“There are a lot of people who don't get along with symbols, so they use words .... As far as I'm concerned, a word is a symbol, it's just a little bit longer.”

Rear Admiral Grace Hopper, Ph.D.

(George Still, “Hopper: We're at the beginning”, PENTAGRAM, 15 Dec 1983, 1)
Data and expressions

reading: 2.1
The computer’s view

- Internally, computers store everything as 1’s and 0’s
  - Example:
    - h → 01101000
    - "hi" → 011010000110100100000000
    - 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, characters, etc.
  - Java also has **object types**, which we'll talk about later

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>integers (up to $2^{31} - 1$)</td>
<td>42, -3, 0, 926394</td>
</tr>
<tr>
<td>double</td>
<td>real numbers (up to $10^{308}$)</td>
<td>3.1, -0.25, 9.4e3</td>
</tr>
</tbody>
</table>
| char     | single text characters       | 'a', 'X', '?', '
'          |
| boolean  | logical values               | true, false                |

- Why does Java distinguish integers vs. real numbers?
Integer or real number?

- Which category is more appropriate?

<table>
<thead>
<tr>
<th>integer (int)</th>
<th>real number (double)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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?

<table>
<thead>
<tr>
<th>CHOICE</th>
<th>Number of days it rained in year</th>
<th>Sum of group of integers</th>
<th>Average of group of integers</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>int</td>
<td>int</td>
<td>double</td>
</tr>
<tr>
<td>B</td>
<td>int</td>
<td>int</td>
<td>int</td>
</tr>
<tr>
<td>C</td>
<td>double</td>
<td>int</td>
<td>int</td>
</tr>
<tr>
<td>D</td>
<td>double</td>
<td>int</td>
<td>double</td>
</tr>
<tr>
<td>E</td>
<td>int</td>
<td>double</td>
<td>double</td>
</tr>
</tbody>
</table>
Expressions

- **expression**: A combination of values and/or operations that results (via computation) in a value.
  
  - Examples: \( 1 + 4 \times 5 \)
    
    \( (7 + 2) \times 6 / 3 \)
    
    42
    
    "Hello, world!"

  - The simplest expression is a *literal value*.
  - A complex expression uses 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`?
When we divide integers, the quotient is also an integer.

\[ 14 \div 4 \text{ is } 3, \text{ not } 3.5 \]

More examples:
- \( 32 \div 5 \) is 6
- \( 84 \div 10 \) is 8
- \( 156 \div 100 \) is 1

Dividing by 0 causes an error when your program runs with integer division. Try floating point division by 0.
The % operator computes the remainder from integer division.

14 % 4  is 2
218 % 5  is 3

4 ) 14
  12
  2

5 ) 218
  20
  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
What does each expression evaluate to?

<table>
<thead>
<tr>
<th>CHOICE</th>
<th>13 % 5</th>
<th>5 % 13</th>
<th>30 % 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>13</td>
<td>6</td>
</tr>
<tr>
<td>E</td>
<td>2.4</td>
<td>13</td>
<td>6</td>
</tr>
</tbody>
</table>
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 $\times / \div$ have a higher level of precedence than $+ -$
    - $1 + 3 \times 4$ is $13$
    - $6 + 8 / 2 * 3$
      - $6 + 4 * 3$
      - $6 + 12$ is $18$
  - Parentheses can force a certain order of evaluation:
    - $(1 + 3) \times 4$ is $16$
  - Spacing does not affect order of evaluation
    - $1 + 3 \times 4 - 2$ is $11$
Precedence examples

\[
\begin{align*}
1 \times 2 + 3 \times 5 \& 4 \\
& 2 + 15 \& 4 \\
& 2 + 3 \\
& 5
\end{align*}
\]

\[
\begin{align*}
1 + 8 \div 3 \times 2 - 9 \\
& 1 + 2 \times 2 - 9 \\
& 1 + 4 - 9 \\
& 5 - 9 \\
& -4
\end{align*}
\]
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

\[\begin{align*}
\text{4.8} &\quad +\quad 2.25 \times 4.0 \div 2.0 \\
\text{4.8} &\quad +\quad 9.0 \div 2.0 \\
\text{4.8} &\quad +\quad 4.5 \\
\text{9.3}
\end{align*}\]
Precision in real numbers

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

Example:

```java
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` is `3.4`
  - `2.5 + 10 / 3 * 2.5 - 6 / 4` is `9.0` (not `9!`)
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

https://cs-comics.com

“Ingredients in a program are like boxes” through

“Primitive data types”
What's bad about the following code?

```java
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
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
I HEREBY DECLARE THAT A VARIABLE OF TYPE 'INTEGER' WITH NAME 'HEIGHT' SHALL BE CREATED!
Declaration

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

- Syntax:
  - `<type> <name>;

- int x;

- double myGPA;
height = 6
Assignment

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

- Syntax:
  ```
  <name> = <expression>;
  ```

```c
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:

  \[
  \text{<type> } \text{<name>} = \text{<expression>};
  \]

  
  \[
  \begin{array}{|c|c|}
  \hline
  \text{x} & 14 \\
  \hline
  \text{myGPA} & 3.95 \\
  \hline
  \end{array}
  \]

  
  int x = (11 % 3) + 12;

  double myGPA = 3.95;
Using variables

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

```java
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:

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

x = 4 + 7;
System.out.println("now x is " + x);  // now x is 11
```
YOU'LL NEVER SEE ME AGAAAAAL\in...

height = 7

OUTTA MY WAY!

(c) 2018 Chand T. John
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; // ???
  ```
What is the output of the following Java code?

```java
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.
  
  ```java
  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.
  
  ```java
  double myGPA = 4;
  ```

<table>
<thead>
<tr>
<th>myGPA</th>
<th>4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg</td>
<td>5.0</td>
</tr>
</tbody>
</table>

  ```java
  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.

```java
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.)

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

```java
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.

```java
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.
Program

Variables
Integer a

Instructions
a = 10
print("a is", a)

(a) 2018 Chand T. John
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

\[ y = \text{year} - (14 - \text{month}) / 12 \]
\[ x = y + y / 4 - y / 100 + y / 400 \]
\[ m = \text{month} + 12 * ((14 - \text{month}) / 12) - 2 \]
\[ d = (\text{day} + x + (31 * m / 12)) \mod 7 \]
0 = Sunday, 1 = Monday, 2 = Tuesday
Receipt question

Improve the receipt program using variables.

```java
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);
    }
}
```
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);
    }
}