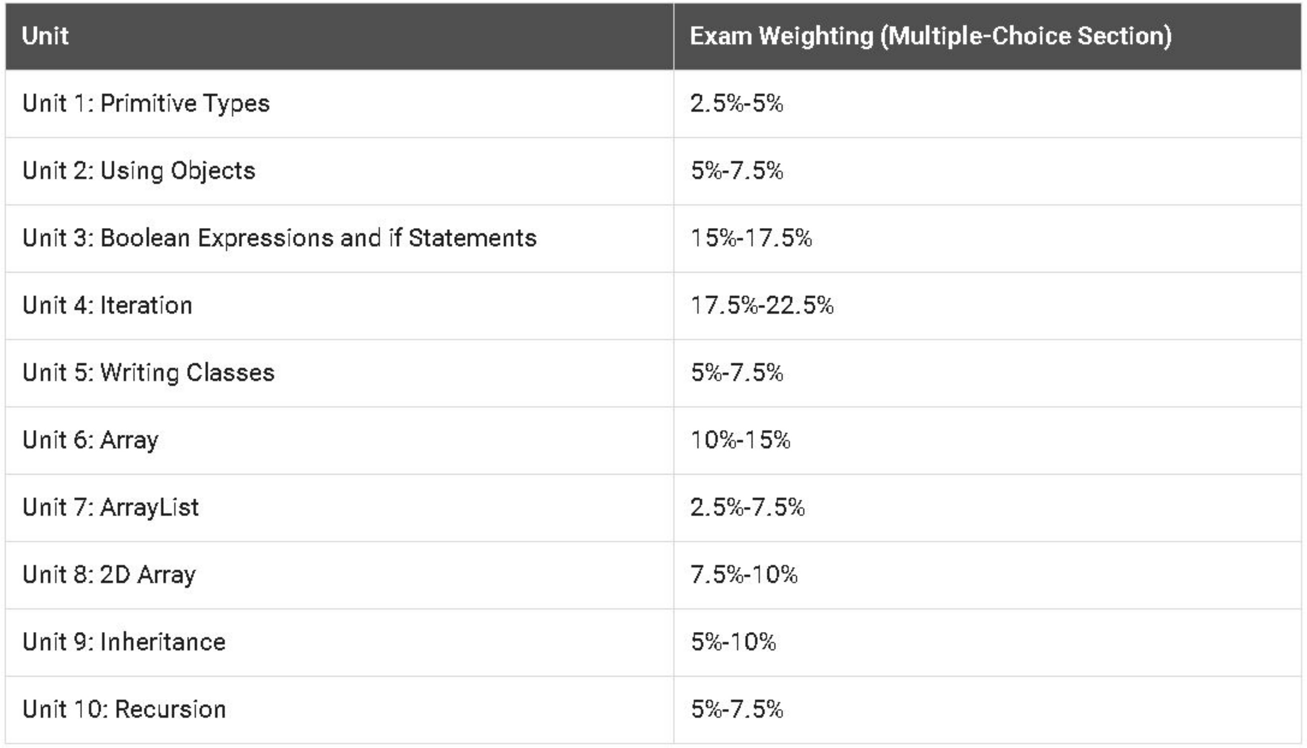
AP CSA

Coding with java

Material



Unit	Exam Weighting (Multiple-Choice Section)
Unit 1: Primitive Types	2.5%-5%
Unit 2: Using Objects	5%-7.5%
Unit 3: Boolean Expressions and if Statements	15%-17.5%
Unit 4: Iteration	17.5%-22.5%
Unit 5: Writing Classes	5%-7.5%
Unit 6: Array	10%-15%
Unit 7: ArrayList	2.5%-7.5%
Unit 8: 2D Array	7.5%-10%
Unit 9: Inheritance	5%-10%
Unit 10: Recursion	5%-7.5%

I plan to cover all material in 12 lectures and there are extra problem solving lectures that designed to go through code and homework questions

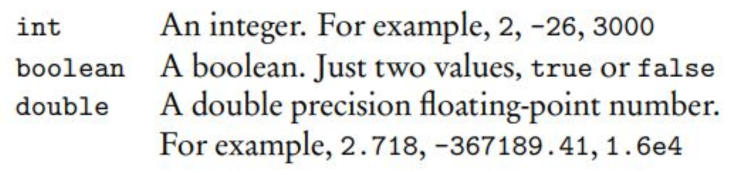
How to succeed in this class

1. Create folder and organize your files
2. Take note, either directly on this pdf via your ipad or on your notebook
3. Review frequently
4. Write code everyday
5. Make sure you can write the code you have written

Type and Identifier

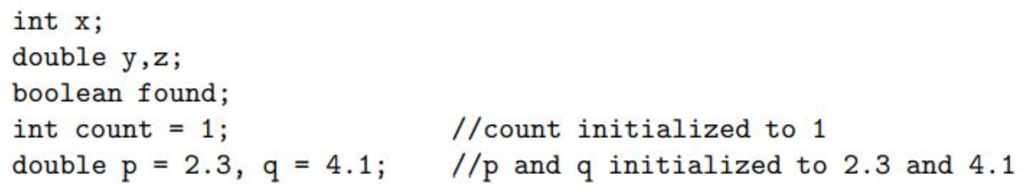
Identifier cannot start with digit

Primitive type:



int	An integer. For example, 2, -26, 3000
boolean	A boolean. Just two values, true or false
double	A double precision floating-point number. For example, 2.718, -367189.41, 1.6e4

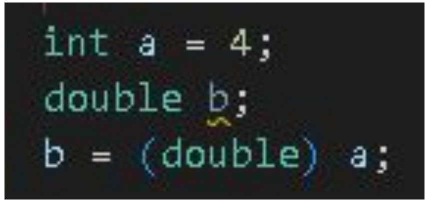
Declare a variable before you use it.



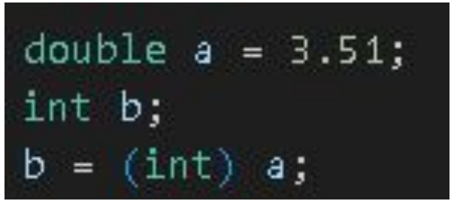
```
int x;
double y,z;
boolean found;
int count = 1;              //count initialized to 1
double p = 2.3, q = 4.1;    //p and q initialized to 2.3 and 4.1
```

Casting

1. Explicit casting:



```
int a = 4;
double b;
b = (double) a;
```



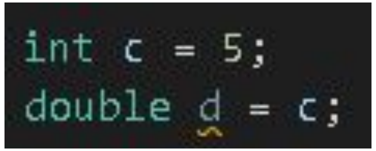
```
double a = 3.51;
int b;
b = (int) a;
```

When you convert a double to int with explicit casting, the double number simply get truncated.

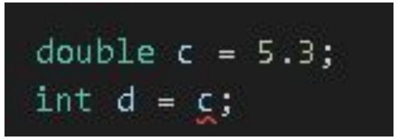
Here b is 4.0

Here b is 3

2. Implicit casting



```
int c = 5;
double d = c;
```

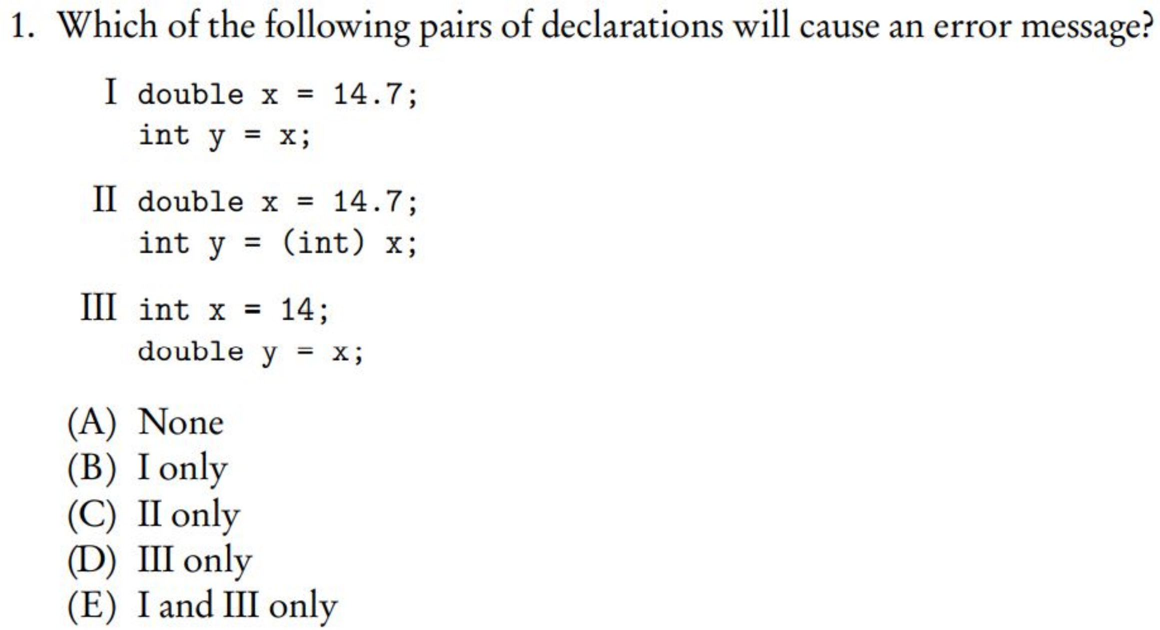


```
double c = 5.3;
int d = c;
```

Here d is 5.0

This will result in compile error

Pop quiz 1



1. Which of the following pairs of declarations will cause an error message?

I
```
double x = 14.7;
int y = x;
```

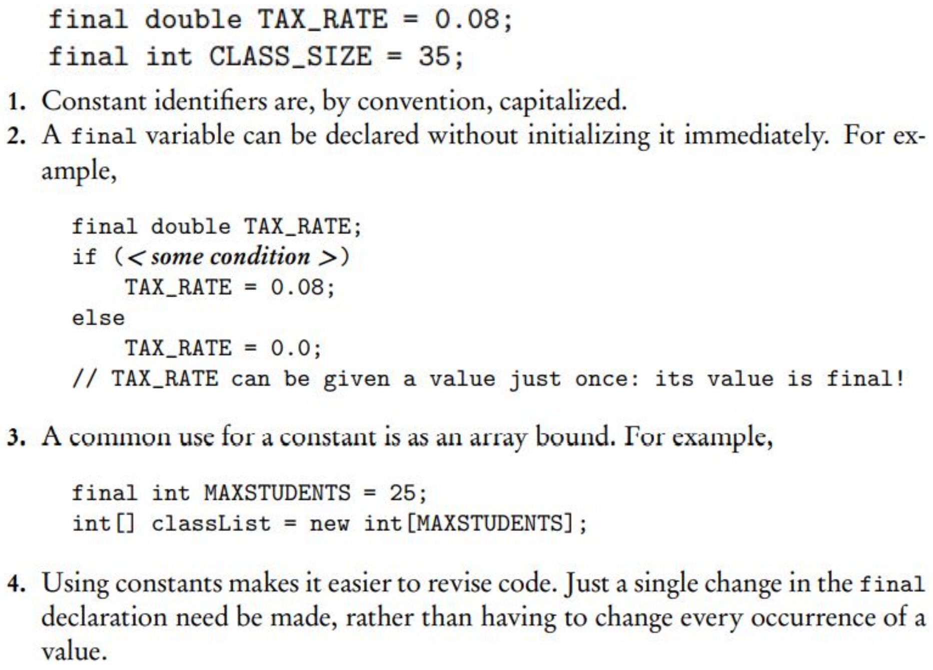
II
```
double x = 14.7;
int y = (int) x;
```

III
```
int x = 14;
double y = x;
```

(A) None
(B) I only
(C) II only
(D) III only
(E) I and III only

Final variable

A final variable or user-defined constant, identified by the keyword final, is used to name a quantity whose value will not change.



```
final double TAX_RATE = 0.08;
final int CLASS_SIZE = 35;
```

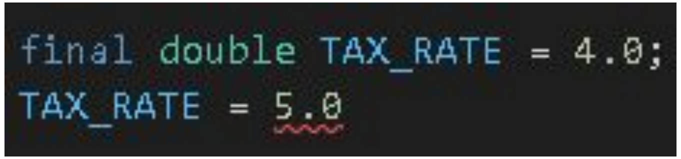
1. Constant identifiers are, by convention, capitalized.
2. A `final` variable can be declared without initializing it immediately. For example,

```
final double TAX_RATE;
if (< some condition >)
    TAX_RATE = 0.08;
else
    TAX_RATE = 0.0;
// TAX_RATE can be given a value just once: its value is final!
```

3. A common use for a constant is as an array bound. For example,

```
final int MAXSTUDENTS = 25;
int[] classList = new int[MAXSTUDENTS];
```

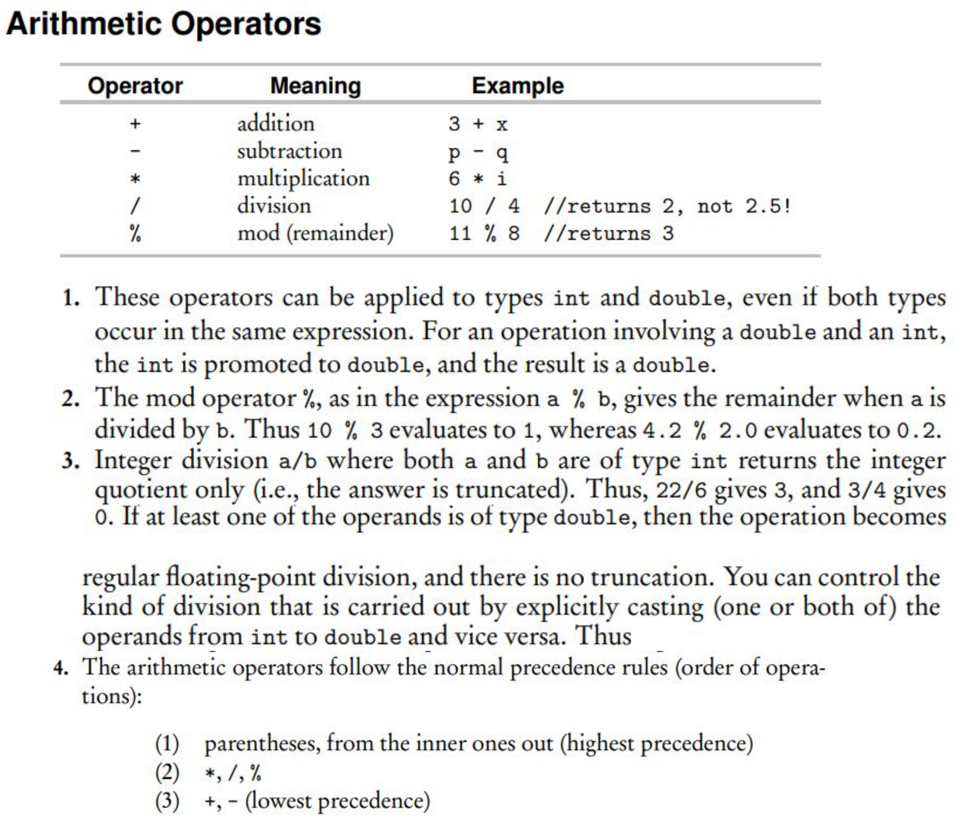
4. Using constants makes it easier to revise code. Just a single change in the `final` declaration need be made, rather than having to change every occurrence of a value.



```
final double TAX_RATE = 4.0;
TAX_RATE = 5.0
```

This will produce compile error

Operator



Arithmetic Operators

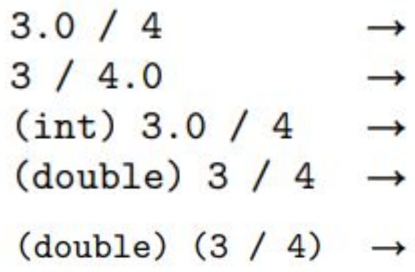
Operator	Meaning	Example
`+`	addition	`3 + x`
`-`	subtraction	`p - q`
`*`	multiplication	`6 * i`
`/`	division	`10 / 4  //returns 2, not 2.5!`
`%`	mod (remainder)	`11 % 8  //returns 3`

1. These operators can be applied to types `int` and `double`, even if both types occur in the same expression. For an operation involving a `double` and an `int`, the `int` is promoted to `double`, and the result is a `double`.
2. The mod operator `%`, as in the expression `a % b`, gives the remainder when `a` is divided by `b`. Thus `10 % 3` evaluates to `1`, whereas `4.2 % 2.0` evaluates to `0.2`.
3. Integer division `a/b` where both `a` and `b` are of type `int` returns the integer quotient only (i.e., the answer is truncated). Thus, `22/6` gives `3`, and `3/4` gives `0`. If at least one of the operands is of type `double`, then the operation becomes

   regular floating-point division, and there is no truncation. You can control the kind of division that is carried out by explicitly casting (one or both of) the operands from `int` to `double` and vice versa. Thus
4. The arithmetic operators follow the normal precedence rules (order of operations):

   (1) parentheses, from the inner ones out (highest precedence)
   (2) `*`, `/`, `%`
   (3) `+`, `-` (lowest precedence)

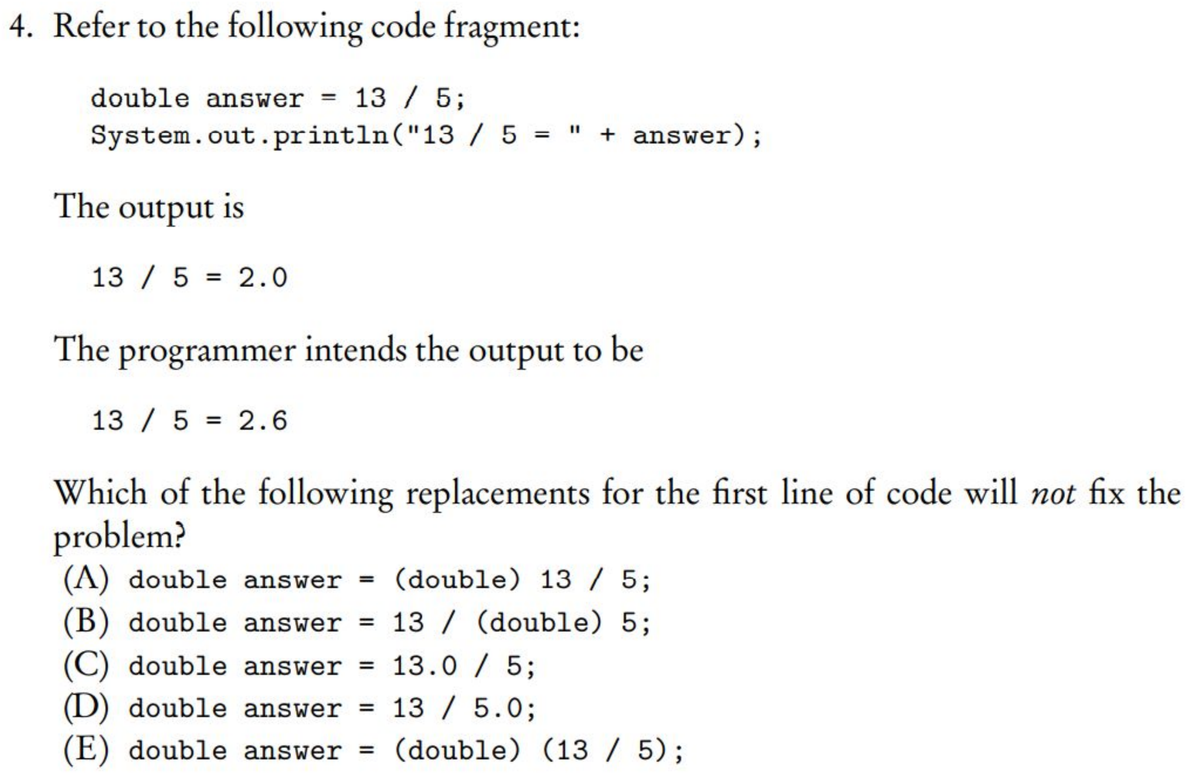
Pop quiz 2:



```
3.0 / 4              →
3 / 4.0              →
(int) 3.0 / 4        →
(double) 3 / 4       →

(double) (3 / 4)     →
```

Pop Quiz 3



4. Refer to the following code fragment:

```
double answer = 13 / 5;
System.out.println("13 / 5 = " + answer);
```

The output is

```
13 / 5 = 2.0
```

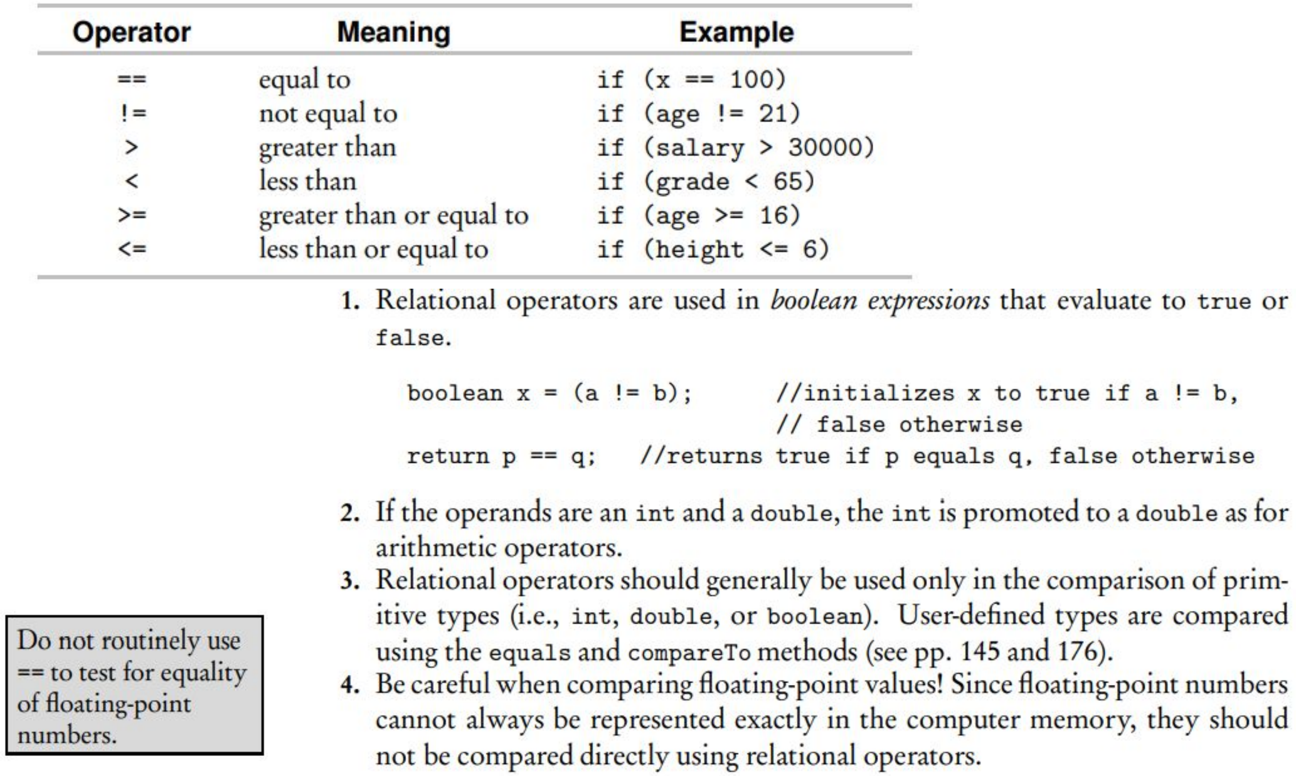
The programmer intends the output to be

```
13 / 5 = 2.6
```

Which of the following replacements for the first line of code will *not* fix the problem?

(A) `double answer = (double) 13 / 5;`
(B) `double answer = 13 / (double) 5;`
(C) `double answer = 13.0 / 5;`
(D) `double answer = 13 / 5.0;`
(E) `double answer = (double) (13 / 5);`

Rational operator



Operator	Meaning	Example
`==`	equal to	`if (x == 100)`
`!=`	not equal to	`if (age != 21)`
`>`	greater than	`if (salary > 30000)`
`<`	less than	`if (grade < 65)`
`>=`	greater than or equal to	`if (age >= 16)`
`<=`	less than or equal to	`if (height <= 6)`

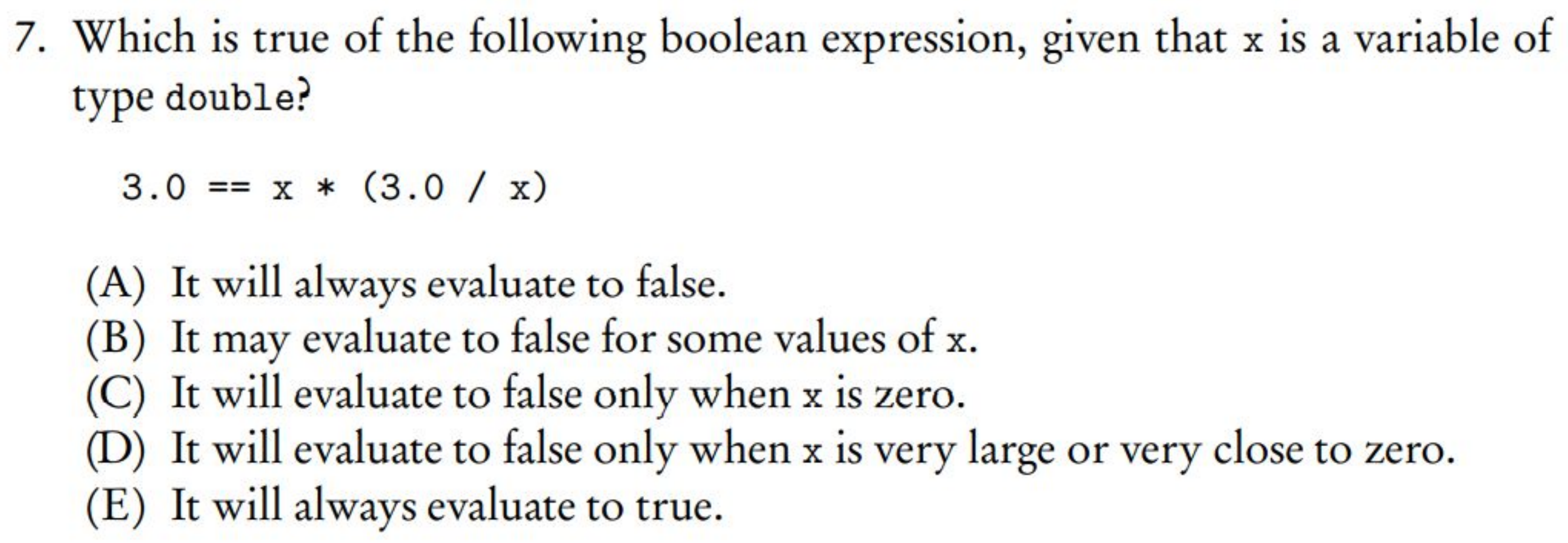
1. Relational operators are used in *boolean expressions* that evaluate to `true` or `false`.

```
boolean x = (a != b);      //initializes x to true if a != b,
                           // false otherwise
return p == q;   //returns true if p equals q, false otherwise
```

2. If the operands are an `int` and a `double`, the `int` is promoted to a `double` as for arithmetic operators.
3. Relational operators should generally be used only in the comparison of primitive types (i.e., `int`, `double`, or `boolean`). User-defined types are compared using the `equals` and `compareTo` methods (see pp. 145 and 176).
4. Be careful when comparing floating-point values! Since floating-point numbers cannot always be represented exactly in the computer memory, they should not be compared directly using relational operators.

Do not routinely use == to test for equality of floating-point numbers.

Pop quiz 4

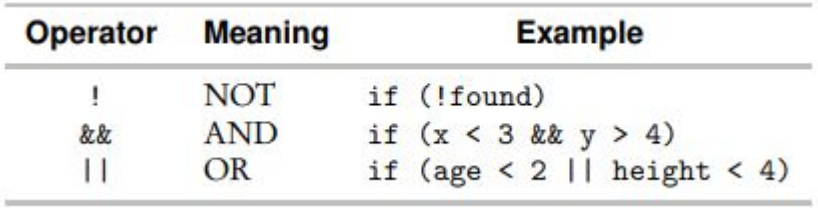


7. Which is true of the following boolean expression, given that `x` is a variable of type `double`?

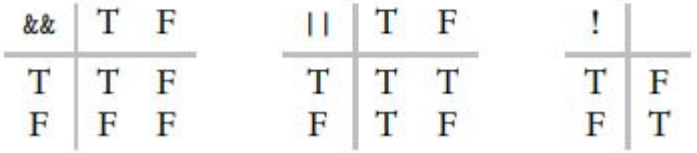
```
3.0 == x * (3.0 / x)
```

(A) It will always evaluate to false.
(B) It may evaluate to false for some values of `x`.
(C) It will evaluate to false only when `x` is zero.
(D) It will evaluate to false only when `x` is very large or very close to zero.
(E) It will always evaluate to true.

Logit operator



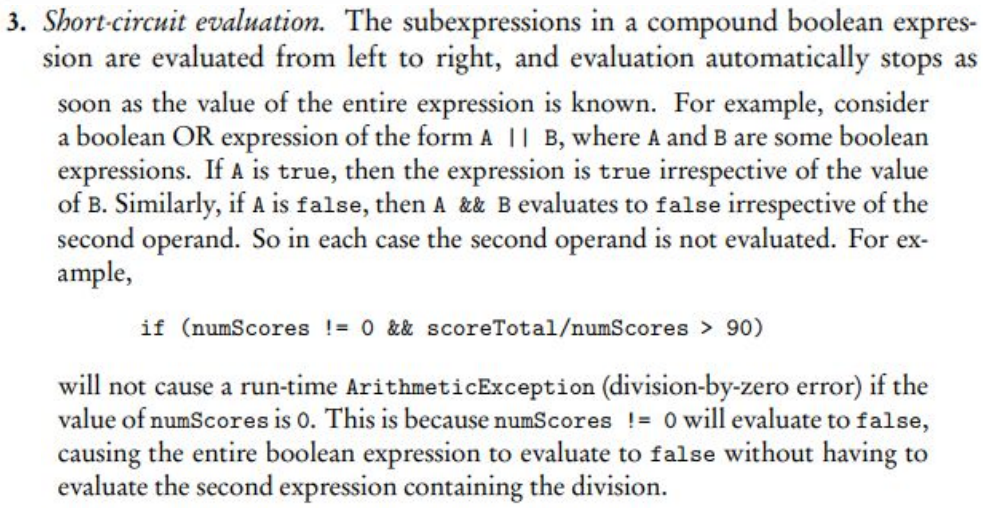
Operator	Meaning	Example
`!`	NOT	`if (!found)`
`&&`	AND	`if (x < 3 && y > 4)`
`||`	OR	`if (age < 2 || height < 4)`



`&&`	T	F
T	T	F
F	F	F

`||`	T	F
T	T	T
F	T	F

`!`	
T	F
F	T

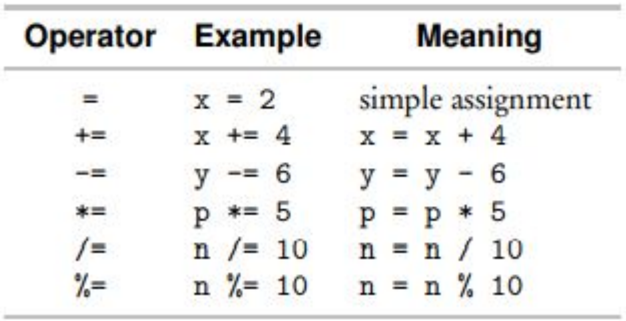


3. *Short-circuit evaluation.* The subexpressions in a compound boolean expression are evaluated from left to right, and evaluation automatically stops as soon as the value of the entire expression is known. For example, consider a boolean OR expression of the form `A || B`, where `A` and `B` are some boolean expressions. If `A` is `true`, then the expression is `true` irrespective of the value of `B`. Similarly, if `A` is `false`, then `A && B` evaluates to `false` irrespective of the second operand. So in each case the second operand is not evaluated. For example,

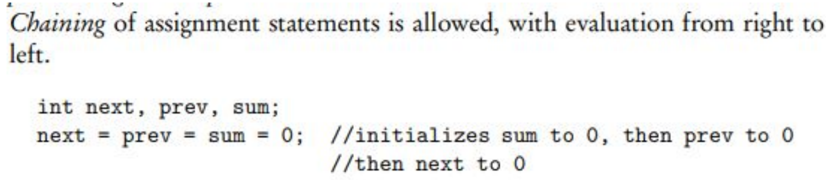
```
    if (numScores != 0 && scoreTotal/numScores > 90)
```

will not cause a run-time `ArithmeticException` (division-by-zero error) if the value of `numScores` is `0`. This is because `numScores != 0` will evaluate to `false`, causing the entire boolean expression to evaluate to `false` without having to evaluate the second expression containing the division.

Assignment operator

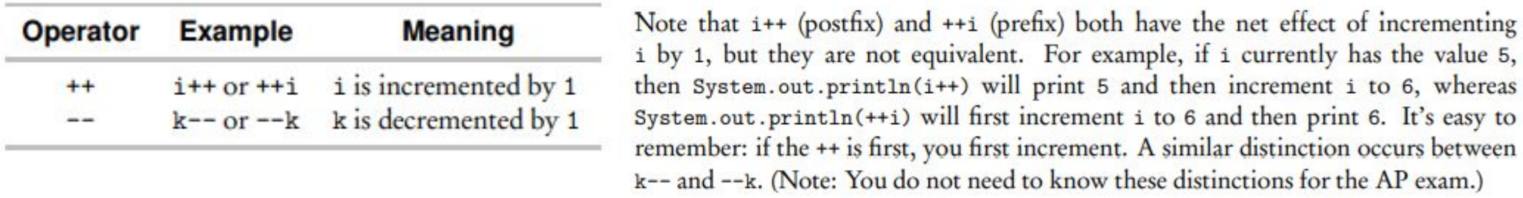


Operator	Example	Meaning
`=`	`x = 2`	simple assignment
`+=`	`x += 4`	`x = x + 4`
`-=`	`y -= 6`	`y = y - 6`
`*=`	`p *= 5`	`p = p * 5`
`/=`	`n /= 10`	`n = n / 10`
`%=`	`n %= 10`	`n = n % 10`



Chaining of assignment statements is allowed, with evaluation from right to left.

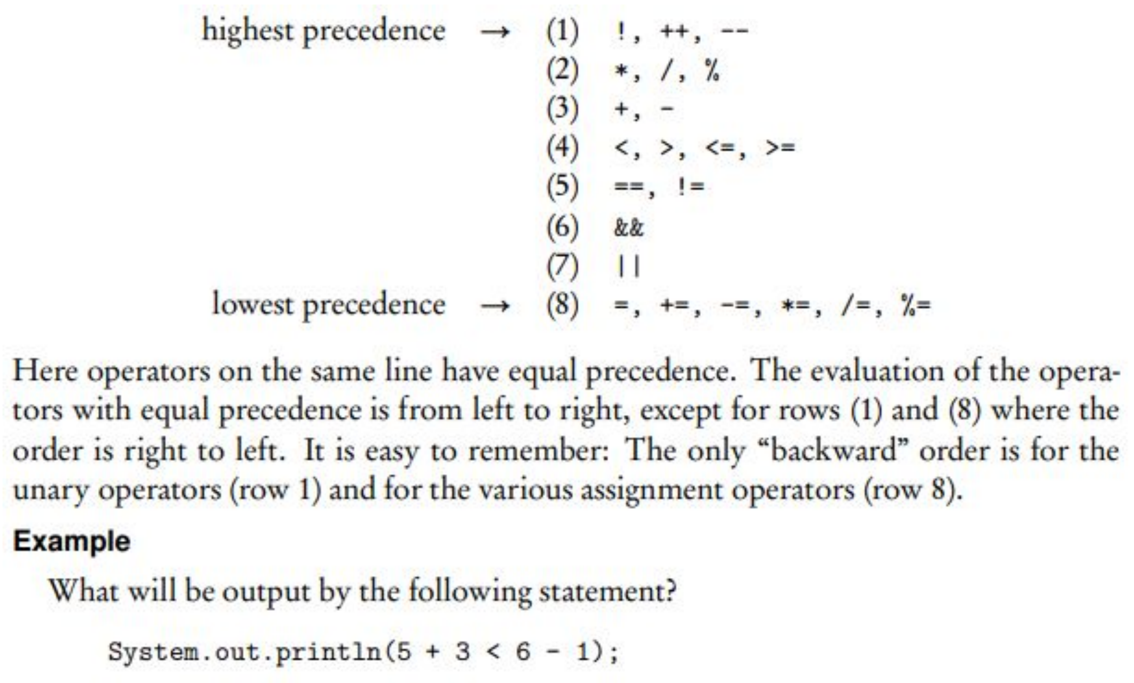
```
int next, prev, sum;
next = prev = sum = 0;  //initializes sum to 0, then prev to 0
                        //then next to 0
```



Operator	Example	Meaning
`++`	`i++` or `++i`	`i` is incremented by 1
`--`	`k--` or `--k`	`k` is decremented by 1

Note that `i++` (postfix) and `++i` (prefix) both have the net effect of incrementing `i` by `1`, but they are not equivalent. For example, if `i` currently has the value `5`, then `System.out.println(i++)` will print `5` and then increment `i` to `6`, whereas `System.out.println(++i)` will first increment `i` to `6` and then print `6`. It's easy to remember: if the `++` is first, you first increment. A similar distinction occurs between `k--` and `--k`. (Note: You do not need to know these distinctions for the AP exam.)

Operator precedence



highest precedence	→	(1)	`!, ++, --`
		(2)	`*, /, %`
		(3)	`+, -`
		(4)	`<, >, <=, >=`
		(5)	`==, !=`
		(6)	`&&`
		(7)	`||`
lowest precedence	→	(8)	`=, +=, -=, *=, /=, %=`

Here operators on the same line have equal precedence. The evaluation of the operators with equal precedence is from left to right, except for rows (1) and (8) where the order is right to left. It is easy to remember: The only "backward" order is for the unary operators (row 1) and for the various assignment operators (row 8).

Example

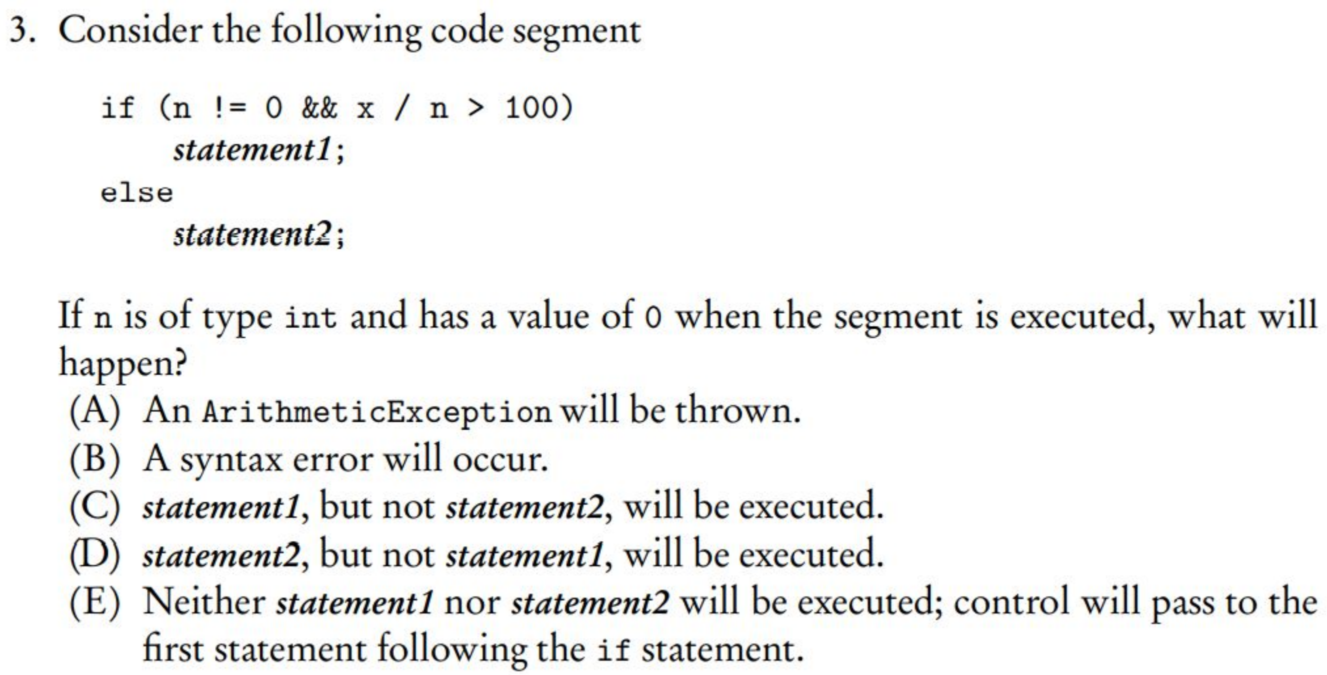
What will be output by the following statement?

```
System.out.println(5 + 3 < 6 - 1);
```

Pop quiz 8

1. 2 > 3 == 5 < 6
2. 2 > 3 == 5 > 6
3. 0 == 0.0
4. 8 >= 5 || 5 < 3 && 7 == 7
5. 8 <= 5 || 5 < 3 && 7 == 7

Pop quiz 5



3. Consider the following code segment

```
if (n != 0 && x / n > 100)
    statement1;
else
    statement2;
```

If `n` is of type `int` and has a value of `0` when the segment is executed, what will happen?

(A) An `ArithmeticException` will be thrown.
(B) A syntax error will occur.
(C) *statement1*, but not *statement2*, will be executed.
(D) *statement2*, but not *statement1*, will be executed.
(E) Neither *statement1* nor *statement2* will be executed; control will pass to the first statement following the `if` statement.

Pop quiz 6

5. What value is stored in `result` if

```
int result = 13 - 3 * 6 / 4 % 3;
```

(A) −5
(B) 0
(C) 13
(D) −1
(E) 12

Pop quiz 7

11. Which of the following will evaluate to true only if boolean expressions `A`, `B`, and `C` are all false?

(A) `!A && !(B && !C)`
(B) `!A || !B || !C`
(C) `!(A || B || C)`
(D) `!(A && B && C)`
(E) `!A || !(B || !C)`

12. Assume that `a` and `b` are integers. The boolean expression

```
!(a <= b) && (a * b > 0)
```

will always evaluate to `true` given that

(A) `a = b`
(B) `a > b`
(C) `a < b`
(D) `a > b` and `b > 0`
(E) `a > b` and `b < 0`