## Today

- Primitive data types
- Project submission
- ... catch-up with our new students

## **Short Questions?**

# Quiz

## Bits and Bytes

- Bits can have one of two values
- Bytes: collection of 8 bits
  - Stored together and read/written in parallel

## Memory (in the abstract)

- Block of bytes
- Each byte has a unique address
  - Allows the computer to ask for the value stored in a particular byte (or to write to that byte)
- Addresses are contiguous

## Memory and Variables

A variable is a container for a value of a particular type

- Often referred to by a name in our program
- Composed of some number of bytes that are contiguous in memory
- The number of bytes is determined by the type of the variable
  - int: 32 bits (4 bytes)
  - double: 64 bits
  - bool: ?
- For primitive types, the number of bytes necessary to store a value is fixed (long before you start writing your program)
- The JVM and compiler handle all of these low level details for us

## Mathematical Operators

- Satisfy standard precedence relationships
  - Level 3: ( ) for grouping of expressions
  - Level 4: \* / %
  - Level 5: + -
- Each operator is potentially defined differently for different data types

#### int vs double

- int: precisely represent integers within a range
  - Need to ensure that our mathematical operations will stay within this range
- double:
  - Precisely represent 0 and 1
  - Many other integers (and values in between) are only approximated

Which one you choose depends on the values that you need to represent

# Phone Contract Example

Should prices be ints or doubles?

## **Example: Printing Ints**

```
System.out.println(5);
```

## "=" Operator

The "=" operator is a storage operation, not a statement of equality

```
foo = 5+3;
```

- Left hand side must be a variable
- Right hand side is an expression that results in the value to be stored

```
int foo;
foo = 5;
System.out.println(foo);
```

```
int foo;
foo = 5;
foo = 3;
System.out.println(foo);
```

```
int foo;
foo = 5;
foo = foo + 3;
System.out.println(foo);
```

```
int foo;
foo = 5;
foo = foo + 3;
System.out.println(foo);
```

"=" is about storage, not equality!

#### Some Syntactic Notes

Curly brackets {} and parentheses () always come in matching pairs

- {}: used to group several statements together
- (): used for method (or function) definition/calls
- Eclipse helps you to keep track of these pairs by:
  - Indenting code within {}
  - Giving errors when one of a pair is missing

Semicolons (;) are necessary to end a single code statement.

Eclipse will also give you an error if you have forgotten one

#### **Camel Case Convention**

- We try to make our identifiers as descriptive as possible by describing them with multiple words
- However, a space character cannot be used as part of an identifier
- So, we cram the words together:

```
int numberOfCamels;
```

- Note:
  - First letter of a variable name is always lower case
  - But the first letter of a class name is always upper case

# Juggling Exercise

## Handing In a Project

#### Process:

- Write, test and debug the code
- Export project to a Zip file
- Submit to D2L dropbox

## **Exporting a Project**

- Select the project in the Package Explorer
- File: Export
- Export destination: General: Double click on "Archive File"
- To archive file: Give the name of the zip file
- Leave "Save in zip format" selected
- Click Finish

## Wrap Up

#### Being released:

- HW 1: Turing's Craft
- Project 0: Eclipse + D2L
- Videos for next week

#### Next time:

 Assignment statements, manipulating variables, characters, mixing types