INTRODUCTION TO PROGRAMMING WITH JAVA - CEJV416

Lecture #3

- Arithmetic Operations
- Casting
- Number formatting
- Classes

Binary Operators

- +
  - Additive (also used for String concatenation)
- -
  - Subtraction
- *
  - Multiplication
- /
  - Division
- %
  - Remainder

Instead of

```
count = count + 1;     // count is increased by 1

count = count – 1;     // count is decreased by 1
```

Unary Operators

- +
  - Unary plus
- -
  - Unary minus; negates an expression
- ++
  - Increment; increments a value by 1
- - -
  - Decrement; decrements a value by 1

Now we have

```
++count;        // count is increased by 1

count++;

--count;        // count is decreased by 1

count--;
```

Position of increment/decrement operator

```
int count = 5;
int value;
```

- then

```
value = ++count; // value will be 6
                 // count will be 6
```

- or

```
value = count++; // value will be 5
                 // count will be 6
```

- The position of the increment/decrement operator determines when the operation is carried out
- Left side means first
- Right side means last

Instead of

```
count = count + 6;     // count is increased by 6
count = count – 7;     // count is decreased by 7
total = total + 100.0; // total is increased by 100.0
total = total – 100.0; // total is decreased by 100.0
price = price * .8;    // price is multiplied by .8
sum = sum + nextNumber;// sum is increased by value
                       // of nextNumber
```

Assignment operators

x = x + y is simplified with x += y

- =
  - Assignment
- +=
  - Addition
- -=
  - Subtraction
- *=
  - Multiplication
- /=
  - Division
- %=
  - Modulus

Now we have

```
count += 6;         // count is increased by 6
count -= 7;         // count is decreased by 7
total += 100.0;     // total is increased by 100.0
total -= 100.0;     // total is decreased by 100.0
price *= .8;        // price is multipled by .8
sum += nextNumber;  // sum is increased by the value
                    // of nextNumber
```

Precedence

1. Increment and decrement
2. Positive and negative
3. Multiplication, division, and remainder
4. Addition and subtraction

- When operators of equal precedence appear in the same expression:
  - All binary operators except for the assignment operators are evaluated from left to right
  - Assignment operators are evaluated right to left.

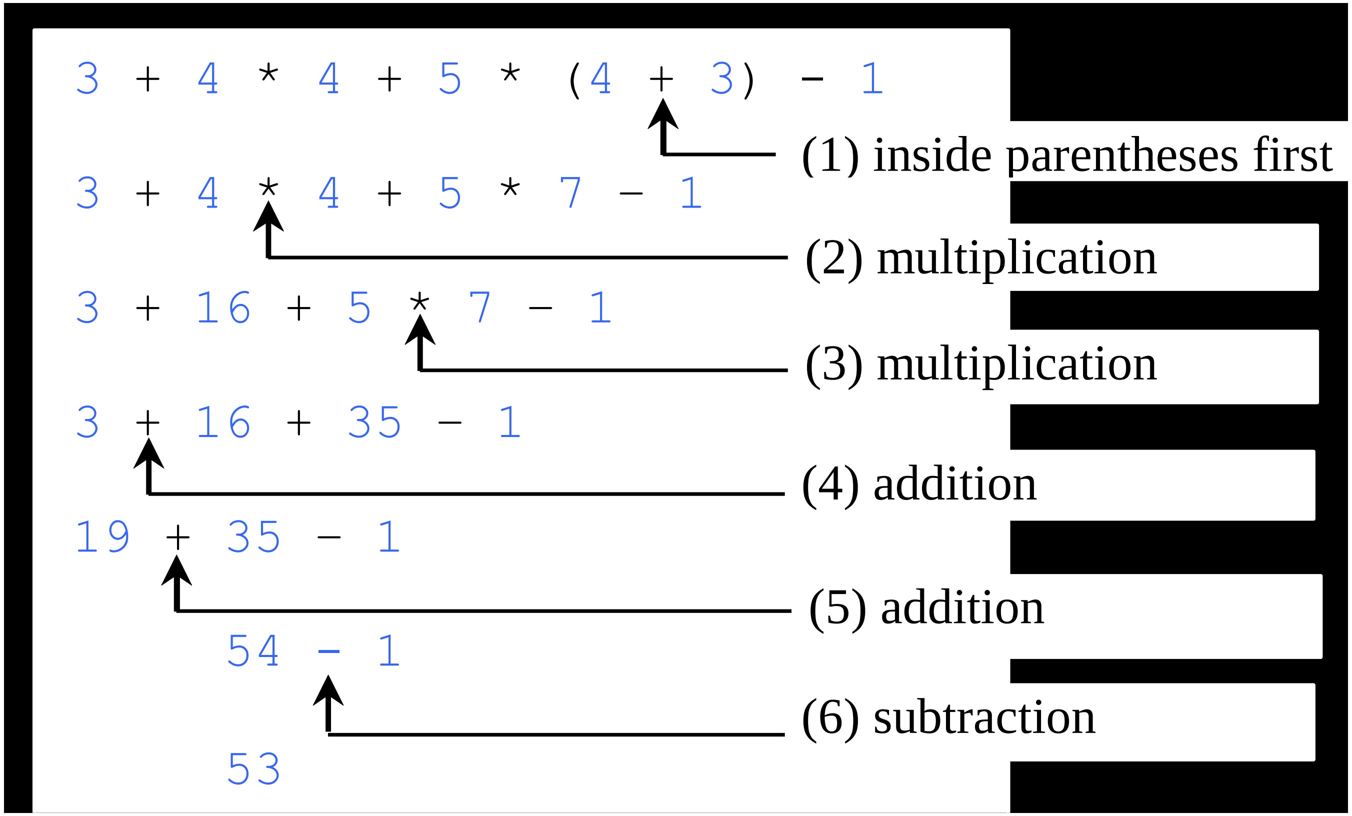
Examples

If x =2 is already declared.

1. ++x*3
2. x++*3
3. 3 + 4 * 4 + 5 * (4 + 3) - 1

1. 9
2. 6
3. 53

Examples



```
3 + 4 * 4 + 5 * (4 + 3) - 1
3 + 4 * 4 + 5 * 7 – 1
3 + 16 + 5 * 7 – 1
3 + 16 + 35 – 1
19 + 35 – 1
    54 - 1
    53
```

(1) inside parentheses first

(2) multiplication

(3) multiplication

(4) addition

(5) addition

(6) subtraction

What happens when types are mixed

- Implicit casting

```
double grade = 93; // convert int to double
double d = 95.0;
int i = 86, j = 91;
double average = (d+i+j)/3;
           // convert i and j to double values
           // average = 90.666666...
```

- Casting from less precise to more precise data types
  - byte→short→int→long→float→double

What happens when types are mixed

- Explicit casting
  - (type) expression

```
int grade = (int) 93.75;
        // convert double to int (grade = 93)

double d = 95.0;
int i = 86, j = 91;
double average = ((int)d+i+j)/3; //very important
      // convert d to int value (average = 90)

double result = (double) i / (double) j;
                 // result has decimal places
```

Exercise 5

Converting Celsius to Fahrenheit

and

Fahrenheit to Celsius

Integer Division

- 5 / 2 yields an integer 2.
- 5.0 / 2 yields a double value 2.5

Casting between char and int

```
char letterChar = 65;
      // convert int to char (letterChar = 'A')
char letterChar2 = (char) 65;
      // this works too

int letterInt = 'A';
      // convert char to int (letterInt = 65)
int letterInt2 = (int) 'A';
      // this works too
```

A word about classes

- From this point forward we will be using existing Java classes and creating our own classes
- A class is the organization of data and code based on the concepts of object oriented programming
- Java has a rich library of classes that we will call upon
- These classes may define how a button works on a form or how to execute an SQL query
- Classes contain methods that perform the work we need done
- In OOP we say that we are sending a message to a class through its methods
- In Structured Programming we say that we are calling a method

Creating classes

- Syntax

```
ClassName objectName = new ClassName(arguments);
```

- Examples

```
// creates a Scanner object named sc
Scanner sc = new Scanner(System.in);

// creates a Date object named now
Date now = new Date();
```

Using methods in a class

- Syntax

```
objectName.methodName(arguments)
```

- Examples

```
// get a double entry from the console
double subtotal = sc.nextDouble();

// convert the date to a string
String currentDate = now.toString();
```

A word about classes

- There are two ways that we can use a class
- This is dependent on how its methods are coded
- Methods called static can be called directly just like a global method in structured programming

```
double d = Math.pow(4,2); // static method
```

- Methods that are non-static require that the class is instantiated first.

```
JButton ok = new JButton();
ok.setText(”OK”);  // non static method
```

- When writing methods non-static is always preferred

Make numbers look nice - formatting

- □ Integers are displayed as they are stored in memory
- □ Floating point numbers display up to their precision
  - Trailing zeros are removed

```
double pi =  3.1415926535897932384626433832795;
System.out.println("pi = " + pi);
```

- □ produces

```
pi = 3.141592653589793
```

NumberFormat class: java.text.NumberFormat

- Three static methods of the NumberFormat class
  - `getCurrencyInstance()`
  - `getPercentInstance()`
  - `getNumberInstance()`
- Three non-static methods of a NumberFormat object
  - `format(anyNumberType)`
  - `setMinimumFractionDigits(int)`
  - `setMaximumFractionDigits(int)`

Using NumberFormat - currency

```
double price = 11.575;
// creates an object with proper formatting based
// on the locale of the use
NumberFormat currency = NumberFormat.getCurrencyInstance();
String priceString = currency.format(price);
// returns $11.58
// if the Locale was CA_fr then returns 11.58$
```

Using NumberFormat - percent

```
double majority = .505;
// creates an object with proper formatting based
// on the locale of the use
NumberFormat percent = NumberFormat.getPercentInstance();
String majorityString = percent.format(majority);
// returns 50%
```

Using NumberFormat - precision

- The number format with one decimal place

```
double miles = 15341.253;
NumberFormat number = NumberFormat.getNumberInstance();
number.setMaximumFractionDigits(1);
String milesString = number.format(miles);
// returns 15,341.3
```

NumberFormat shortcut

- □ Two NumberFormat methods coded in one statement

```
String majorityString =
    NumberFormat.getPercentInstance().format(majority);
```

Exercise 6

Number Formatting

Exercise

Modify your "Celsius to Fahrenheit“ and " Fahrenheit to Celsius " to show exactly two digits for fraction.