Chapter 3
Object Interaction

3.7 to 3.8.3

Primitive vs Object Types
Logical Operators
String Concatenation
Modulo Operator

Primitive vs Object Data Types

Primitive Data Type

- int, boolean, float, etc.
- predefined in Java
- simple, only a single value
- stored directly in a variable

Object Data Type

- String, Student, or any other class
- defined in libraries or by programmer
- complex, values and methods possible
- stored by *reference*

Direct Storage vs Storing by Reference

When a primitive data type is declared, the compiler will request space in RAM that is the exact size of that data type. The value of the variable is stored at that location.

When an object data type is declared, the compiler requests space in RAM to store the *address* of the variable, which will actually exist somewhere else in RAM.

More on this later...

Logical Operators

A *boolean expression* resolves to *true* or *false*. Most simple expressions are a comparison.

e.g., if (alpha >= beta) ...

To form more complicated boolean expressions, we combine simple expressions using the boolean operators.

e.g., if (alpha >= beta && beta <= gamma) ...

Logical AND – &&

Logical AND checks that all simple expressions are *true*, which gives a *true* result. Otherwise, the result in *false* (i.e., if any are false, the result is false).

```
3 < 4 && 5 < 6
```
is true

```
3 < 4 && 5 > 6
```
is false

```
3 == 4 && 5 == 6
```
is false

```
3 < 4 && 5 < 6 && 7 == 8
```
is false

Logical OR – ||

Logical OR checks that any simple expressions are *true*, which gives a *true* result. A *false* result only occurs if all are *false*.

```
3 < 4 || 5 < 6
```
is true

```
3 < 4 || 5 == 6
```
is true

```
3 == 4 || 5 == 6
```
is false

```
3 < 4 || 5 == 6 || 7 == 8
```
is true

String Concatenation

+ means “addition” for numbers

+ means “join” or “concatenate” for strings

You can use the empty string “” to force numbers to be treated as strings.

```
return 3 + 4;        // returns the integer 7
return “” + 3 + 4    // returns the string “34”
```

Modulo Operator

The modulo operator (%) provides the *remainder* from an *integer division*.

Consider 27 / 4

- 4 goes into 27 six (6) times
- 4 x 6 is 24
- there is a remainder of 27 – 24 = 3

```
int remainder = 27 % 4;
```

The variable remainder will contain the value 3.

Modulo Operator

The modulo operator allows us to conveniently perform useful mathematical operations in computing.

e.g., to determine if an integer is even or odd,

value % 2

will have zero remainder for even.

With our number display, we use modulo the "roll over" at the limit, back to zero (e.g., 59 minutes rolls over to 00, rather than 60).

Assigned Work

Read Chapter 3:
3.7 to 3.8.3

Complete exercises 3.5 to 3.21

Record your answers in a text document or OpenOffice document