Identifiers, Variables, Operators

Colorado State University
Computer Science Department
Slides Originally Created by Albert Lionelle (Albert.Lionelle@colostate.edu)updated by Marcia Moraes (marcia.moraes@colostate.edu)

Colorado State University

Announcements

- Reminder – readings are due **before** lecture
  - You don't have to do all of it - challenge problems can be challenging…
  - You can return to them.
  - We start off lecture with a quiz from your reading!

Todo:
Busy Week! (readings + labs)
Lab projects start!
Remember: build a habit of doing a little every night!

Recall Activity

- Individually
  - Grab a paper and write at least three concepts that you can remember from your readings
- With your neighbor(s)
  - Discuss what each other could remember. Did you remember the same things? What did you learn from each other?
- Turn you paper to the TAs or myself at the end of the class, this will count as your participation activity for this lecture
- Don’t forget to write your name as it is in our Canvas course!

Reading Check-in

Which of the following are considered *primitives* in Java?

A. int

B. double

C. String

D. char

E. System

Types

- TYPE tells the computer how much room to save!
- int
  - Whole numbers only
  - 1, 2, 3, 1000
- double
  - Floating point numbers
  - 1.0, 2.5, 3.33333, 1000 (which is 1000.0)
- char
  - Every character on a keyboard- stored as int
- boolean
  - true or false
- String
  - collection of ordered characters
  - It is more unique (Object)

Variables

- Identifiers are WORDs
  - You use the to *hold* information
  - Cannot be a reserved word
  - Cannot start with numbers or special characters outside of underscore
  - Use real words! **int x** doesn't mean much, but **int puppyCounter** - has meaning and readable!
- **Declaring** Variables:
  - <TYPE> <IDENTIFIER>;
  - Can be declared in the same line
  - Declaring **reserves** or **allocates** memory! But doesn't store!

```
int myInt;
double myDouble;
int x, y, z;
double dbl, dbl2, longerDoubleName;
String firstProgrammer;
```

Assigning / Storing Values

- Single equals sign (=) assigns values
- The value **must** match the type
  - Strongly typed language
- You can change the value as much as you want
  - But it must still be the same type
- Assigning the first time is called
  - Initialization
  - Often done in the same line as declaring
- Objects have the **null** value if not assigned

```java
String firstProgrammer;
firstProgrammer = "Countess Lovelace";
firstProgrammer = "Ada Lovelace"; // new value

int puppyCounter = 100; // initialization
int a = 5, b, c; // allowed, but not clear
double my_value = a; // allowed! makes it 5.0
int _int = 10.5; // won't compile!
```

In zyBooks has a in-class if you want to try it.

Practice 1 – Group Reflection

```
int A = 5;
int B = 2;
int C = 10;
A = B;
B = C;
```

What are the final values for A, B, and C?

```
-----------------------------------------------------------------------------
```

```
int A = 10;
int B = 20;
A = B;
B = A;
```

What are the final values for A and B?

```
-----------------------------------------------------------------------------
```

```
????
```

If we wanted to swap the values A and B, how would we modify the code above? Would we need to use a third variable?

Operators

- Operators are MATH
    - = (assignment)
    - + (add)
    - - (subtract or negative)
    - / (divide)
    - * (multiply)
    - % (modulo) - remainder!
- Numeric Types
    - int - always whole number
    - `int myVal = 1 / 2; // evaluates to 0!`
    - double - has decimals
    - `double doubleVal = 1.0 / 2; // evaluates to 0.5!`

Practice 2 – Group Reflection

```
int A, B, C;
A = 10;
A = A + 1;
B = A/2;
A = 6;
C = B + 1;
C = C + 2;
A = B/2;
```

What are the final values for A, B, and C?

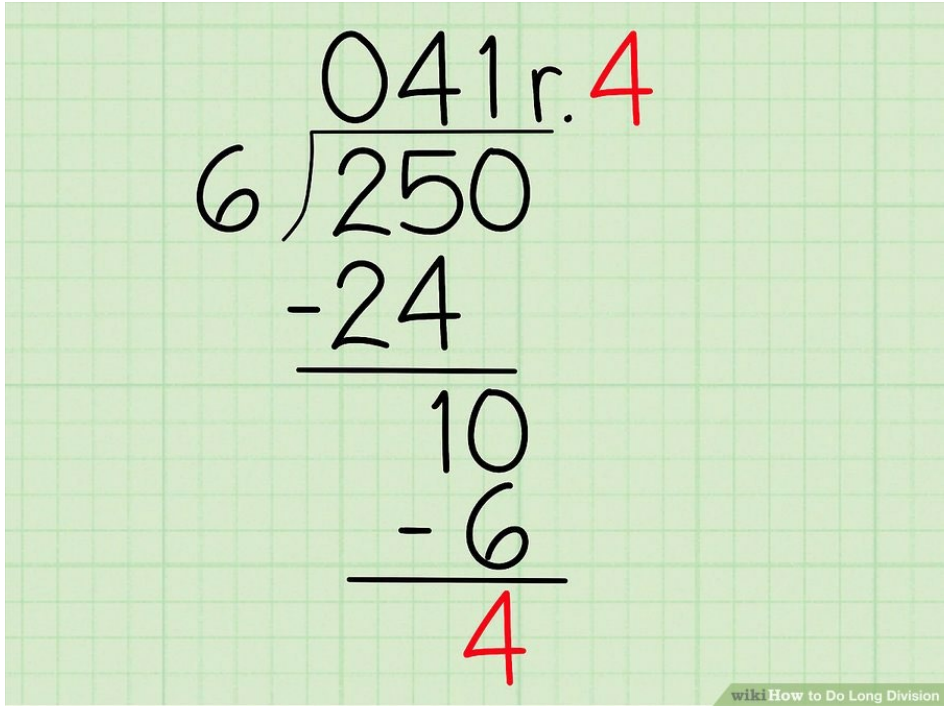
What happens if A is double instead of int?

What happens if A and B are double instead of int?

Integer Division

- `int val = 5 / 6;  // sets val to 0`
- `int val2 = 10 / 3; // sets val2 to 3`
- You lose the decimal point
  - Truncates, does not round!
- This is a very, very common thing
  - both to our advantage
  - and often to our error

Modulo - Extremely useful operation



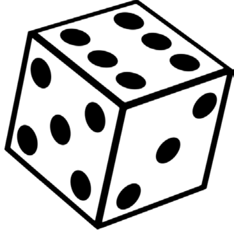
- modulo (%) gives you the **remainder**
- 4th grade math!
- This example:

  `int x = 250 % 6; // would be 4`

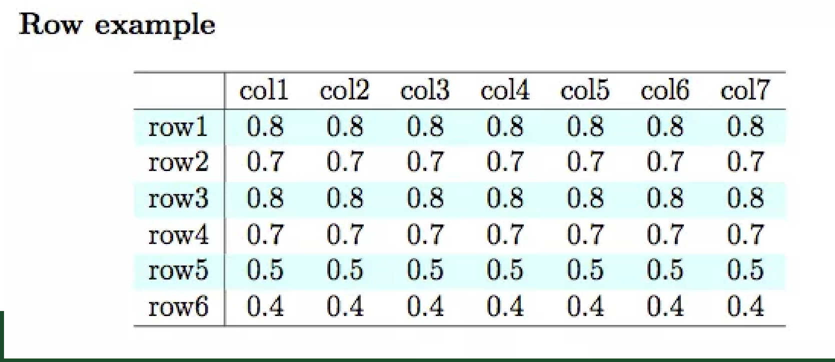
So combining them

```
int whole = 250 / 6; // 41
int remainder = 250 % 6; //
4
```

What are some cases to use it?



- Forming groups
  - The remainder is always between 0 and n-1
  - Value % 6 has a range of 0-5
  - Value % 4 has a range of 0-3
- Think about rolling dice
  - Math.random() % 6; - random number between 0 and 5
- Determining Even and Odd - Math.random() %2 - if 0, even, if 1, odd



Row example

	col1	col2	col3	col4	col5	col6	col7
row1	0.8	0.8	0.8	0.8	0.8	0.8	0.8
row2	0.7	0.7	0.7	0.7	0.7	0.7	0.7
row3	0.8	0.8	0.8	0.8	0.8	0.8	0.8
row4	0.7	0.7	0.7	0.7	0.7	0.7	0.7
row5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
row6	0.4	0.4	0.4	0.4	0.4	0.4	0.4

Pro Tip
Adding a remainder operator, allows us to handle complex math like GCD, and others.

Scanner

- Read something from the console/terminal
- First needs to create an object of the class Scanner
  - `Scanner in = new Scanner(System.in);`
- Second needs to know the type of variable that wants to read – that will define the method you will use
  - `int val = in.nextInt();`
  - `double valD = in.nextDouble();`
  - `String str = in.nextLine();`

Activity : Seat Finder

- Assigned Programmer (only one needed per table)
  - Go to zyBooks
  - Click In Class Activity: Seat Finder
- Everyone else: Help that person code it
  - **Make sure you all explain and know what is going on!**
- Before start programming think about:
  - What is the problem that you need to solve?
  - How you are going to solve it?
  - Write a possible solution in English
  - After that translate your solution to a sequence of instructions in Java

Convenience Operators

- We often find ourselves doing things like
  - `int value = 100;`
  - `value = value + 10;`
- Introducing operator plus assignment
  - value += 10; // same as value = value + 10;
  - +=
  - -=
  - /=
  - *=
  - %=
- We also like to add and subtract by 1
  - --value and ++value
  - value++ and value—
    - Happens after using the value

```
int value = 100;
value++; // value is now 101
value += 10; // value is now 111
value /= 10; // value is now 11
value *= 2; // value is now 22
--value; // value is now 21
value %= 20; // value is now 1
```

Examples

```
int puppyCounter = 100; // so many puppies!


String puppyName = “Spot”;

String puppyLongName = “Cerberus”;


double amountOfFoodPerDayLbs = 20.56;

amountOfFoodPerDayLbs = amountOfFoodPerDayLbs + 10.0; // assigns 30.56 to the variable


boolean isPettable = true;  // only options for boolean is true or false


char singleLetter = ‘c’;  //characters are single letters, notice single quote
```

Advanced Concept:
puppyCounter (and others) follow “camel case” a naming convention that capitalizes every word after the first - very common for java programs.