# CSE 1710

Lecture 4

The assigned reading was:

- **Relational Operators**
  - the text of sec 3.2.4, pp. 110-111, including figure 3.10 on p. 112, but not PT 3.3 nor IMD 3.1
- **Selection**
  - sec 5.1.1, pp. 173
- **The `if` Statement**
  - sec 5.1.2 pp. 173-177
- **The Relational Expression topic**
  - a component of sec 5.1.3, pp.177-178, but not PT 5.3 or the remainder of section 5.1.3

**Tasks you should be able to perform:**

- Given an expression, determine its **outcome** (not only the value but the value's type)
- Characterize what is meant by **closure**
- Given a **symbol**, state which operators it represents
- Given an **operator**, describe its behaviour (closure, undefined operations)

Expressions (so far)

**1.** arithmetic operators and single data type

**2.** arithmetic operators and mixed data types (auto promotion and operation ambiguity)

**3.** arithmetic operators, mixed data types, and manual cast operators

**4.** arithmetic, relational, and boolean operators, mixed data types, and manual cast operators

Determine the outcome (data type and value)

## 1. single data type, arithmetic operators:

```
5 / 2
5.0 * 2.0
8f % 3F
8L / 3l
```

and so on for all of the 11 arithmetic operators…

5 basic: `+ - * / %` (in infix form)

6 other: `++ --` (each in a prefix and postfix version)

`- +` (in prefix form)

## what are these ++ and -- operators anyways?

*these are 4 of the 11 arithmetic operators…*

`3345L++` and `++3345L` are identical

they evaluate to `3346L`

The difference is revealed in the context of an assignment statement…

## so when would the ++ and -- operators be useful?

```
int x1 = 10;
int x2 = ++x1;
if (x2 > 10) {
       System.out.println("The value of x1 is: " + x1);
       System.out.println("The value of x2 is: " + x2);
}

int y1 = 10;
int y2 = y1++;
if (y2 > 10) {
       System.out.println("The value of y1 is: " + y1);
       System.out.println("The value of y2 is: " + y2);
}
```

## suppose we want to print to a file instead of the console – this would require changes to 4 statements

int x1 = 10;<br>
int x2 = ++x1;<br>
if (x2 > 10) {<br>
       ~~System.out.~~println("The value of x1 is: " + x1);<br>
       ~~System.out.~~println("The value of x2 is: " + x2);<br>
}

int y1 = 10;<br>
int y2 = y1++;<br>
if (y2 > 10) {<br>
       ~~System.out.~~println("The value of y1 is: " + y1);<br>
       ~~System.out.~~println("The value of y2 is: " + y2);<br>
}

```
PrintStream output = System.out;
// here you could reassign the variable output to
// instead refer to a file
// For example:
// output = ...
int x1 = 10;
int x2 = ++x1;
if (x2 > 10) {
        output.println("The value of x1 is: " + x1);
        output.println("The value of x2 is: " + x2);
}

int y1 = 10;
int y2 = y1++;
if (y2 > 10) {
        output.println("The value of y1 is: " + y1);
        output.println("The value of y2 is: " + y2);
}
```

Determine the outcome (data type and value)

**2. mixed data types, arithmetic operators**:

```
5 / 2 + 2.5
5.0 / 2
128L + 23f
```

Determine the outcome (data type and value)

**3.** arithmetic operators, mixed data types, and manual cast operators

```
(double) 5 / 2 + 2.5
5.0 / 2
(double) 128L + 23f
```

**Key points to remember:**

Precedence Levels:

| | |
|---|---|
| manual cast<br>(e.g., `(double)` ) | -3 (p. 32)<br>right-to-left assoc. |
| `+ - * / %` infix operators | -5 up to -4 |
| `+ - ++ --` pre/postfix operators | -2 up to -1 |

cast will be performed before `+ - * / %`

## What has which type of property?

| entity | property |
|---|---|
| data type | has a representation scheme |
| operator | might or might not have closure |
| data type | might or might not have a set of operators |

For ________________, what is its representation scheme?
(insert data type here; we have covered 8 different ones)

Does ________________ have closure?
(insert operator name here; we have covered 11+7+3=21 different ones!)

Does ________________ have any operators?
(insert data type here)

**Can you answer these questions for EVERY one of the possible completions?**

## Sometimes we want division and not quotient

E.g., 17 birds hit the window per hour, and we want an expression that gives the value of *bird hits per second…*

```
int secondsPerHour = 60*60;
int hitsPerHour = 17;
double hitsPerSecond = ????;
```

**which is correct for the RHS??**

```
hits / secondsPerHour
(double) hits / secondPerHour
```

## Relational Operators

pp. 110-112, Fig 3.10

- they are (not an official categorization):
  - basic: `> < <= >= == !=`
  - advanced: `instanceof`
- their precedence is **lower** than the arithmetic operators
  - relational operators have precedence levels ranging between -8 up to -7
- in general, they DO NOT have closure
  - numeric operands produce a boolean outcome
  - can you find the exception to this statement?

What is a Boolean Expression?
(and why do we need them?)

**an expression that evaluates to a boolean value**

**we need them to perform selection**

## why do we need them?

```
if (true) {
    // these statements are executed
} else {
    // these statements are executed
}
```

## this is the same as…

```
boolean isConditionMet = true;
if (isConditionMet) {
    // these statements are executed
} else {
    // these statements are executed
}
```

## and now the RHS could be replaced with **ANY** boolean expression

```
boolean isConditionMet = true;
if (isConditionMet) {
    // these statements are executed
} else {
    // these statements are executed
}
```