CMPS 12A 4-8-10

Ex. charCode.java

why: x = (char) sc.nextInt();

not: x = sc.nextInt();

(char) is a Caste Operator

In general

(data_type) expression

converts expression to type data_type.

Java Arithmetic: +, -, *, %

four data types:

type	literal expressions
double	6.0
float	6.0f or 6.0F
long	6l or 6L
int	6 or 0110 (binary) or 0x6 (hex)

To evaluate: a + b when a or b are char, short, or byte, both are converted to int.

Mixed expressions

Ex.

```
int a = 6;
double b = 6.0;
```

a + b

↯

1st convert to double.

Ex.

```
int x = 6;
long y;
double z;

y = x;  // widening
z = y;  // conversions
System.out.println(z);
// 6.0 is Printed
```

Ex. double z = 6.5;
int x;
x = z; // narrowing conversion
// syntax error.

Ex. double z = 6.5;
int x;
x = (int) z; // works

must use explicit cast to narrow.

System.out.println(x);
// prints 6

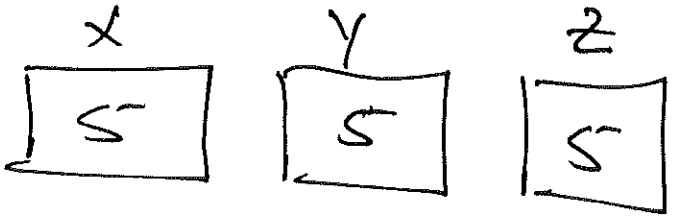
Round instead of truncate

Ex. double z = 6.6;

```
int x, y;
x = (int) z;
y = (int) Math.round(z);
System.out.println(x); // 6
System.out.println(y); // 7
```

Note: assignment x = y is itself an expression with value the value assigned.

Ex. `int x, y, z = 5;`




```
x = (y = z);
```

Can also do

```
x = y = z;
```

Compound Assignments:

`a += b`	same as	`a = a + b`
`a -= b`	...	`a = a - b`
`a *= b`	...	`a = a * b`
`a /= b`	...	`a = a / b`
`a %= b`	...	`a = a % b`

note difference between floating pt. Division and integer division.

21.0 / 10.0 eval. to 2.1

21 / 10 " " 2

21 % 10 " " 1

Ex.

```
double time = 3725.7;
int h, m, s;
s = (int) Math.round(time);
m = s/60; //m=62
s = s%60; //s=6,   could do s %= 60
h = m/60; // h=1
m = m%60; // m=2
```

Auto increment & Auto decrement OPS.

Long WAY	Compound Assgn	Auto inc/dec
`i = i + 1`	`i += 1`	`i++` or `++i`
`i = i - 1`	`i -= 1`	`i--` or `--i`
		↑ Postfix ↑ Prefix

- `i++` : 1st eval to old val of i, then increment i
- `++i` : 1st increment i, then evaluate to new val of i.

Ex.

```
int i = 6, a, b, c, d;

a = i++;
b = ++i;
c = i--;
d = --i;
```

Trace:

i	a	b	c	d
6	-	-	-	-
7	6	-	-	-
8	6	8	-	-
7	6	8	8	-
6	6	8	8	6

Operator Precedence:

$$a \ op_1 \ b \ op_2 \ c$$

If op_1 is higher precedence (or equal precedence) this is

$$(a \ op_1 \ b) \ op_2 \ c$$

If op_2 is higher, we get

$$a \ op_1 \ (b \ op_2 \ c)$$

Ex. 7 + 5 * 3 eval. to 22

Ex. 100 / 5 * 2 eval to 40

2 * 100 / 5 ---- 40

Conditional evaluation

most ops are binary:

a op b — 2 operands

EX.

Compare for equality: ==

exp1 == exp2

evaluates to true iff exp1 and exp2 have same value, otherwise eval. to false.

6.0 == 6.1 eval to false

6.0 == 6 eval to true (I think)

Conditional evaluation :

cond ? exp1 : exp2 } ternary op i.e. 3 operands.

↑

boolean exp
i.e true
or false