Unit 1: Primitive Types
Arithmetic Operations

Adapted from:

1) Building Java Programs: A Back to Basics Approach

by Stuart Reges and Marty Stepp

2) Runestone CSAwesome Curriculum

Expressions

- **expression**: A value or operation that computes a value.

  - Examples:
    ```
    1 + 4 * 5
    (7 + 2) * 6 / 3
    42
    ```

  - – The simplest expression is a *literal value*.
  - – A complex expression can use operators and parentheses.

Arithmetic operators

- **operator**: Combines multiple values or expressions.

  `+`   addition
  `–`   subtraction (or negation)
  `*`   multiplication
  `/`   division
  `%`   modulus (a.k.a. remainder)

- As a program runs, its expressions are *evaluated*.
  - `1 + 1` evaluates to `2`
  - `System.out.println(3 * 4);` prints `12`
    - How would we print the text `3 * 4` ?

Integer division with /

- When we divide integers, the quotient is also an integer.
  - `14 / 4` is `3`, not `3.5`

```
        3                    4                      52
   4 ) 14           10 ) 45                27 ) 1425
       12                40                        135
        2                 5                         75
                                                    54
                                                    21
```

- More examples:
  - `32 / 5` is `6`
  - `84 / 10` is `8`
  - `156 / 100` is `1`
  - Dividing by 0 causes an error when your program runs. This error is also called an **ArithmeticException**.

Integer remainder with %

- The `%` operator computes the remainder from integer division.
  - `14 % 4`   is `2`
  - `218 % 5`   is `3`

```
        3                      43
  4 ) 14                 5 ) 218
      12                     20
       2                     18
                             15
                              3
```

What is the result?
`45 % 6`
`2 % 2`
`8 % 20`
`11 % 0`

- Applications of `%` operator:
  - Obtain last digit of a number: `230857 % 10` is `7`
  - Obtain last 4 digits: `658236489 % 10000` is `6489`
  - See whether a number is odd: `7 % 2` is `1`, `42 % 2` is `0`

Expressions

Find the exact change for 137 cents using quarters, dimes, nickels and cents. Use the least number of coins.

How many quarters? 137 / 25 = 5 quarters (Integer Division!)

What's leftover? 137 % 25 = 12 cents

How many dimes? 12 / 10 = 1 dime

What's leftover? 12 % 10 = 2 cents

How many nickels? 2 / 5 = 0 nickels.

What's leftover? 2 % 5 = 2 cents.

Precedence

- **precedence**: Order in which operators are evaluated.
  - Generally operators evaluate left-to-right.

    `1 - 2 - 3` is `(1 - 2) - 3` which is `-4`

  - But `* / %` have a higher level of precedence than `+ -`

    `1 + ` **`3 * 4`** `               ` is `13`

    `6 + ` **`8 / 2`** ` * 3`
    `6 + ` `    ` **`4`** `     ` **`* 3`**
    `6 + ` `      12         ` is `18`
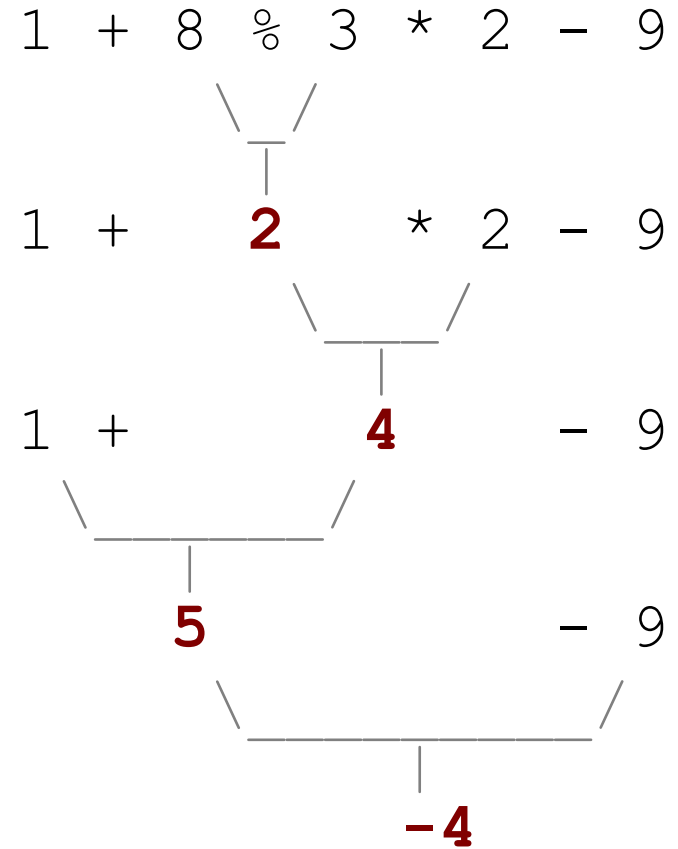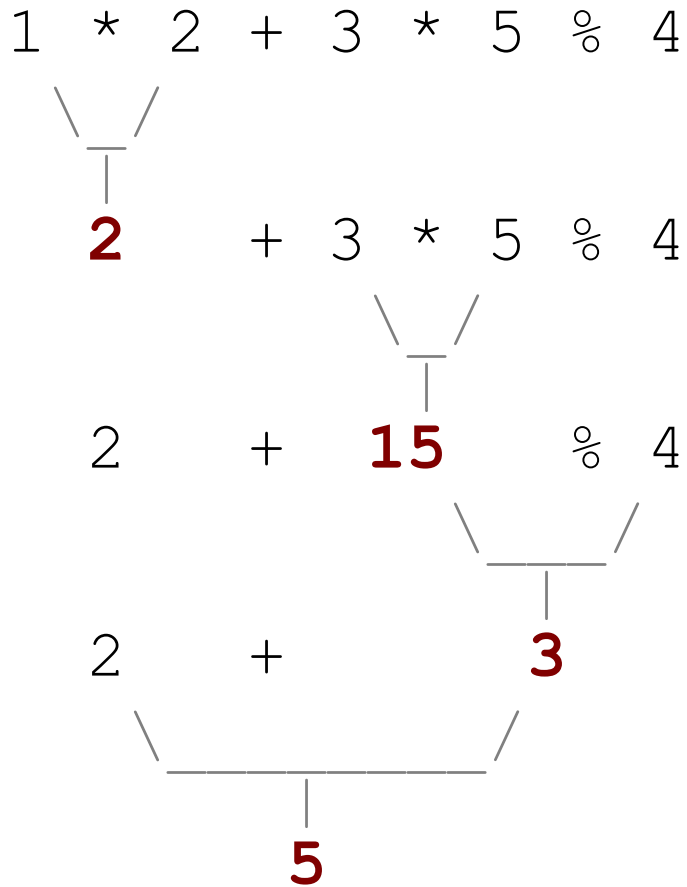
  - Parentheses can force a certain order of evaluation:

    `(1 + 3) * 4          ` is `16`

  - Spacing does not affect order of evaluation

    `1+3 * 4-2              ` is `11`

1 * 2 + 3 * 5 % 4

2   + 3 * 5 % 4

2   + **15**   % 4

2   +   **3**

5

1 + 8 % 3 * 2 - 9

1 +   **2**   * 2 - 9

1 +   **4**   - 9

5   - 9

-4

Real numbers (type `double`)

- Examples: `6.022, -42.0, 2.143`
  - Placing `.0` or `.` after an integer makes it a `double`.

- The operators `+ - * / % ()` all still work with `double`.
  - `/` produces an exact answer: `15.0 / 2.0` is `7.5`
  - Precedence is the same: `()` before `* / %` before `+ -`

Real number example

```
2.0 * 2.4 + 2.25 * 4.0 / 2.0
     \_____/
        |
      4.8     + 2.25 * 4.0 / 2.0
                    \_____/
                       |
      4.8     +       9.0    / 2.0
                          _____/
                             |
      4.8     +              4.5
           _____/
                    |
                   9.3
```
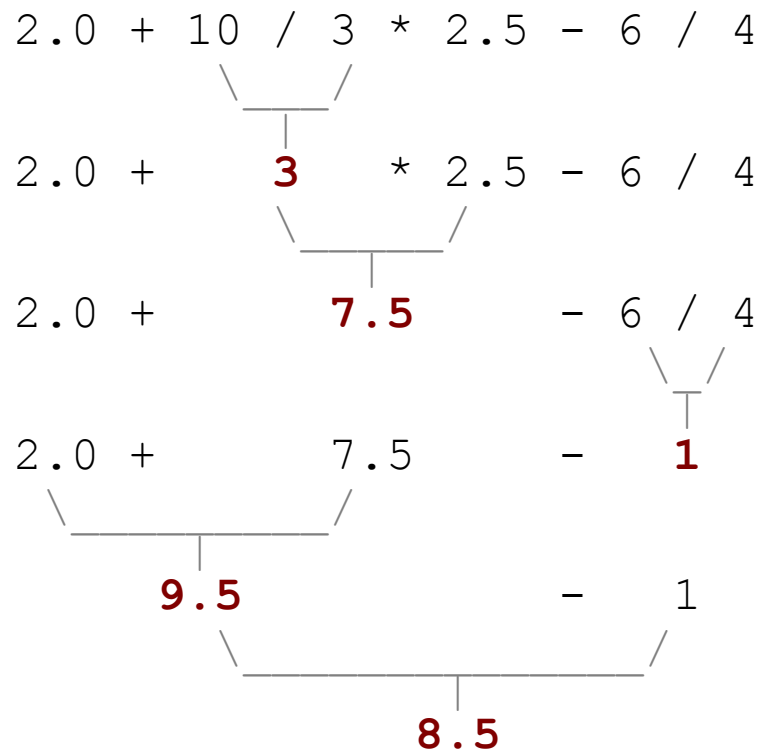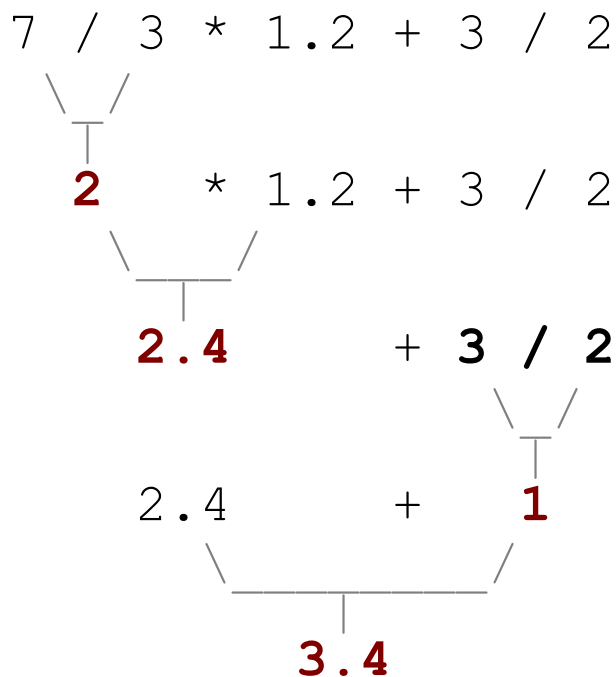
Mixing types

- When `int` and `double` are mixed, the result is a `double`.
  - `4.2 * 3` is `12.6`

- The conversion is per-operator, affecting only its operands.

```
7 / 3 * 1.2 + 3 / 2
    \ /

    2     * 1.2 + 3 / 2
      \  /

        2.4       + 3 / 2
                     \ /

        2.4       +    1
          \          /

              3.4
```

```
2.0 + 10 / 3 * 2.5 - 6 / 4
        \ /

2.0 +     3    * 2.5 - 6 / 4
          \      /

2.0 +        7.5      - 6 / 4
                        \ /

2.0 +        7.5      -    1
  \                      /

     9.5                -    1
       \                   /

              8.5
```

  - `3 / 2` is `1` above, not `1.5`.

Mixing types

- In Java, **type casting** is used to convert variables from one type to another.

```
public class Test{
    public static void main(String[] args){
        System.out.println(1 / 3);
        System.out.println(1.0 / 3);
        System.out.println(1 / 3.0);
        System.out.println((double) 1 / 3);
    }
}
0
0.3333333333333333
0.3333333333333333
0.3333333333333333
```

Type casting

- **type cast**: A conversion from one type to another.
  - To promote an `int` into a `double` to get exact division from `/`
  - To truncate a `double` from a real number to an integer

- Syntax:

  (**type**) **expression**

  Examples:
  ```
  double result = (double) 19 / 5;      // 3.8
  int result2 = (int) result;           // 3
  int x = (int) Math.pow(10, 3);        // 1000
  ```

More about type casting

- Type casting has high precedence and only casts the item immediately next to it.

  ```
  – double x = (double) 1 + 1 / 2;        // 1.0
  – double y = 1 + (double) 1 / 2;        // 1.5
  ```

- You can use parentheses to force evaluation order.

  ```
  – double average = (double) (a + b + c) / 3;
  ```

- A conversion to `double` can be achieved in other ways.

  ```
  – double average = 1.0 * (a + b + c) / 3;
  ```

Round to the nearest integer

- casting can be used to round a number to its nearest integer .

```
double number = 10 / 3;
// round a positive number to its nearest integer
int nearestInt = (int)(number + 0.5);
double negNumber = -10 / 3;
// round a positive number to its nearest integer
int nearestNegInt = (int)(negNumber – 0.5);

What is the value of nearestInt and nearestNegInt?
Answer: 3 and -3
```

Increment and decrement

shortcuts to increase or decrease a variable's value by 1

Shorthand	Equivalent longer version
variable++;	**variable** = **variable** + 1;
variable--;	**variable** = **variable** - 1;

```
int x = 2;
x++;                          // x = x + 1;
                              // x now stores 3

double gpa = 2.5;
gpa--;                        // gpa = gpa - 1;
                              // gpa now stores 1.5
```

Modify-and-assign

shortcuts to modify a variable's value

Shorthand	Equivalent longer version
variable += **value**;	**variable** = **variable** + **value**;
variable -= **value**;	**variable** = **variable** - **value**;
variable *= **value**;	**variable** = **variable** * **value**;
variable /= **value**;	**variable** = **variable** / **value**;
variable %= **value**;	**variable** = **variable** % **value**;

```
x += 3;                    // x = x + 3;

gpa -= 0.5;                // gpa = gpa - 0.5;

number *= 2;               // number = number * 2;
int x = 0; int y = 5; int z = 1; x++; y -= 3; z = x + z; x = y * z; y
    %= 2; z--;
```

Code Tracing

What are the values of x, y and z after tracing through the following code?

```
int x = 0;
int y = 5;
int z = 1;
x++;
y -= 3;
z = x + z;
x = y * z;
y %= 2;
z--;
```

Lab 1

- Let {a1,a2,a3,…,an} be a list of n real numbers.

- The average of the list is **ave** = (a1+a2+…+an)/n.

- The variance of the list =
  [(a1 - ave)^2 + (a2 - ave)^2 + … + (an - ave)^2] / n.

- The standard deviation of the list = the square root of the variance of the list.

Lab 1

For example, if the list is {2,4,5,8,16}.

Average=7.0

Variance=[(-5.0)^2+(-3.0)^2+(-2.0)^2+1.0^2+9.0^2]/5=24.0
Standard deviation=square root of 24.0=4.898979486

Lab 1

Create a new repl on repl.it and follow the comments below to write a program that compute some statistics.

```java
public class Statistics
{
    public static void main(String[] args)
    {
// 1. Declare 3 int variables for grades and initialize them to 3 values
// 2. Declare an int variable for the sum of the grades
// 3. Declare a double variable for the average of the grades
// 4. Write a formula to calculate the sum of the 3 grades
// 5. Write a formula to calculate the average of the 3 grades from the
//     sum using division and type casting.
// 6. Print out the average
// 7. Declare a double variable and calculate the variance
// 8. Declare a double variable to compute the standard deviation.
    }
}
```

Lab 2

Use the following template(or something similar) to write a program that gives exact change with the least number of coins for a given number of cents. **Use intermediate variables to help your calculation.**

```
public static void main(String[] args){
    int totalCents = 137;  //137 can be any number
    .....
    // your code here.

}
```

Output: 5 quarters, 1 dimes, 0 nickels, 2 pennies.