Expressions and Control Statements

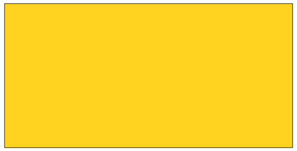
Recap From Last Time

Variables

- A **variable** is a location where a program can store information for later use.

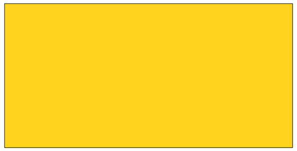
Variables

- A **variable** is a location where a program can store information for later use.



Variables

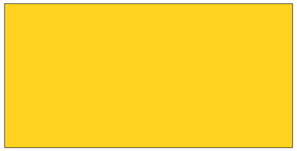
- A **variable** is a location where a program can store information for later use.



- Each variable has three pieces of information associated with it:

Variables

- A **variable** is a location where a program can store information for later use.



- Each variable has three pieces of information associated with it:
  - **Name**: What is the variable called?

Variables

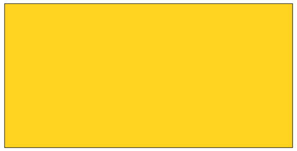
- A **variable** is a location where a program can store information for later use.

`numVoters`

- Each variable has three pieces of information associated with it:
  - **Name**: What is the variable called?

Variables

- A **variable** is a location where a program can store information for later use.



`numVoters`

- Each variable has three pieces of information associated with it:
  - **Name**: What is the variable called?
  - **Type**: What sorts of things can you store in the variable?

Variables

- A **variable** is a location where a program can store information for later use.

`int numVoters`

- Each variable has three pieces of information associated with it:
    - **Name**: What is the variable called?
    - **Type**: What sorts of things can you store in the variable?

Variables

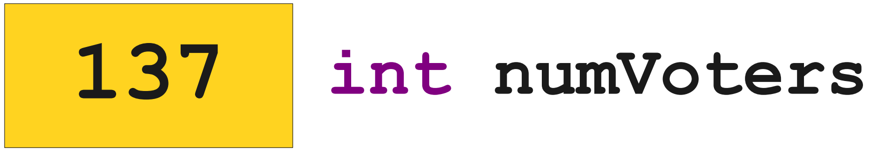
- A **variable** is a location where a program can store information for later use.

`int numVoters`

- Each variable has three pieces of information associated with it:
  - **Name**: What is the variable called?
  - **Type**: What sorts of things can you store in the variable?
  - **Value**: What value does the variable have at any particular moment in time?

Variables

- A **variable** is a location where a program can store information for later use.



137

`int numVoters`

- Each variable has three pieces of information associated with it:
  - **Name**: What is the variable called?
  - **Type**: What sorts of things can you store in the variable?
  - **Value**: What value does the variable have at any particular moment in time?

Expressions

Expressions

- Variables and other values can be used in **expressions**.
- Some familiar mathematical operators:
  - `+` (addition)
  - `-` (subtraction)
  - `*` (multiplication)
  - `/` (division)

The Mod Operator

- The special operator `%` (called the **modulus operator** or **mod operator**) computes the remainder of one value divided by another.
- *a* `%` *b* is pronouned “*a* mod *b*.”
- For example:
    - `15 %  3 =  0`
    - `14 %  8 =  6`
    - `21 %  2 =  1`
    - `14 % 17 = 14`

Operator Precedence

- Java's mathematical operators have the following precedence:

`()`	*(highest)*
`* / %`	
`+ -`	*(lowest)*

- Operators of equal precedence are evaluated left-to-right.

“Fun” with Division

Rounding Down

- In Java, dividing two `int`s will divide and then round down.
- Dividing two `double`s will do the division correctly.

Rounding Down

- In Java, dividing two `int`s will divide and then round down.
- Dividing two `double`s will do the division correctly.
- If either operand is a `double`, the division will be done correctly.
- Option 1: use a **typecast**:

```
double average = (double)(n1 + n2) / 2;
```

- Option 2: divide by 2.0:

```
double average = (n1 + n2) / 2.0;
```

Fun with Division

She got more than me!

butterfly kisses

Cookies for everyone!
butterfly kisses

Some Announcements

Announcements

- Programming Assignment #1 Out:
  - Karel the Robot: Due Friday at 3:15 PM.
    - Submissions now open.
  - Email: Due Sunday, January 19 at 11:59PM.
    - Remember to include your SL!
  - Need help?
    - Stop by the LaIR!
    - Stop by our office hours!
    - Ask your section leader!
- Sections start today!
- Ready to start coding in Java? Check out the **Blank Java Project** link on the CS106A website!

CS Casual Dinner

- Casual dinner for women studying computer science next **Wednesday, January 22** on the Gates fifth floor.
- Wonderful event, and everyone is welcome!
- RSVP appreciated at **this link**, also sent out earlier today.

Back to Java!

A Useful Shorthand

- Commonly, programs contain code like this:

```
x = x + 1;                 y = y * 137;

z = z / 14;                w = w - 3;
```

A Useful Shorthand

- Commonly, programs contain code like this:

```
x = x + 1;                  y = y * 137;
z = z / 14;                 w = w - 3;
```

- The statement

variable = ***variable op value*** ;

can be rewritten as

variable op= ***value*** ;

A Useful Shorthand

- Commonly, programs contain code like this:

```
x += 1;                        y *= 137;
z /= 14;                       w -= 3;
```

- The statement

  variable = ***variable op value*** ;

  can be rewritten as

  variable op= ***value*** ;

Another Useful Shorthand

- In the special case of writing

```
variable = variable + 1;
```

  we can instead write

```
variable++;
```

- In the special case of writing

```
variable = variable – 1;
```

  we can instead write

```
variable--;
```

Control Statements Revisited

Control Statements

```
if
for
while
```

Control Statements

```
if
for
while
```

`if` statements

`if (`*condition*`) {`
… statements to run if condition holds …
`}`

Boolean Expressions

- A **boolean expression** is a test for a condition (it is either **`true`** or **`false`**).
- Value comparisons:

`==`	"equals"	*(note: not single* `=`*)*
`!=`	"not equals"	*(cannot say* `<>`*)*
`>`	"greater than"	
`<`	"less than"	
`>=`	"greater than or equal to"	
`<=`	"less than or equal to"	

Based on slides by Mehran Sahami

Logical Operators

- We use **Logical operators** combine or modify boolean values.
- Logical **NOT**: **`!p`**

```
if (!safeToGoOutside()) {
    takeThis();
}
```

- Logical **AND**: **`p && q`**

```
if (youreHappy() && youKnowIt()) {
    clapYourHands();
}
```

- Logical **OR**: **`p || q`** *(inclusive OR)*

```
if (hasPuppy() || hasKitty()) {
    beHappy();
}
```

- Order of precedence given above.

Short-Circuit Evaluation

- Cute observations:
  - `true || p` is always `true`.
  - `false && p` is always `false`.
- The logical operators **short-circuit**: if the answer is known from the left operand, the right side is not computed.
- Example: The code

```
boolean b = (x == 0) || ((y / x) < 20)
```

will never divide by zero.

Based on slides by Mehran Sahami

Or else

```
if (condition) {
… statements to run if condition holds …
} else {
… statements to run if condition doesn't hold …
}
```

Cascading `if`

```
if (score >= 90) {
  println(" AWWWW YEAHHHHH ");
} else if (score >= 80) {
  println(" <(^_^)> ");
} else if (score >= 70) {
  println(" : - | ");
} else if (score >= 60) {
  println(" ಠ_ಠ ");
} else {
  println(" (╯°□° )╯︵ ┻━┻ ");
}
```

Based on slides by Mehran Sahami