

Precedence, Mixed Expressions, Math Methods, and Output Formatting

I. Precedence Rules - order of operations

Introduction example: $4 + 2 \times 3 = ?$

- A) Parenthesis,
Division or Multiplication (from left to right),
Addition or Subtraction (from left to right)

ex.1) $2 - 5 + 1$

ex.2) $4 / 2 * 2$

ex.3) $3 + 6 / 3 * 2 - 1$

ex.4) $(3 + 6) / 3 * 2 - 1 + 2$

- B) Convert each of the following mathematical expressions to a Java arithmetical expression. Use parenthesis only where needed.

ex.1) $y = 5 \div a$ (integer quotient)

ex.2) $x = B + 2AC$

ex.3) $x = 2(m + n)$

ex.4) $x = \frac{3 - b}{r + 4}$ (floating point result)

II. Data Conversion with Mixed Data Types

Recall:

- 1) If x is an integer type, then after the statement $x = 5;$ x has the value of **5** stored in its memory location.
- 2) If x is a double or float type, then after the statement $x = 5.0;$ x has the value of **5.0** stored in its memory location.

Note: You can only store an integer in an integer variable and a floating point decimal in a float or double. But what if we tried to store an integer in a double or a decimal in an integer variable??

A) Assignment conversion (or also called widening, promotion, or type coercion) - the automatic conversion of a data type to another type due to mixing types in an expression.

1) Storing an integer in a decimal type:

ex.1) If x is a double type, then after the statement $x = 5;$
 x has the value of **5.0** stored in its memory location.

*When an integer is assigned to a decimal (double or float type) variable the integer is automatically converted to a number with a decimal point and a zero after it!

2) Storing a decimal in an integer:

ex.2) If x is an integer type, then the statement $x = 5.9;$
is illegal in Java (but not in C++)!

*When a decimal (**double** or **float** type) can not directly be assigned to an integer variable.

B) Type Casting - the explicit conversion of a value from one data type to another, by using a typecast operator which is really just the name of the desired data type in parenthesis

```
ex.1) int x;  
      double y = 5.9;  
      x = y; // illegal
```

but you can force the value of `y` into `x` by casting it:

```
x = (int) y;
```

or

```
x = (int) (y);
```

but not

```
x = int (y);
```

**But the value in `x` is not 5.9 but 5.

*When a decimal (**double** or **float** type) is assigned to an integer variable by type casting the decimal part is truncated (dropped off)!

Mixing data types can cause problems in a program.

```
ex.1) double x;  
      int a = 30, b = 40;  
      x = a / b;
```

What is x after the last line? Answer: 0.0 !!!!!

We can fix this result a couple of different ways:

```
ex.1) double x;  
      int a = 30, b = 40;  
      x = (double)a/b;
```

or

```
x = a/(double)b;
```

or

```
x = (double)a/(double)b;  
(a and/or b could also be in parenthesis but are usually not  
in any of the above)
```

But NOT

```
x = (double)(a/b); Why?
```

Because, the integers are divided first resulting in an integer result and when the result is converted the decimal part is truncated again to 0.0 as a final value in x.

C) Arithmetic Promotion - the explicit conversion of a value from one data type to another due to an arithmetic operation involving mixed data types.

1) In a calculation involving a mixture of integers and decimals the integers are temporarily converted to a decimal and the answer is a decimal.

```
ex.1) y = 2;  
      x = y + 3.5;
```

- a) What is x if x is an integer type? Answer: illegal statement
- b) What is x if x is a double or float type? Answer: 5.5
- c) What is y after the last line? Answer: 2 (not 2.0)

Computer Programming I - Unit 2 Lecture

5 of 13

III. Math methods - predefined (or prewritten) methods that are in the Math class that is included in every Java program because the Math class is found in the **java.lang** package which is automatically imported into all Java programs

A) Basic Methods to Know: (Note: **random** will covered in two pages)

<u>Name</u>	<u>Description</u>	<u>Type of argument</u>	<u>Type of value returned</u>	<u>Example</u>	<u>Result</u>
sqrt	square root	double	double	Math.sqrt(4.0)	2.0
		int	double	Math.sqrt(9)	3.0
		int/double	none	Math.sqrt(-9.0)	error
pow	powers	double	double	Math.pow(2.0,3.0)	8.0
		int	double	Math.pow(5,2)	25.0
abs	absolute value	int	int	Math.abs(-5)	5
		int	int	Math.abs(5)	5
		double	double	Math.abs(-5.3)	5.3
		double	double	Math.abs(5.3)	5.3
ceil	ceiling (round up)	double	double	Math.ceil(5.2)	6.0
		double	double	Math.ceil(5.9)	6.0
		int	double	Math.ceil(5)	5.0
floor	floor (round down)	double	double	Math.floor(5.2)	5.0
		double	double	Math.floor(5.9)	5.0
		int	double	Math.floor(5)	5.0
round	round to integer	double	double	Math.round(5.2)	5.0
		double	double	Math.round(5.9)	6.0
		double	double	Math.round(-5.9)	-6.0
		int	double	Math.round(5)	5.0

(Note: All double's could also be floats.)

Advanced topic:

Math class is a **static** class therefore objects can not be created for it. The methods in Math class are **static** methods. We saw in Unit 1 that to use a method it needs to be accessed by connecting it to an object by the dot operator. Static methods are accessed by connecting them to the name of the static class by the dot operator so no object is needed (nor could one even be created). ex.) Math.sqrt(x);

III. Math methods - continued

- 1) Determine the value of the following Java arithmetic expressions.
(Note: Be careful of the answer's data type.)

ex.1a) `Math.sqrt(25.0)`

ex.1b) `Math.sqrt(25)`

ex.2a) `Math.pow(2.0, 5.0)`

ex.2b) `Math.pow(2, 5)`

ex.3a) `Math.abs(5)`

ex.3b) `Math.abs(-5)`

ex.4a) `Math.abs(3.9)`

ex.4b) `Math.abs(-3.9)`

ex.5a) `Math.ceil(5.1)`

ex.5b) `Math.ceil(5.8)`

ex.6a) `Math.floor(5.1)`

ex.6b) `Math.floor(5.9)`

ex.7a) `Math.round(5.5)`

ex.7b) `Math.round(-5.9)`

ex.8) `Math.sqrt(Math.abs(-36))`

ex.9) `Math.abs(Math.sqrt(pow(3,2)))`

- 2) Convert the following mathematical expressions to a Java arithmetical expression

ex.)
$$\frac{\sqrt{x^5 + y^7}}{2 - d}$$

- 3) Convert the following Java code into an algebraic expression.

ex.) `x = (Math.abs(Math.pow(b,2) - 4.0 * a * c))/(3 * a)`

III. Math methods - continued

A) **random** method (Note: Our blue book does not cover this one.)

`Math.random()` produces a *pseudorandum* random decimal number between 0 (inclusive) and `n` (exclusive)

The following line would generate a random number between *Big* number and *Small* number inclusive:

`(int)(Math.random()*(Big - Small + 1)) + Small` or

`(int)(Math.random()*(Big - Small + 1) + Small)`

ex.1) Generate a random number between 5 and 10 inclusive

Note 1: *Big* is the larger number, 10 in this example

Note 2: *Small* is the smaller number, 5 in this example

Note 3: the formula needs a typecasting of **int**

Answer: `(int)(Math.random()*(10 - 5 + 1) + 5)` or `(int)(Math.random()*(6) + 5)`

ex.2) generate a random number between 15 and 25 inclusive and store in `x`

Answer: `int x = (int)(Math.random()*(11) + 15);`

ex.3) `(int)(Math.random()*(100) + 10)` will produce a random number between what two numbers?

Answer: between 10 and 109 inclusive (work: $100 = \text{last} - 10 + 1$)

IV. Formatting Output

Recall: Given that $a = 1$, $b = 2$; find the Output of the following code fragment

```
c.print("Start" + a + b + "End");
```

or

```
c.print("Start" +  
    a +  
    b +  
    "End");
```

or

```
c.print("Start");  
c.print(a);  
c.print(b);  
c.print("End");
```

OUTPUT:

Start12End

A) Creating Blank Lines - use multiples of either:

1) **println** or

2) **\n** enclosed in quotes (single or double quotes if alone)

Given that $a = 1$, $b = 2$;

```
ex.1) c.println("Start"); c.println( ); c.println( );  
c.println(a);  
c.println(b);  
c.print("End");
```

or

```
ex.2) c.print("Start\n\n\n");  
c.print(a + '\n');  
c.print(b + "\n\n");  
c.print("End");
```

OUTPUT:

```
Start  
-->  
-->  
1  
2  
-->  
End
```

IV. *Formatting Output* - continued

B) Inserting blanks within a line - use space(s),
inside of quote marks, between items

Note: Blank spaces are indicated by a

ex.) Given that $a = 1$, $b = 2$;

```
c.print("Start " + a + " " + " " + " " + b + " " + " " + "End");  
or  
c.print("Start " + a + "  " + b + "  End");
```

Output:

```
Start 1  2 End
```

IV. *Formatting Output* - continued

D) Using Field Widths

1) Formatting Strings - a number after the string will create a left justified partition

ex.1) `c.print("Bob A",7); c.print("G",5);`

Ans: Bob _ A _ _G _ _ _ _

ex.2) `c.print("Bob A",2); c.print("G",5);`

Ans: Bob _ AG _ _ _ _

(Note: If the width is too small it is ignored)

2) Formatting Integers - a number after the integer will create a right justified partition

ex.1) `c.print(123,7); c.print(56,4);` Ans: _ _ _ _ _ _ _ _

Ans: _ _ _ _ 1 2 3 _ _ 5 6

3) Formatting Decimals - the first number after the integer will create a right justified partition and the second number is how many places after the decimal to round to

ex.1) `c.print(123.56,8,1);` Ans: _ _ _ _ _ _ _ _

Ans: _ _ _ 1 2 3 . 6

ex.2) `c.print(123.56,8,4);` Ans: _ _ _ _ _ _ _ _

Ans: 1 2 3 . 5 6 0 0

(Note: If the width is too small it is ignored)

Computer Programming I - Unit 2 Lecture

IV. *Formatting Output* - continued
ex.)

```
c.println("12345678901234567890"); // already done below
c.print("Hey",5);
c.print("Whats Up",12);
c.println("Doc");
c.println(512, 5);
c.print(67.8, 6, 1);
c.println(765.987, 7, 2);
c.println(765.987, 8, 1);
c.println(765.987, 1, 3);
c.println(765.987, 10, 5);
```

Output:

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0

Answer:

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
H	e	y			W	h	a	t	s		U	p					D	o	c
		5	1	2															
		6	7	.	8		7	6	5	.	9	9							
			7	6	6	.	0												
7	6	5	.	9	8	7													
	7	6	5	.	9	8	7	0	0										

*Note: You can not have a string and a number that is formatted in the same print or println statement

ex1) c.println("Hi there" + 67.89, 5, 1); is an error
do:

```
c.print("Hi there ");
c.println(67.89, 5, 1);
```

Output: Hi there 67.9

V. String Class

A) Instantiating (declaring) a string

*Note: A string is really an object of the String class

Method 1: ex.1a) `String x;`
`x = new String();`
`x = "hello there";`

ex.1b) `String x = new String();`
`x = "hello there";`

Method 2: ex.2a) `String x;`
`x = "hello there";`

ex.2b) `String x = "hello there";`

B) String Class Methods (see page 81 of the blue book for others)

1) `toLowerCase()` - returns the string as all lower case letters

2) `toUpperCase()` - returns the string as all upper case letters

3) `length()` - returns the length of the string including spaces etc.

4) `concat(stg2)` - makes one longer string from the two

ex.1)

`String string1 = "WHEATON";`

`String string2 = "warrenville";`

`String string3 = "Wheaton Warrenville South";`

`string1.toLowerCase()` result: wheaton

`string2.toUpperCase()` result: WARRENVILLE

`string3.length()` result: 25

`string1.concat(string2)` result: WHEATONwarrenville

ex.2) `String z = string2.concat(string1);`

`c.print(z);` output: warrenvilleWheaton

Extra:

`System.out.print(5 + 2 + "A");` output: 7A

`System.out.print("A" + (5 + 2));` output: A7

`System.out.print("A" + 5 + 2);` output: A52

Computer Programming I - Unit 2 Lecture

13 of 13

VI. Graphics

A) Graphics

```
import java.awt.*;           // awt stands for the Abstract Window Toolkit
                             /* The .* means that all Classes in this package will
                             be imported and we will not have to list each one */

import hsa.Console;
public class P1ch2lab7
{
    static Console c;
    public static void main(String[ ] args)
    {
        c = new Console( );
        c.setColor(Color.blue);
        c.drawRect(0, 0, 640, 450);
        c.fillRect(0, 0, 640, 450);
        c.setColor(Color.red);
        c.drawOval(100, 100, 50, 50);
        c.fillOval (100, 100, 50, 50);
        c.setColor(Color.green);
        c.drawLine(200, 10, 30, 300);
    }
}
```

VI. Applets

```
import java.applet.*;           //A package containing the Applet Class
import java.awt.*;
public class P1ch2lab8 extends Applet // Needed whenever Applets are used
{                                     // so that it can run in a browser window
    public void paint (Graphics g) // Needed instead of main() when
                                    // doing Applets
    {
        setBackground(Color.cyan); // Does not work on Console screen
        g.setColor (Color.red);
        g.fillRect (0, 0, 100, 100);
    }
}
```