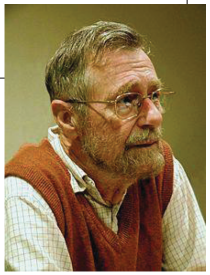
Topic 4
Expressions and variables

> "Once a person has understood the way variables are used in programming, he has understood the quintessence of programming."
>
> *-Professor Edsger W. Dijkstra*



Based on slides bu Marty Stepp and Stuart Reges
from http://www.buildingjavaprograms.com/

Data and expressions

reading: 2.1

The computer’s view

- Internally, computers store everything as 1’s and 0’s
  - Example:

```
h      → 01101000
"hi"   → 0110100001101001
104    → 01101000
```

- How can the computer tell the difference between an `h` and `104`?
- **type**: A category or set of data values.
  - Constrains the operations that can be performed on data
  - Many languages ask the programmer to specify types
  - Examples: integer, real number, string
- Binary Numbers

Java's primitive types

- **primitive types**: 8 simple types for numbers, text, etc.
  - Java also has **object types**, which we'll talk about later

Name	**Description**		**Examples**
`int`	integers	(up to 2^{31} - 1)	`42, -3, 0, 926394`
`double`	real numbers	(up to 10^{308})	`3.1, -0.25, 9.4e3`
`char`	single text characters		`'a', 'X', '?', '\n'`
`boolean`	logical values		`true, false`

- Why does Java distinguish integers vs. real numbers?

Integer or real number?

- Which category is more appropriate?

integer (`int`)	real number (`double`)

1. Temperature in degrees Celsius
2. The population of lemmings
3. Your grade point average
4. A person's age in years
5. A person's weight in pounds
6. A person's height in meters
7. Number of miles traveled
8. Number of dry days in the past month
9. Your locker number
10. Number of seconds left in a game
11. The sum of a group of integers
12. The average of a group of integers

- credit: Kate Deibel, http://www.cs.washington.edu/homes/deibel/CATs/

Clicker question

- What is best choice for data type?

CHOICE	Number of days it rained in year	Sum of group of integers	Average of group of integers
A	`int`	`int`	`double`
B	`int`	`int`	`int`
C	`double`	`int`	`int`
D	`double`	`int`	`double`
E	`int`	`double`	`double`

Expressions

- **expression**: A combination of values and / or operations that results (via computation) in a value.
  - Examples: `1 + 4 * 5`

    `(7 + 2) * 6 / 3`

    `42`

    `"Hello, world!"`
  - 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 with integer division. Try floating point division by 0.

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

Clicker question

- What does each expression evaluate to?

CHOICE	13 % 5	5 % 13	30 % 5
A	3	3	0
B	3	5	0
C	2	5	5
D	2	13	6
E	2.4	13	6

Clicker question

- What does the following expression evaluate to?

`1017 % 100 + (12 % 100)`

A. 10

B. 17

C. 12

D. 22

E. 29

Remember PEMDAS?

- **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`

Precedence examples

```
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
```

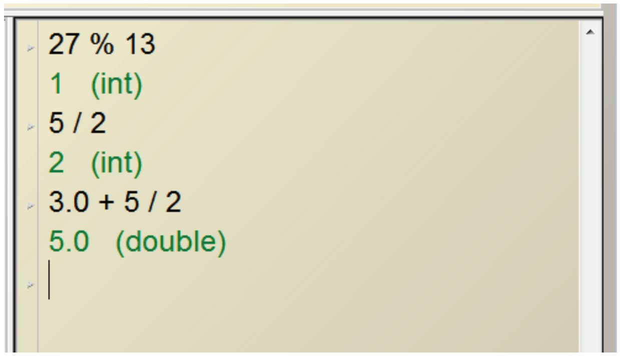
Precedence questions

- What values result from the following expressions?

```
9 / 5
695 % 20
7 + 6 * 5
7 * 6 + 5
248 % 100 / 5
6 * 3 - 9 / 4
(5 - 7) * 4
6 + (18 % (17 - 12))
```

Practice!!

- BlueJ includes a *Code Pad*
  - *View -> Show Code Pad*
- *read - eval - print* loop
- Useful to try various expressions



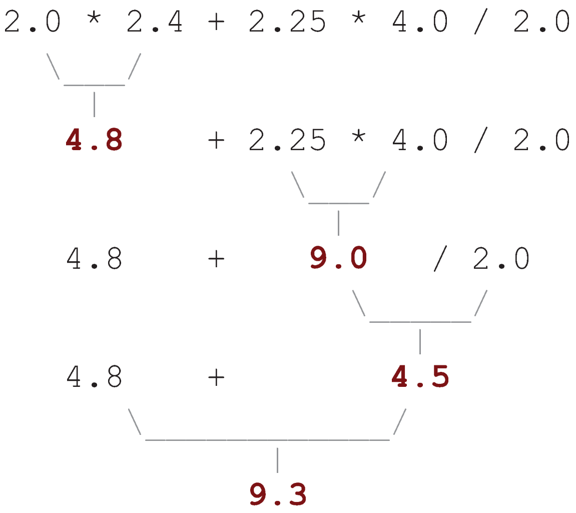
```
27 % 13
1   (int)
5 / 2
2   (int)
3.0 + 5 / 2
5.0   (double)
```

Real numbers (type `double`)

- Examples: `6.022` , `-42.0` , `2.143e17`
  - 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
```

Precision in real numbers

- The computer internally represents real numbers in an imprecise way.

- Example:

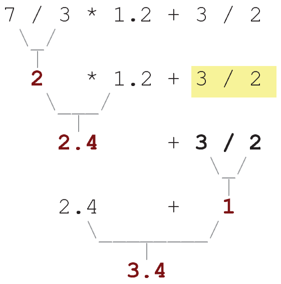
```
System.out.println(0.1 + 0.2);
```

  - The output is `0.30000000000000004`!

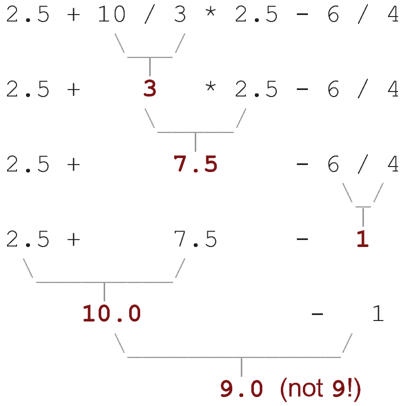
Mixing types

- 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
```



```
2.5 + 10 / 3 * 2.5 - 6 / 4
2.5 +      3 * 2.5 - 6 / 4
2.5 +        7.5   - 6 / 4
2.5 +        7.5   -   1
      10.0         -   1
            9.0 (not 9!)
```

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

String concatenation

- **string concatenation**: Using + between a string and another value to make a longer string.

```
"hello" + 42  is "hello42"
1 + "abc" + 2 is "1abc2"
"abc" + 1 + 2 is "abc12"
1 + 2 + "abc" is "3abc"
"abc" + 9 * 3 is "abc27"
"1" + 1       is "11"
4 - 1 + "abc" is "3abc"
```

- Use + to print a string and an expression's value together.

```
System.out.println("Grade: " + (95.1 + 71.9) / 2);
```

  - Output: `Grade: 83.5`

What does the following expression evaluate to?

1.25 + 7 / 4 + "CS" + 3 + 4

A. "3.0CS34"

B. "2.25CS7"

C. "2CS7"

D. "2.25CS34"

E. Something other than A - D

Variables

reading: 2.2

Receipt example

What's bad about the following code?

```
public class Receipt {
    public static void main(String[] args) {
        // Calculate total owed, assuming 8% tax / 15% tip
        System.out.println("Subtotal:");
        System.out.println(38 + 40 + 30);
        System.out.println("Tax:");
        System.out.println((38 + 40 + 30) * .08);
        System.out.println("Tip:");
        System.out.println((38 + 40 + 30) * .15);
        System.out.println("Total:");
        System.out.println(38 + 40 + 30 +
                           (38 + 40 + 30) * .08 +
                           (38 + 40 + 30) * .15);
    }
}
```

- The subtotal expression `(38 + 40 + 30)` is repeated
- So many `println` statements

Variables

- **variable**: A piece of the computer's memory that is given a name and type, and can store a value.
  - Like preset stations on a car stereo, or cell phone speed dial:

  - Steps for using a variable:
    - *Declare* it - state its name and type
    - *Initialize* it - store a value into it
    - *Use* it - print it or use it as part of an expression

Declaration

- **variable declaration**: Sets aside memory for storing a value.
  - Variables must be declared before they can be used.

- Syntax:

  <type> ***<name>***;

```
– int x;
```

x	

myGPA	

```
– double myGPA;
```

Assignment

- **assignment**: Stores a value into a variable.
  - The value can be an expression; the variable stores its result.

- Syntax:

  <name> = ***<expression>***;

x	3

myGPA	3.25

```
int x;
x = 3;  // or int x = 3;

double myGPA;
myGPA = 1.0 + 2.25;  // or double myGPA = 3.25
```

Declaration/initialization

- A variable can be declared/initialized in one statement.

- Syntax:

  <type> ***<name>*** = ***<expression>***;

x	14

myGPA	3.95

```
int x = (11 % 3) + 12;

double myGPA = 3.95;
```

Using variables

- Once given a value, a variable can be used in expressions:

```
int x = 3;
System.out.println("x is " + x);       // x is 3
System.out.println(5 * x - 1);         // 14
```

- You can assign a value more than once:

```
int x = 3;
System.out.println(x + " here");       // 3 here

x = 4 + 7;
System.out.println("now x is " + x);  // now x is 11
```

x	11

Assignment vs. algebra

- Assignment uses = , but it is not an algebraic equation.
  - = means, *"store the value at right in variable at left"*
  - `x = 3;` means, *"x becomes 3"* or *"x should now store 3"*

- **ERROR**: `3 = 1 + 2;` is an illegal statement, because 3 is not a variable.

- What happens here?

```
int x = 3;
x = x + 2;    // ???
```

x	5

Assignment exercise

- What is the output of the following Java code?

```
int x = 3;
int y = x; // y stores 3
x = 5; // x now stores 5
y = y + x;
System.out.println(x + " " + y);
```

A: "5 8"   B: 5 10   C: 10 10

D: 5 + 10   E: 5 8

Swapping the Contents of Two Variables

- Output of this code?

```
int x = 12;
int y = 32;
x = y;
y = x;
System.out.println(x + " " + y);
```

- Output of this code?

```
int x = 12;
int y = 32;
int t = x;
x = y;
y = t;
System.out.println(x + " " + y + " " + t);
```

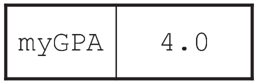
Assignment and types

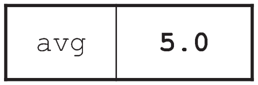
- A variable can only store a value of its own type.

```
int x = 2.5;    // ERROR: incompatible types
```

- An `int` value can be stored in a `double` variable.
  – The value is converted into the equivalent real number.

```
double myGPA = 4;
```


myGPA 4.0


avg 5.0

```
double avg = 11 / 2;
```

Why does `avg` store `5.0` and not `5.5` ?

Compiler errors

- A variable can't be used until it is assigned a value.

```
int x;
System.out.println(x);// ERROR: x has no value
```

- You may not declare the same variable twice (in the same block of code. methods for now.)

```
int x;
int x;          // ERROR: x already exists

int x = 3;
int x = 5;      // ERROR: x already exists
```

- How can this code be fixed?

Printing a variable's value

- Use `+` to print a string and a variable's value on one line.

```
double grade = (95.1 + 71.9 + 82.6) / 3.0;
System.out.println("Your grade was " + grade);

int students = 11 + 17 + 4 + 19 + 14;
System.out.println("There are " + students +
       " students in the course.");
```

- Output:

```
Your grade was 83.2
There are 65 students in the course.
```

Example Problem - Day of Week

- For the Gregorian Calendar
- Given month, day, and year, calculate day of week
- months, 1 = January, 2 = February, … 12 = December

y = year - (14 - month) / 12

x = y + y / 4 - y / 100 + y / 400

m = month + 12 * ((14 - month) / 12) - 2

d = (day + x + (31 * m) / 12) % 7

0 = Sunday, 1 = Monday, 2 = Tuesday

Receipt question

Improve the receipt program using variables.

```
public class Receipt {
    public static void main(String[] args) {
        // Calculate total owed, assuming 8% tax / 15% tip
        System.out.println("Subtotal:");
        System.out.println(38 + 40 + 30);

        System.out.println("Tax:");
        System.out.println((38 + 40 + 30) * .08);

        System.out.println("Tip:");
        System.out.println((38 + 40 + 30) * .15);

        System.out.println("Total:");
        System.out.println(38 + 40 + 30 +
                           (38 + 40 + 30) * .15 +
                           (38 + 40 + 30) * .08);
    }
}
```

Receipt answer

```
public class Receipt {
    public static void main(String[] args) {
        // Calculate total owed, assuming 8% tax / 15% tip
        int subtotal = 38 + 40 + 30;
        double tax = subtotal * .08;
        double tip = subtotal * .15;
        double total = subtotal + tax + tip;

        System.out.println("Subtotal: " + subtotal);
        System.out.println("Tax: " + tax);
        System.out.println("Tip: " + tip);
        System.out.println("Total: " + total);
    }
}
```