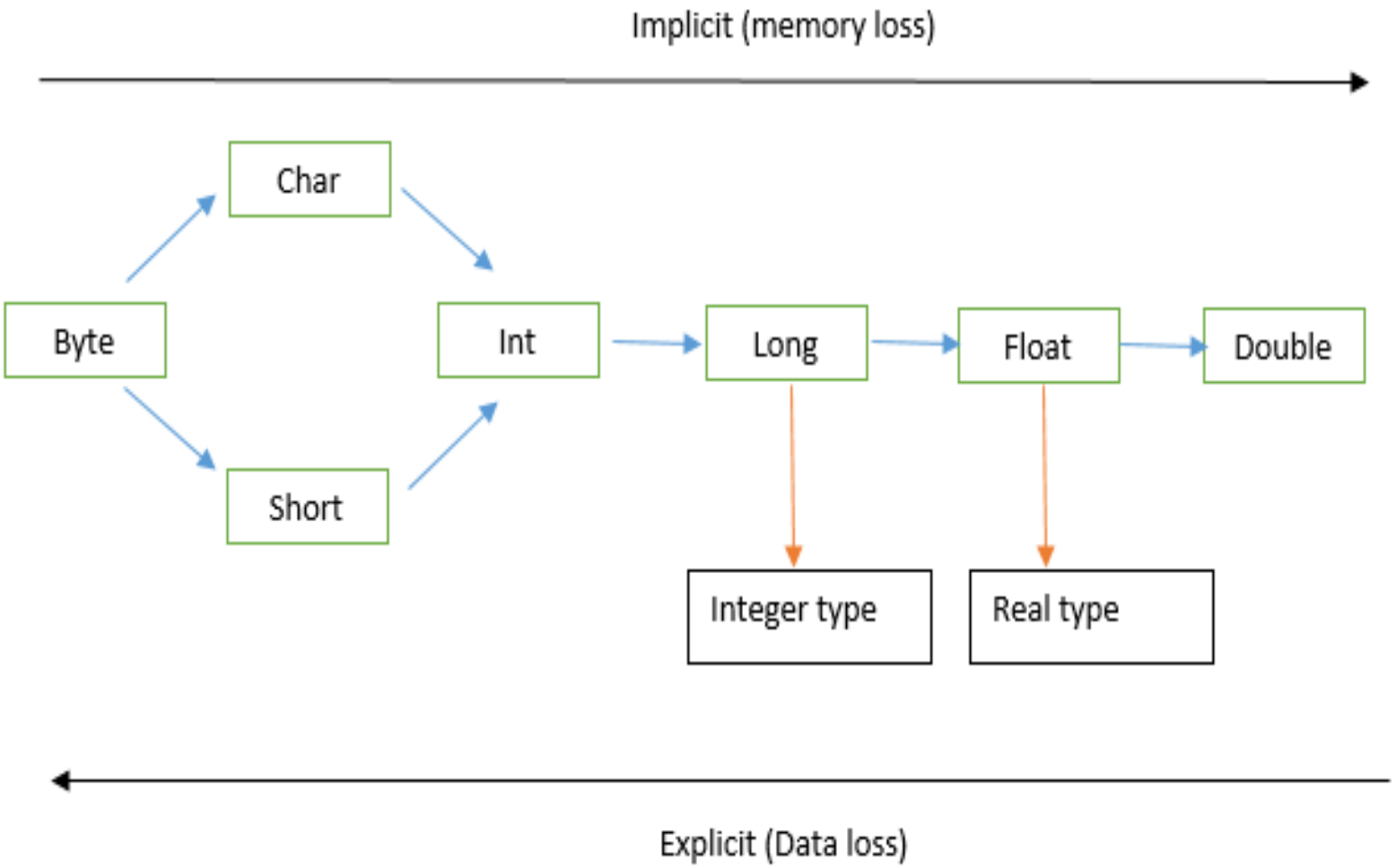
Dr. Rekha P
Amrita Center for Wireless Networks & Applications
Amrita School of Engineering
Amrita Vishwa Vidyapeetham
rekhap@am.amrita.edu

Casting

- Converting a datatype into another
- Implicit casting
  - Happens automatically converting from a narrower range datatype to a wider range type
    - Converting an int to a double/float/long
    - Converting float to double
- Explicit casting
  - Does not happen automatically.

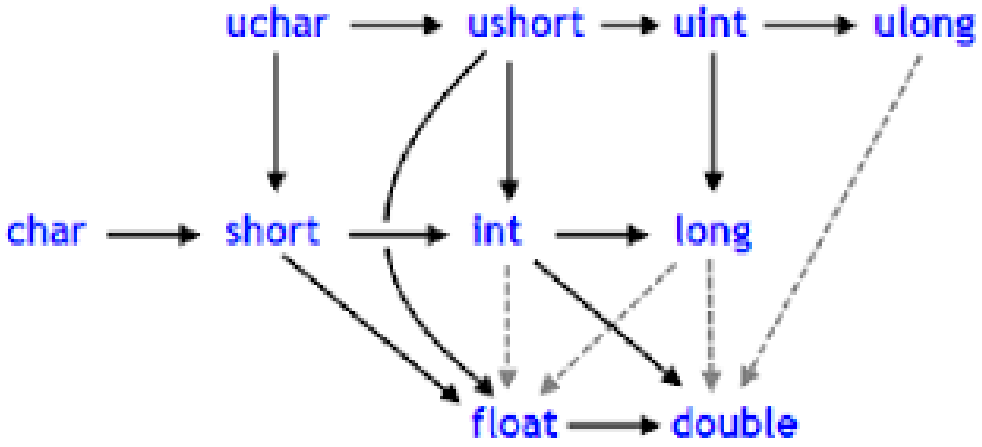
  Should be done by the programmer explicitly when converting from a wider datatype to narrower datatype
    - Converting a double/float/long to an int
    - Converting double to float

Casting…


Implicit (memory loss)
Char
Byte
Int
Long
Float
Double
Short
Integer type
Real type
Explicit (Data loss)

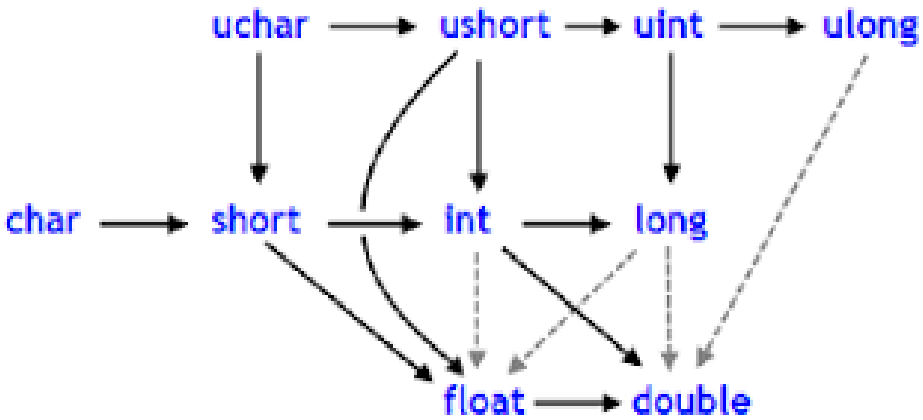
Implicit Casting

- double d1 = 4 ;// int to double
- double d2 = 5.7f ;// float to double
- long l1 = 100 ; //int to long
- Implicit casting happens because
  - The range of double is wider than int/float
  - The range of longis wider than int



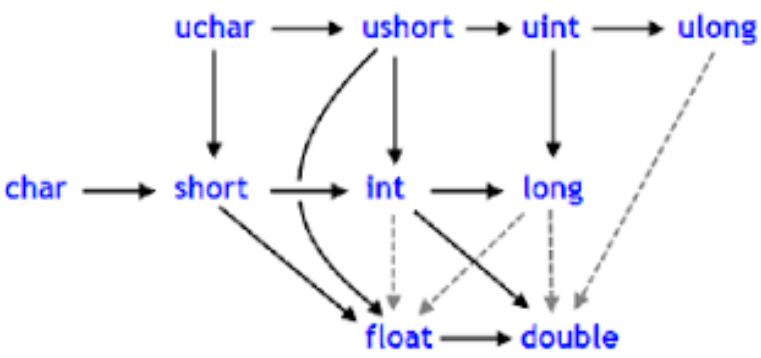
Explicit Casting

- int d1 = 4.5 ;// ERROR
- int d2 = 4L ;// ERROR
- float f1 = 4.5 ;// ERROR
- Implicit casting *CAN NOT* happen because
  - The range of int/float is NARROWER than double
  - The range of int is NARROWER than long
- Explicit casting should be Used..



Explicit Casting…

- (new datatype)expression
- int d1 = (int)4.5 ;// 4 – Data loss
- int d2 = (int)8L ;// 8
- float f1 = (float)4.5 ;// 4.5



- double d1 = 5.2 + 3 ;
- Every arithmetic operator should be applied between values of same type
- 3 will be casted to 3.0
- 5.2+3.0 is evaluated and stored in d1

Concluding Thought…

"In times of tragedies, our duty is to lend a helping hand to those in grief and thus light lamps of kindness and compassion."

— **Amma**

Arithmetic Operators-Division

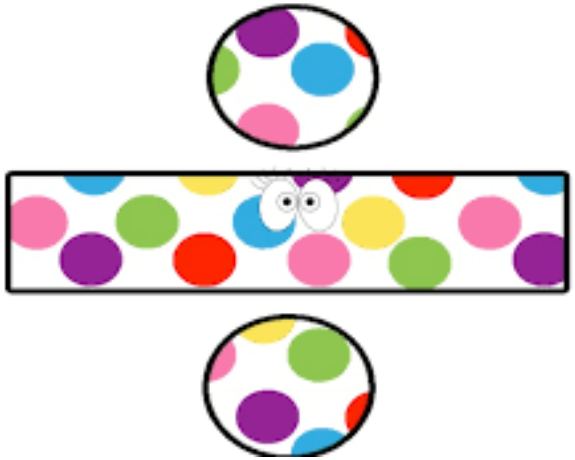
Dr. Rekha P
Amrita Center for Wireless Networks & Applications
Amrita School of Engineering
Amrita Vishwa Vidyapeetham
rekhap@am.amrita.edu

Dividing Integers

- Dividing an int by an int gives an int
- System.out.println(1/2) ; //0
- double d = ½ ; //0.0
  - int to double
  - implicit casting
  - 0 will be casted to 0.0
- int i = ½ ; 0

Dividing Doubles

- Dividing double by a double gives a double
- System.out.println(1.0/2.0) ; //0.5
- double d = 1.0/2.0 ; //0.5
  - Assigning double to a double
- int i = 1.0/2.0 ; //ERROR
  - Trying to assign double to an int
  - Implicit casting will not happen



Dividing Float

- Dividing float by a float gives a float
- System.out.println(1.0f/2.0f) ; //0.5f
- double d = 1.0f/2.0f ; //0.5
  - Assigning float to a double
  - Implicit casting
- int i = 1.0f/2.0f ; //ERROR
  - Trying to assign float to an int
  - Implicit casting will not happen

Dividing Integers and Doubles

- Integer will be casted to double automatically
- System.out.println(1/2.0) ; //0.5
- System.out.println(1.0/2) ; //0.5
- double d = 1 / 2.0 // 0.5
  - double to double
- int d = 1 / 2.0 // ERROR
  - double to int
  - Implicit casting will not work

Dividing Floats and Doubles

- Float will be casted to double automatically
- System.out.println(1.0f/2.0) ; //0.5
- System.out.println(1.0/2.0f) ; //0.5
- double d = 1 / 2.0f // 0.5
  - double to double
- float d = 1 / 2.0 // ERROR
  - Trying to assign double to float
  - Implicit casting will not work

Dividing Floats and Integers

- Integer will be casted to float automatically
- System.out.println(1.0f/2) ; //0.5f
- System.out.println(1/2.0f) ; //0.5f
- double d = 1 / 2.0f // 0.5
  - float to double
- int d = 1 / 2.0f // ERROR
  - float to int
  - Implicit casting will not work

Concluding Thought…

"In times of tragedies, our duty is to lend a helping hand to those in grief and thus light lamps of kindness and compassion."

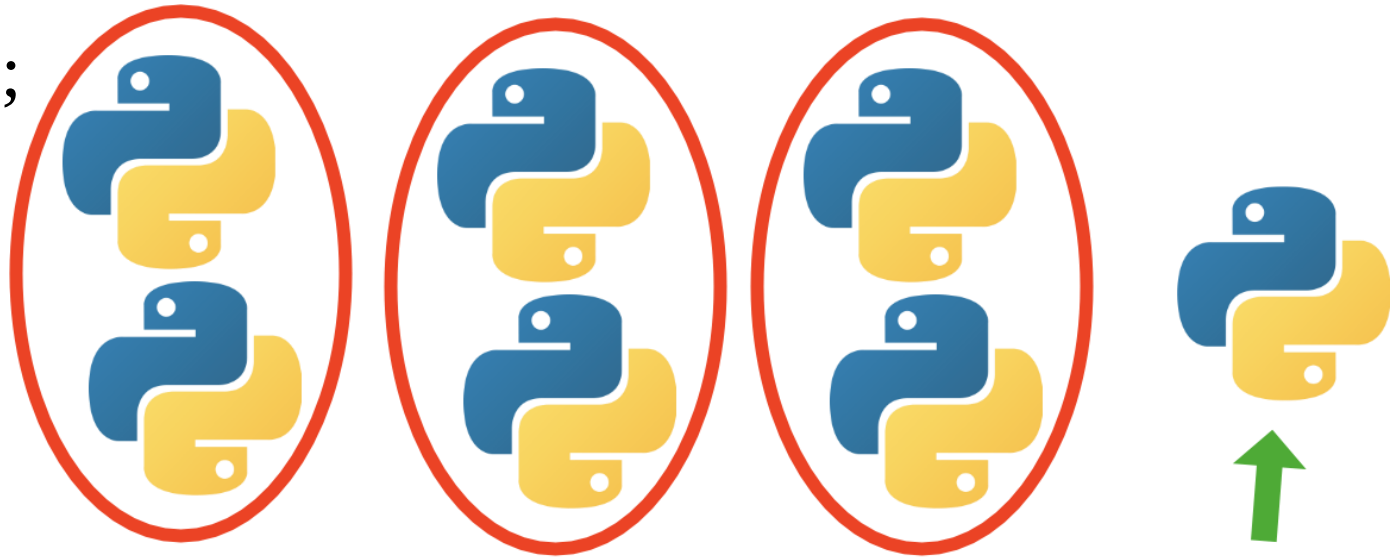
— **Amma**

Arithmetic Operators – Modulo and Increment/Decrement

Dr. Rekha P
Amrita Center for Wireless Networks & Applications
Amrita School of Engineering
Amrita Vishwa Vidyapeetham
rekhap@am.amrita.edu

Modulo Operator

- Operator to get reminder of a division
- Using ‘%’
- int x = 7%2 ;

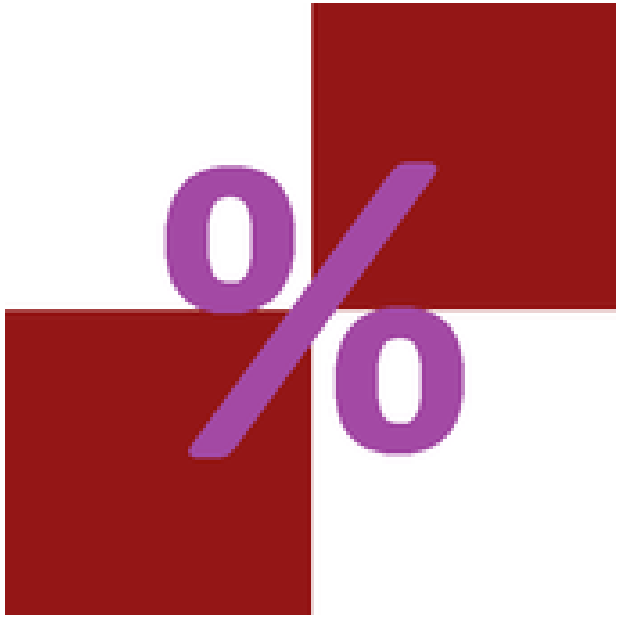


7 / 2 = 3 R 1

7 % 2 = 1

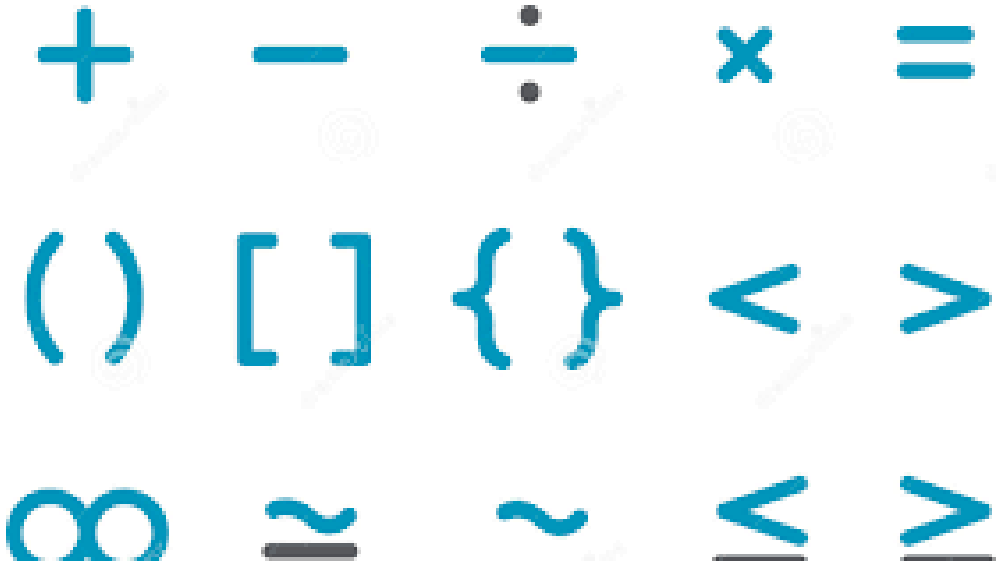
Modulo Operator…

- int x = 7%2 ;
- System.out.println(“The result is “+x) ;
- We can display the difference directly
- System.out.println(“The result is “ + (7%2)) ;
- System.out.println(“The result is “ + 7%2) ;
- Works with and Without Parenthesis
  - ‘%’ is evaluated first
  - Operator precedence



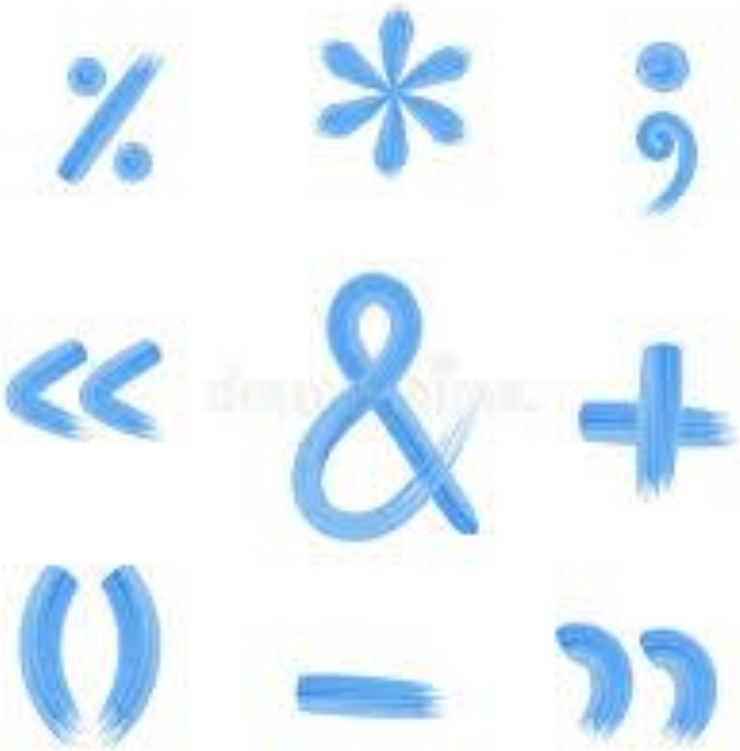
Operator Precedence

- Operations are done from left to right
- Operations between parenthesis
- Multiplication and Division
- Addition and Subtraction
- 2 * (1+5) – 12/ (4+2)



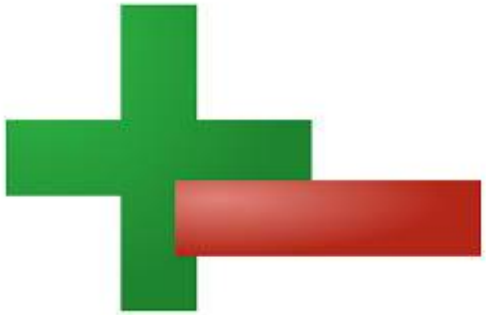
Operator Precedence…

- 2 * (1+5) – 12/ (4+2) - brackets
- 2 * 6 -12/6 – multiplication and division
- 12 – 2
- 10



Increment Operators

- Used to increase the value of a variable by 1
- Increment operator - ++
- Can be used in two ways
- Pre-increment
- Use before the variable - ++i
- Post-increment
- Use after the variable - i++



Increment Operators…

```
int i = 5;
i++;
System.out.println(i) ;
```

```
int i = 5;
++i;
System.out.println(i) ;
```

Output 6

```
int i = 1;
j = ++i;
System.out.println("i is " +i+" j is "+j) ;
```

i is 2 j is 2

```
int i = 1;
j = i++;
System.out.println("i is " +i+" j is "+j) ;
```

i is 2 j is 1

++i increments the value by 1 and uses the *NEW* value in the statement

i++ uses the *ORIGINAL* value in the statement and increments the value by 1

Increment Operators Example

```
int i = 1, j = 1;
j = i++ + 4 ;
System.out.println("i is " +i+" j is "+j) ;
```

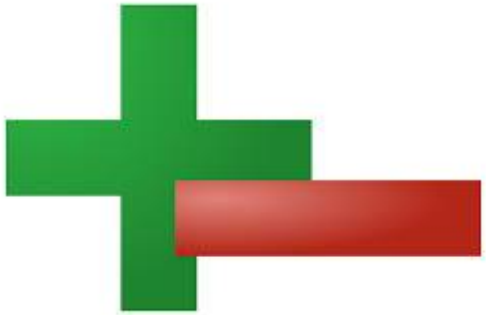
i is 2 j is 5

```
int i = 1, j = 1;
j = ++i + 4 ;
System.out.println("i is " +i+" j is "+j) ;
```

i is 2 j is 6

Decrement Operators

- Used to decrease the value of a variable by 1
- Decrement operator --
- Can be used in two ways
- Pre-decrement
- Use before the variable --i
- Post-decrement
- Use after the variable i—
- SAME RULES as that of Increment Operators



Decrement Operators…

```
int i = 5;
i--;
System.out.println(i) ;
```

```
int i = 5;
--i;
System.out.println(i) ;
```

Output 4

```
int i = 1;
j = --i;
System.out.println("i is " +i+" j is "+j) ;
```

i is 0 j is 0

```
int i = 1;
j = i--;
System.out.println("i is " +i+" j is "+j) ;
```

i is 0 j is 1

--i decrements the value by 1 and uses the *NEW* value in the statement

i-- uses the *ORIGINAL* value in the statement and decrements the value by 1

Concluding Thought…

"In times of tragedies, our duty is to lend a helping hand to those in grief and thus light lamps of kindness and compassion."

— **Amma**

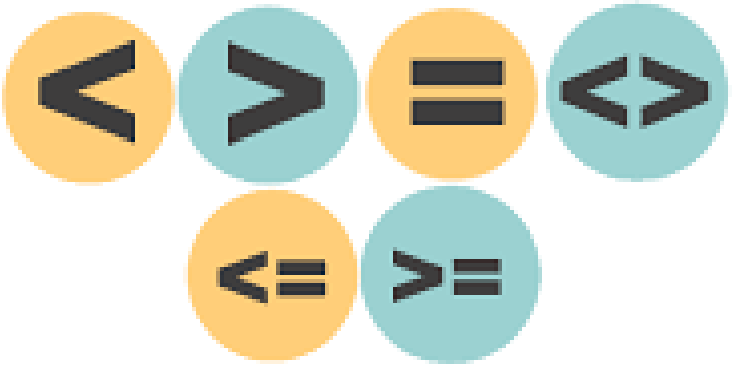
Relational Operators

Dr. Rekha P
Amrita Center for Wireless Networks & Applications
Amrita School of Engineering
Amrita Vishwa Vidyapeetham
rekhap@am.amrita.edu

Relational Operators

- Operators used to make Comparisons
- Equality Operators: ==
- Inequality Operators: !=
- Greater than Operator: >
- Greater than or Equal Operator: >=
- Less than Operator: <
- Less than or Equal Operator: <=

Result is Boolean



Equality Operator

- Equality Operators: ==
- Test two expressions are Equal
- boolean b1 = (1==1) ; //true
- boolean b2 = (1==2) ; //false
- boolean b3 = ((1+4)==(7-2)); //true

Inequality Operator

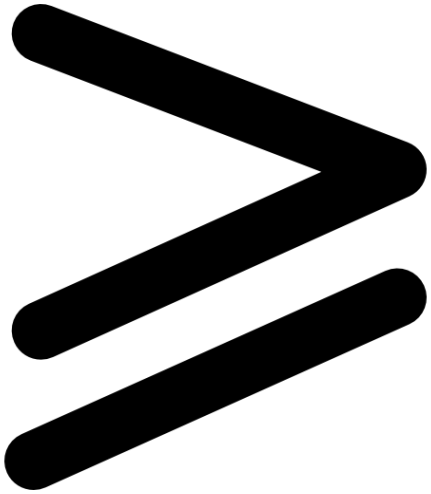
- Inequality Operators: !=
- Test two expressions are NOT Equal
- boolean b1 = (1!=1) ; //false
- boolean b2 = (1!=2) ; //true
- boolean b3 = ((1+4)!=(7-3)); //true

Greater than Operator

- Greater than Operator: >
- Test if an expression is greater than another
- boolean b1 = (1>5) ; //false
- boolean b2 = (3>2) ; //true
- boolean b3 = ((1+4)>(4-3)); //true
- boolean b4 = (1>1) ; //false

Greater than or Equal Operator

- Greater than or Equal Operator: >=
- Test if an expression is greater than or equal to another
- boolean b1 = (1>=5) ; //false
- boolean b2 = (3>=2) ; //true
- boolean b3 = ((1+4)>=(8-3)); //true
- boolean b4 = (1>=1) ; //true



Less than Operator

- Less than Operator: <
- Test if an expression is less than another
- boolean b1 = (1<5) ; //true
- boolean b2 = (3<2) ; //false
- boolean b3 = ((1+4)<(4-3)); //false
- boolean b4 = (1<1) ; //false

Less than or Equal Operator

- Less than or Equal Operator: <=
- Test if an expression is less than or equal to another
- boolean b1 = (5<=2) ; //false
- boolean b2 = (3<=7) ; //true
- boolean b3 = ((1+4)<=(8-3)); //true
- boolean b4 = (1<=1) ; //true

Concluding Thought…

"In times of tragedies, our duty is to lend a helping hand to those in grief and thus light lamps of kindness and compassion."

— **Amma**

Logical Operators

Dr. Rekha P
Amrita Center for Wireless Networks & Applications
Amrita School of Engineering
Amrita Vishwa Vidyapeetham
rekhap@am.amrita.edu

Logical Operators

- Operators used to construct complex conditions
- Logical AND - &&
- Logical OR -||
- Logical NOT- !
- Used between boolean values

Result is Boolean

NOT (!)

AND (&&)

OR (||)

Combining Conditions

- AND:condition_1&&condition_2&&condition_3
- True – if all conditions are true
- OR:condition_1||condition_2||condition_3
- True – if at least one condition is true

Negating a boolean Value

- Logical NOT - !
- !True = False
- !False = True

Examples

- boolean b1 = true && true ; //true
- boolean b2 = b1&& false && true ; //false
- boolean b3 = b2||true; //true
- boolean b4 = !b2; //true
- boolean b5 = !(b4&&b2); //true
- Usage
  - If isRaining || isCold – Wear a Jacket
  - If n>=1 && n<=10 n is between 1 and 10

Concluding Thought…

"In times of tragedies, our duty is to lend a helping hand to those in grief and thus light lamps of kindness and compassion."

— **Amma**

Conditional Operators

Dr. Rekha P
Amrita Center for Wireless Networks & Applications
Amrita School of Engineering
Amrita Vishwa Vidyapeetham
rekhap@am.amrita.edu

Conditional Operator

- An operator that evaluates to an expression based on a condition - ?:

```
boolean expression?expression1:expression2;
```

- Evaluates to expression1 if boolean expression is TRUE
- Evaluates to expression2 if boolean expression is FALSE

Conditional Operator-Example

- Max of 2 numbers

```
int a = 10; int b = 20;
int max = a>b?a:b;
System.out.println("Max Value is "+max) ;
```

```
int a = 10; int b = 20;
int max = a<b ? b : a;
System.out.println("Max Value is "+max) ;
```

Concluding Thought…

"In times of tragedies, our duty is to lend a helping hand to those in grief and thus light lamps of kindness and compassion."

— **Amma**