Introduction to Java for C++ Programmers

Segment - 3

JAC 444

By: Mahboob Ali

Objectives

In this segment you will be learning about:

- Numeric Operators in Java
- Type Conversion
- If, switch, For, While, Do-While Statements
- Labeled Break and Labeled Continue

Numeric Operators in Java

Operator	Description	Example
++	Increment by 1(or 1.0)	k++
--	Decrement by 1(or 1.0)	k--
+	Unary plus	+value
-	Unary minus	-value
*	Multiplication	x * y
/	Division	x / y
%	Modulo	x % y
+	Addition	x + y
-	Subtraction	x – y
<	Less than	x < y
>	Greater than	x > y
<=	Less than/equal	x <= y
>=	Greater than/equal	x >= y
==	Equals (identical values)	x == y
!=	Is not equal to	x != y
op=	op assignment(+ =, - =, *=, etc)	x += y

if, if-else, if else –if else

1.
```
if (boolean-expression) {
        statements;
    }
```

2.
```
if (boolean-expression) {
        statements;
     } else {
        statements;
    }
```

3.
```
if (boolean-expression) {
        statements;
    } else if (boolean-expresion ) {
        statements;
    } else {
        statements;
    }
```

Trace if-else statement

Suppose score is 70.0

The condition is false

```
if (score >= 90.0)
  grade = 'A';
else if (score >= 80.0)
  grade = 'B';
else if (score >= 70.0)
  grade = 'C';
else if (score >= 60.0)
  grade = 'D';
else
  grade = 'F';
```

Trace if-else statement

Suppose score is 70.0

The condition is false

```
if (score >= 90.0)
  grade = 'A';
else if (score >= 80.0)
  grade = 'B';
else if (score >= 70.0)
  grade = 'C';
else if (score >= 60.0)
  grade = 'D';
else
  grade = 'F';
```

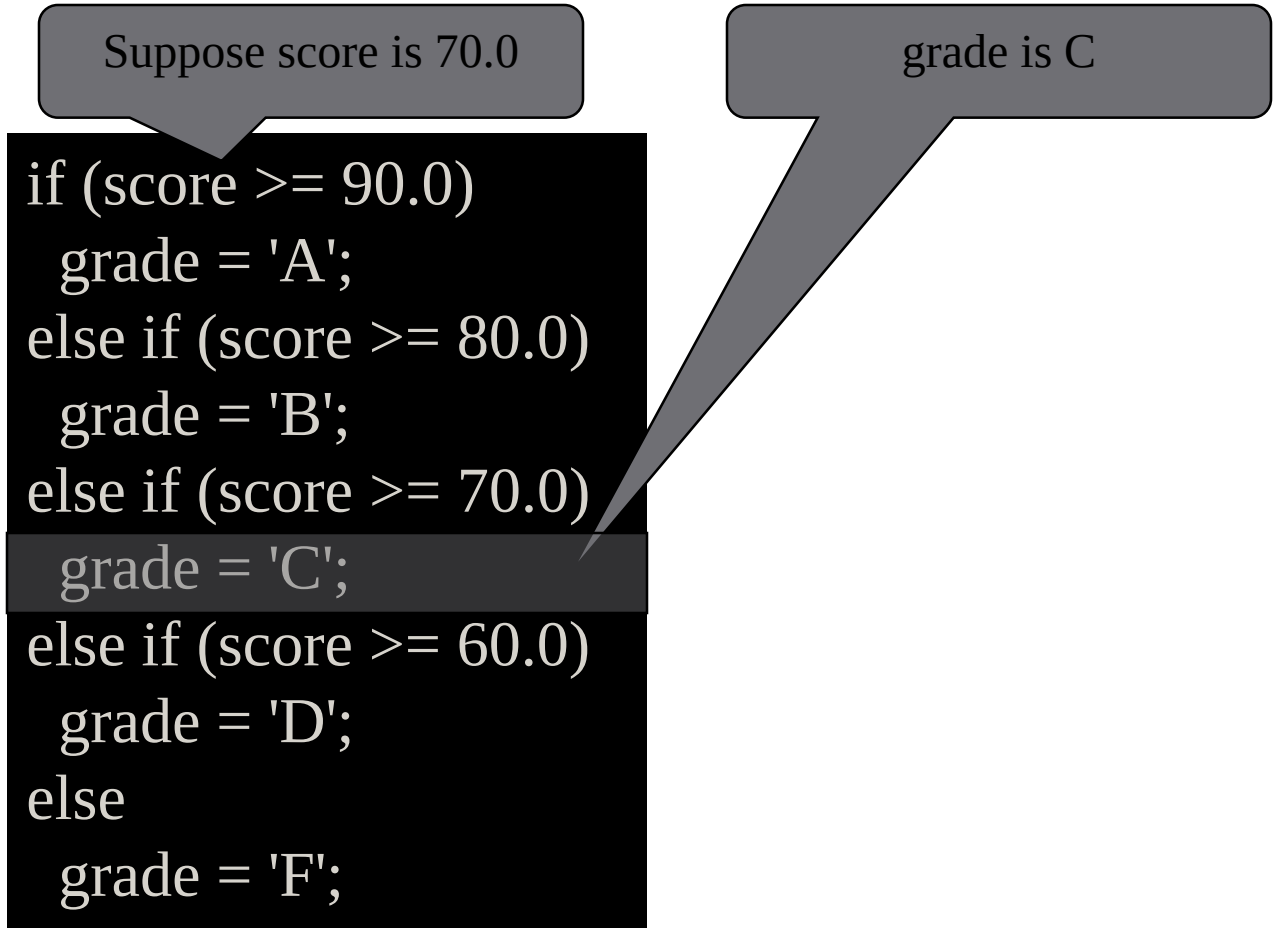
Trace if-else statement

Suppose score is 70.0

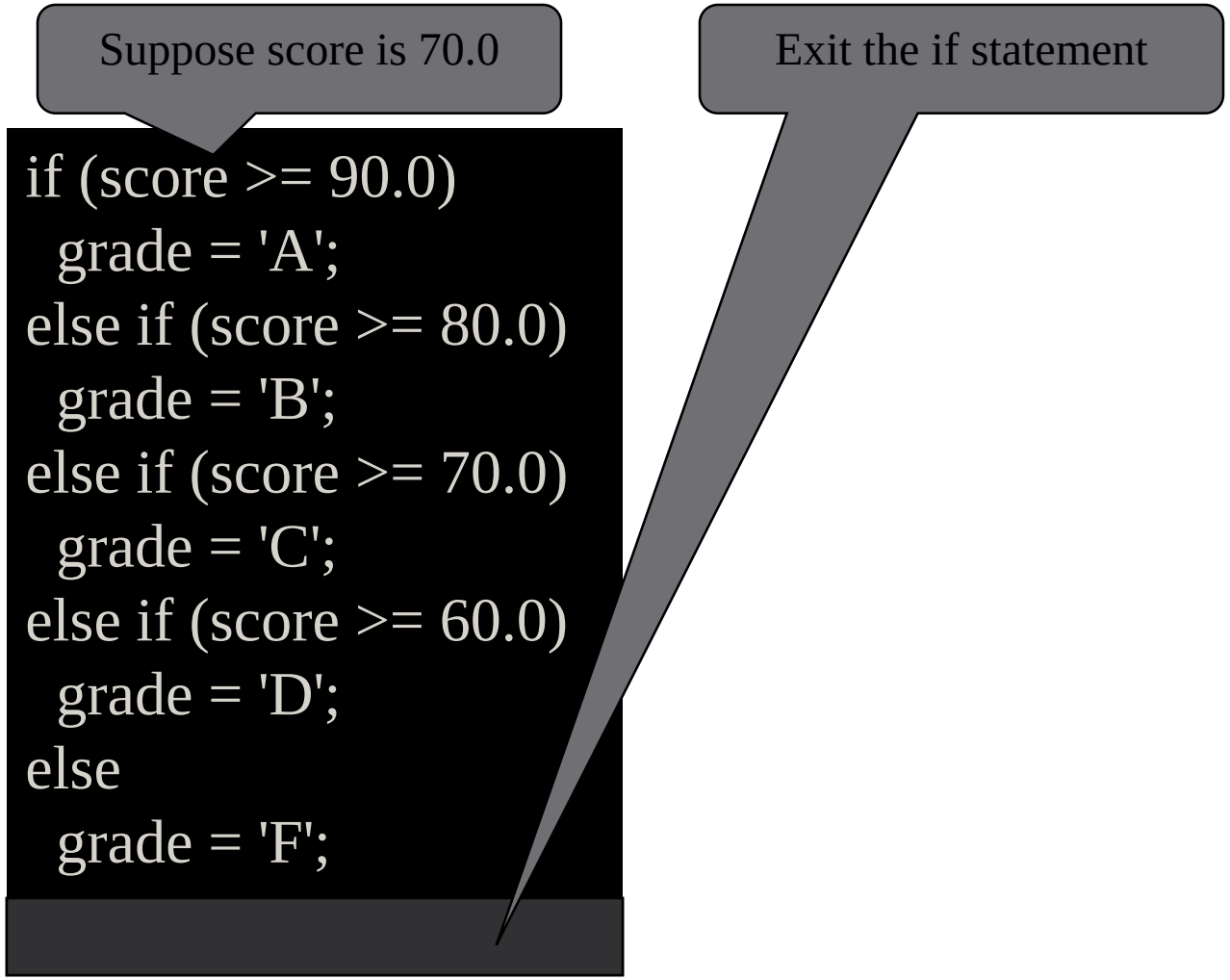
The condition is true

```
if (score >= 90.0)
  grade = 'A';
else if (score >= 80.0)
  grade = 'B';
else if (score >= 70.0)
  grade = 'C';
else if (score >= 60.0)
  grade = 'D';
else
  grade = 'F';
```

Trace if-else statement



Trace if-else statement



switch

- The switch is classified as a selection statement

```
switch (switch-expression) {
case value1: statements;
break;
                    ....
case valueN: statements;
break;
default: statements;
}
```

`switch` Statement Rules

The switch-expression must yield a value of char, byte, short, or int type and must always be enclosed in parentheses.

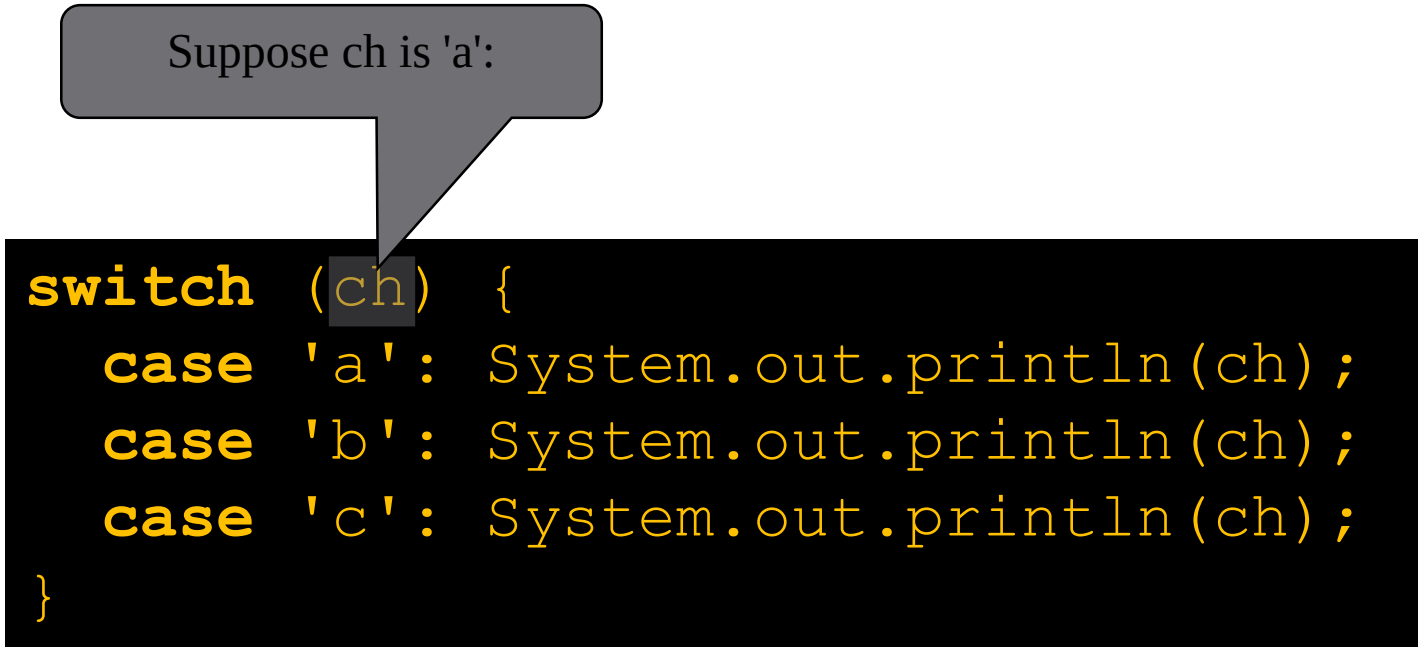
The value1, ..., and valueN must have the same data type as the value of the switch-expression.

The keyword break is optional, but it should be used at the end of each case in order to terminate the remainder of the switch statement. If the break statement is not present, the next case statement will be executed.

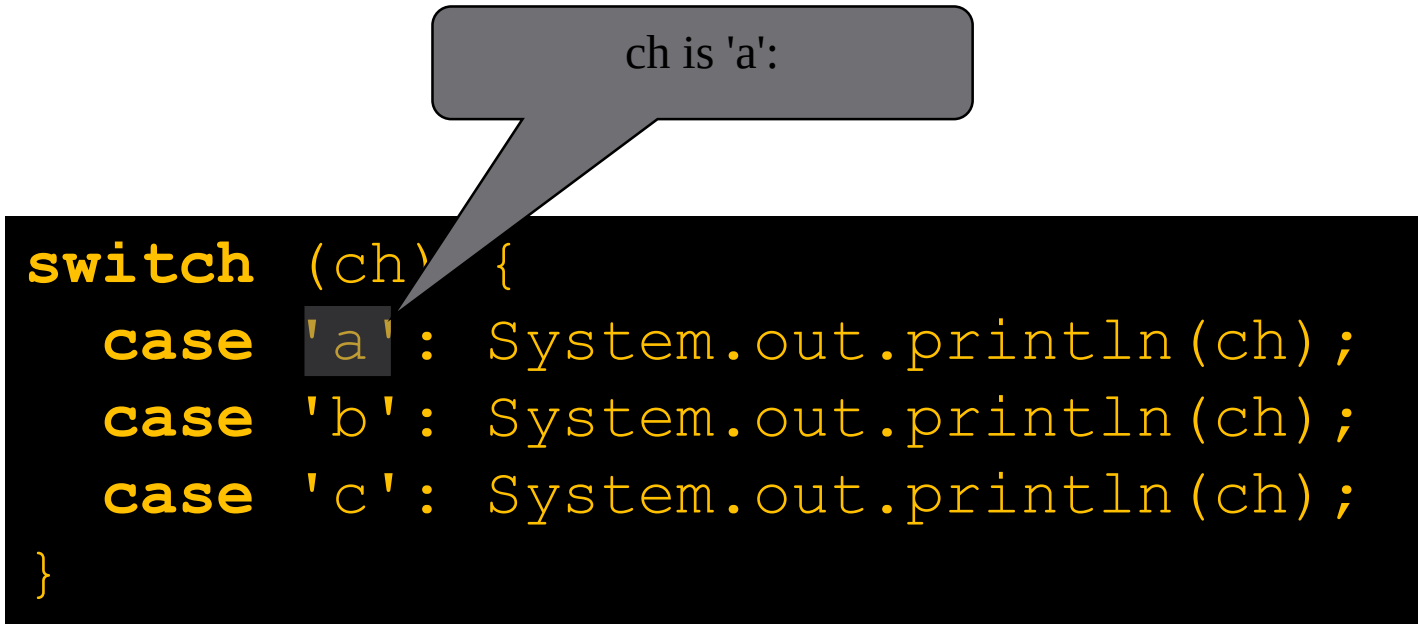
```
switch (switch-expression) {
  case value1:  statement(s)1;
        break;
  case value2: statement(s)2;
        break;
  ...
  case valueN: statement(s)N;
        break;
  default: statement(s)-for-default;
}
```

The default case, which is optional, can be used to perform actions when none of the specified cases matches the switch-expression.

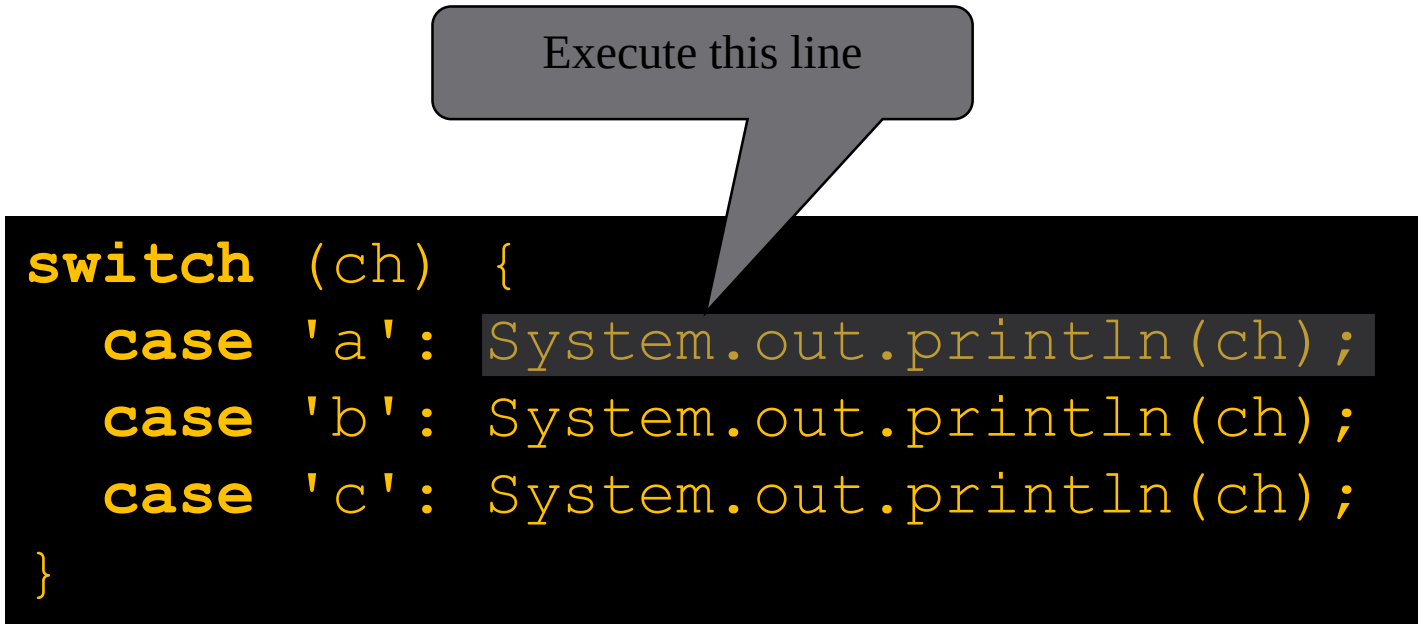
Trace switch statement



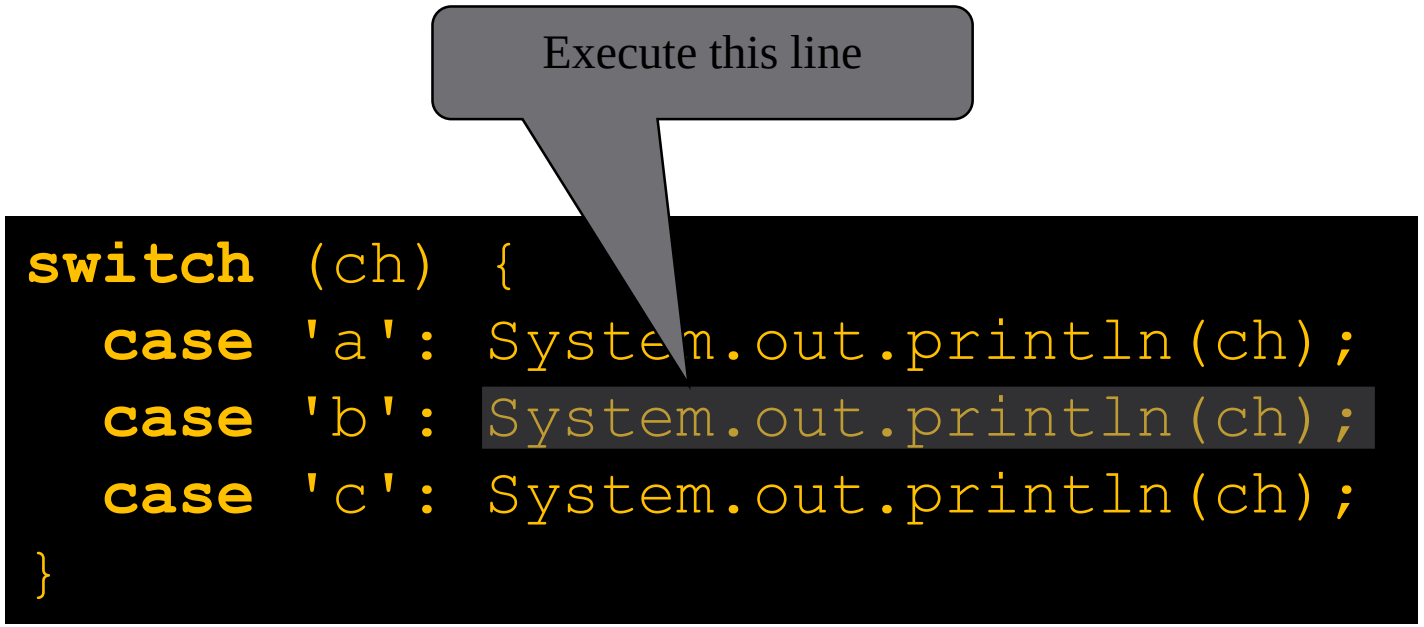
Trace switch statement



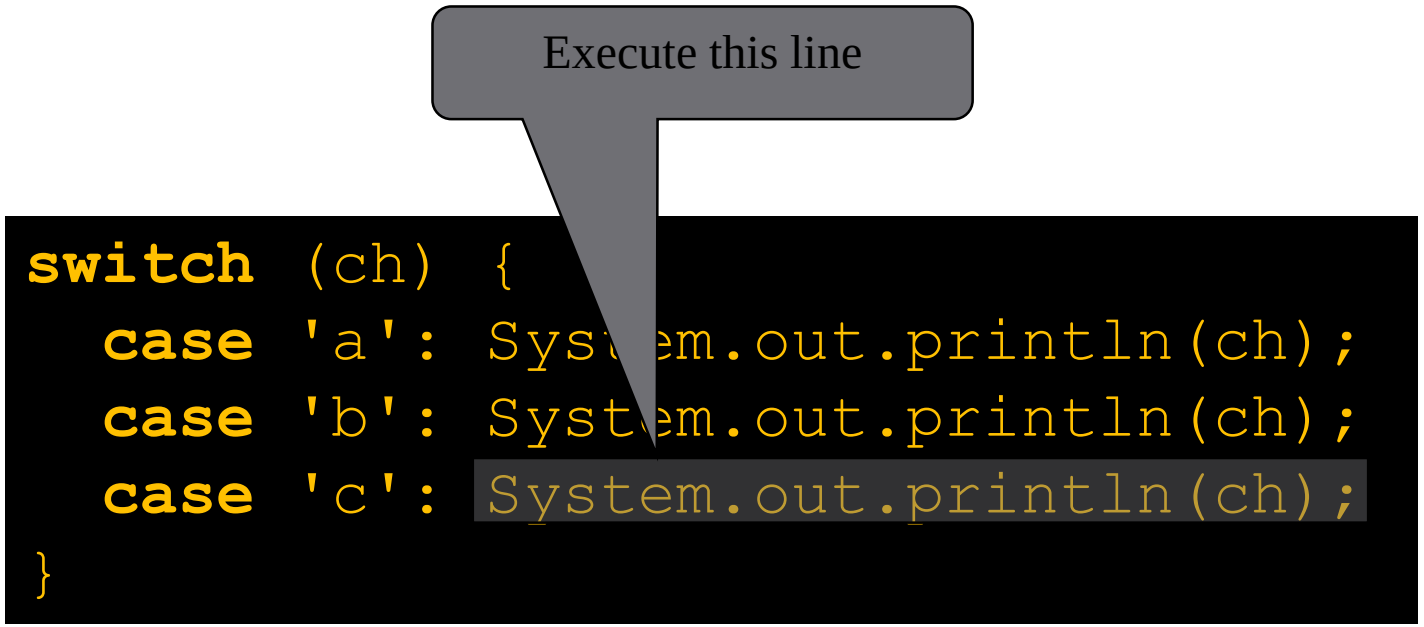
Trace switch statement



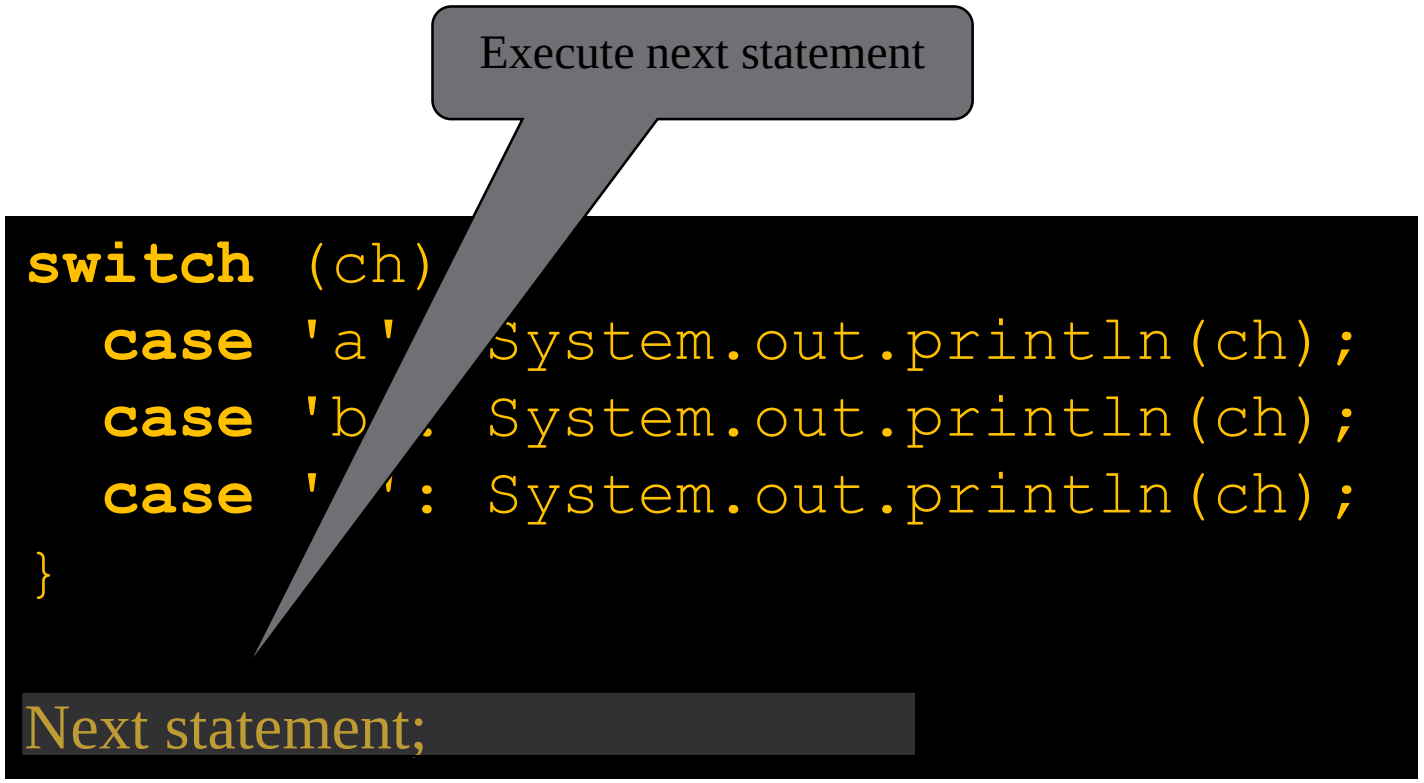
Trace switch statement



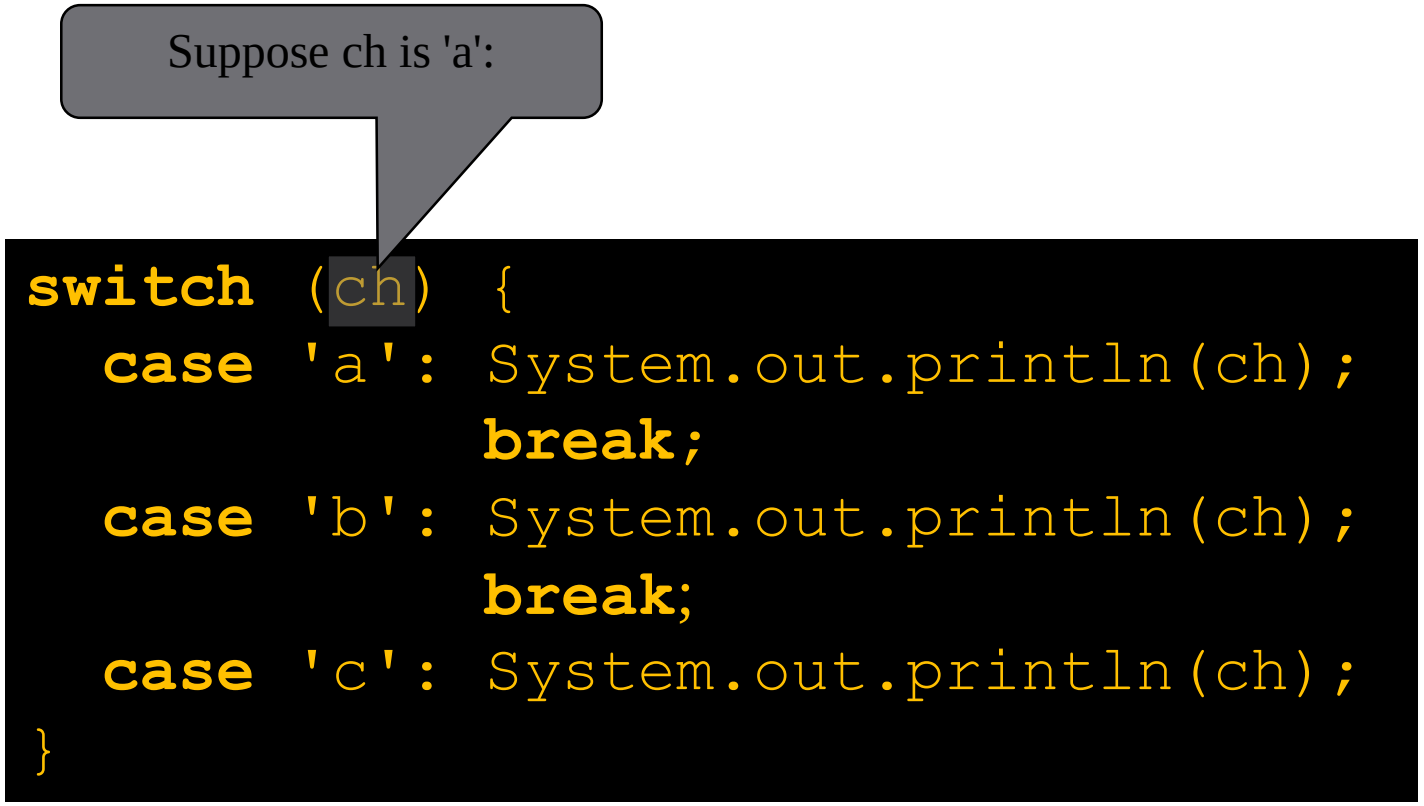
Trace switch statement



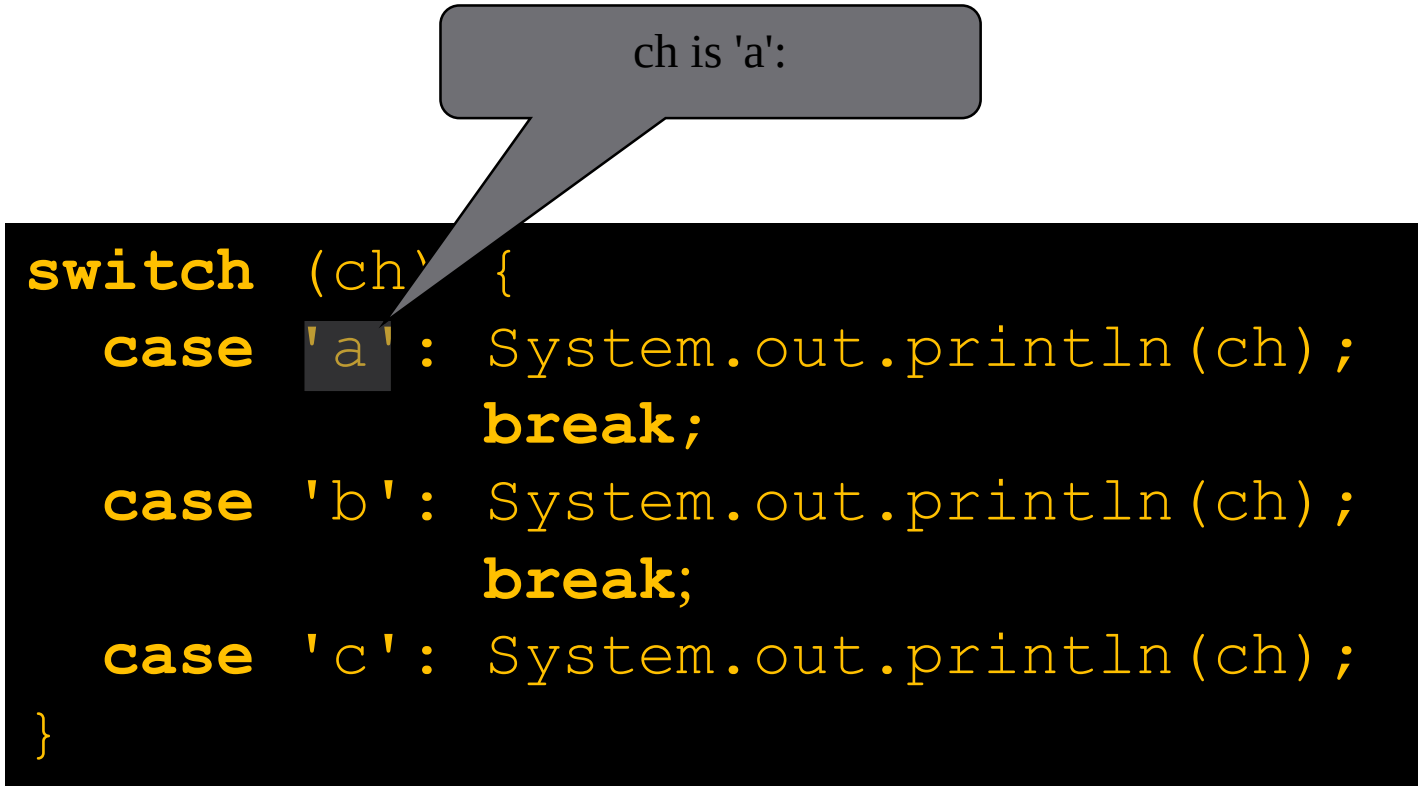
Trace switch statement



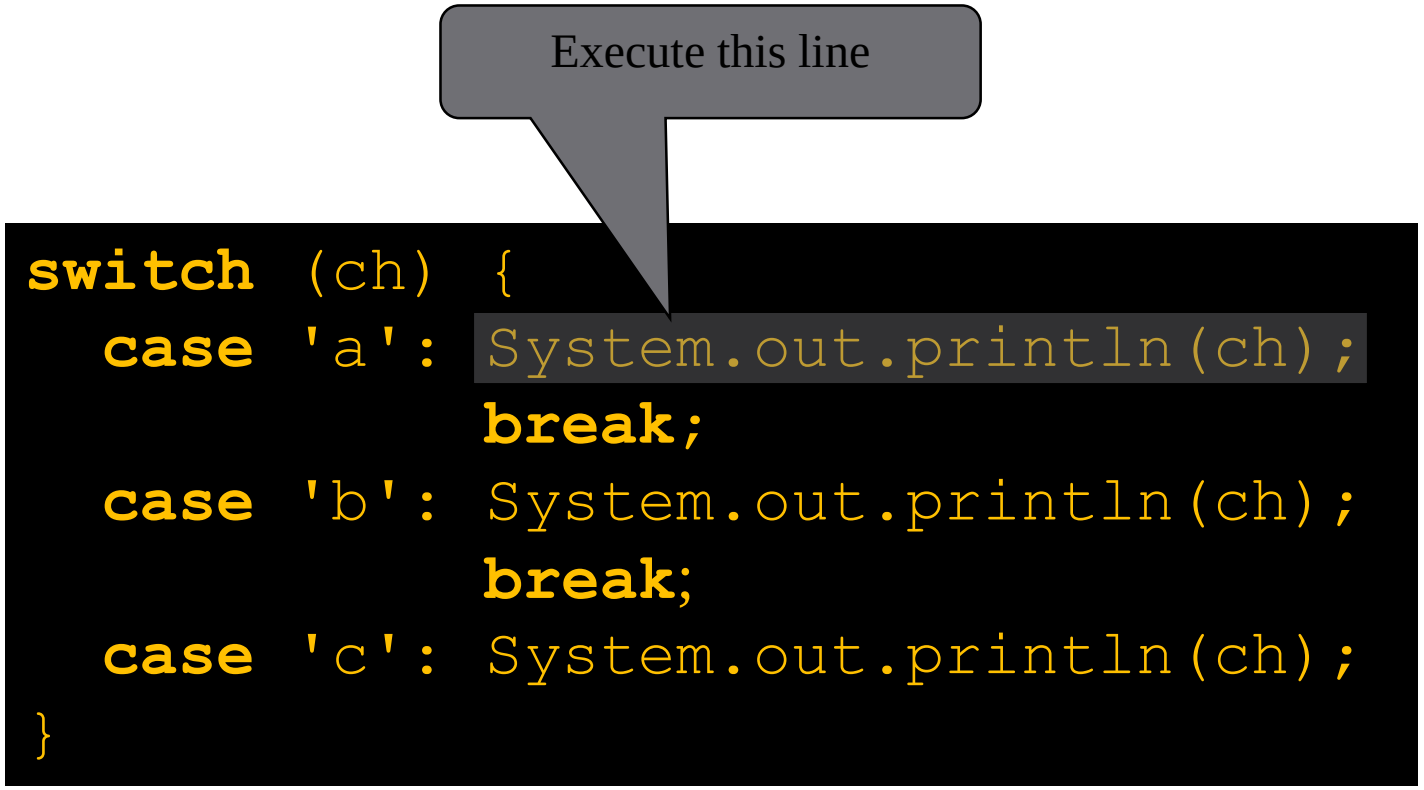
Trace switch statement



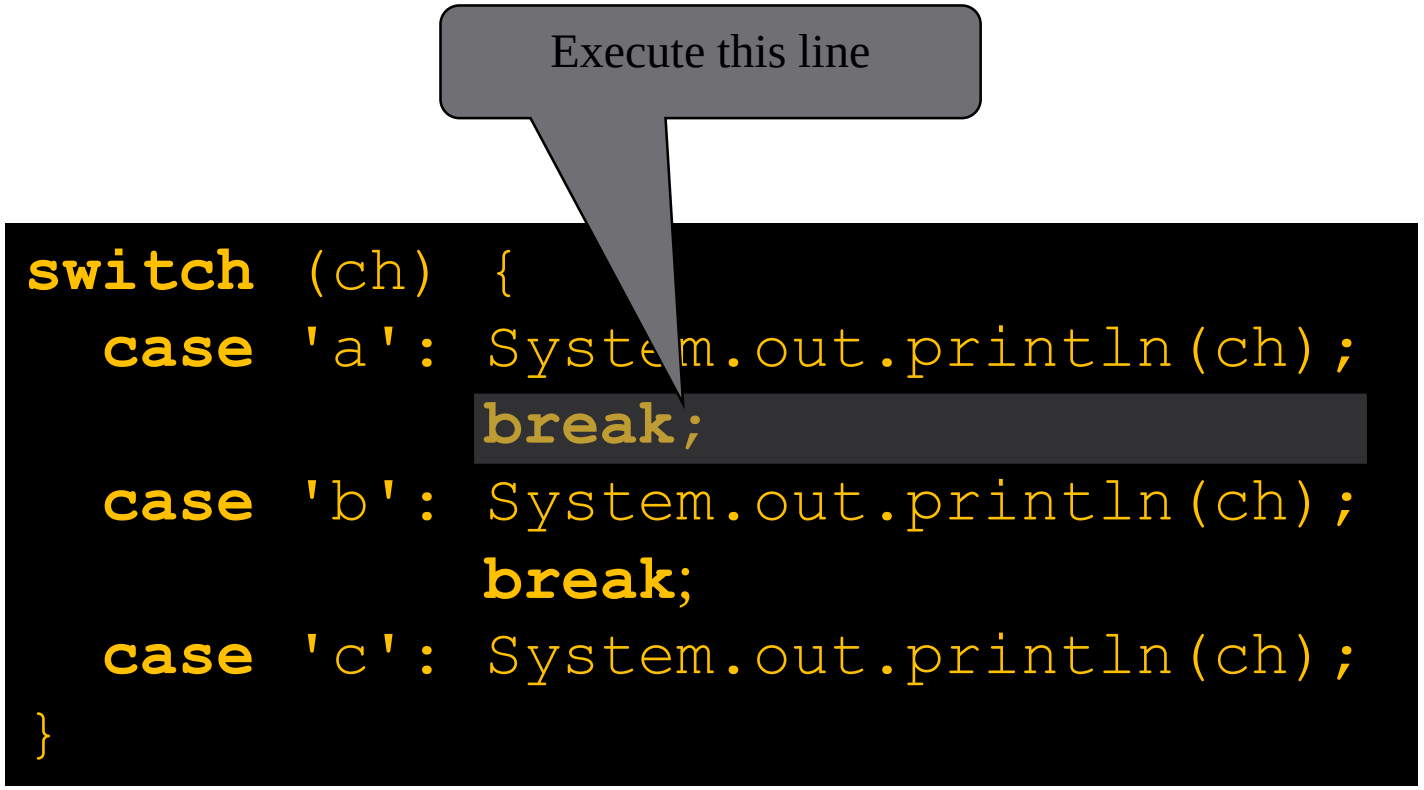
Trace switch statement



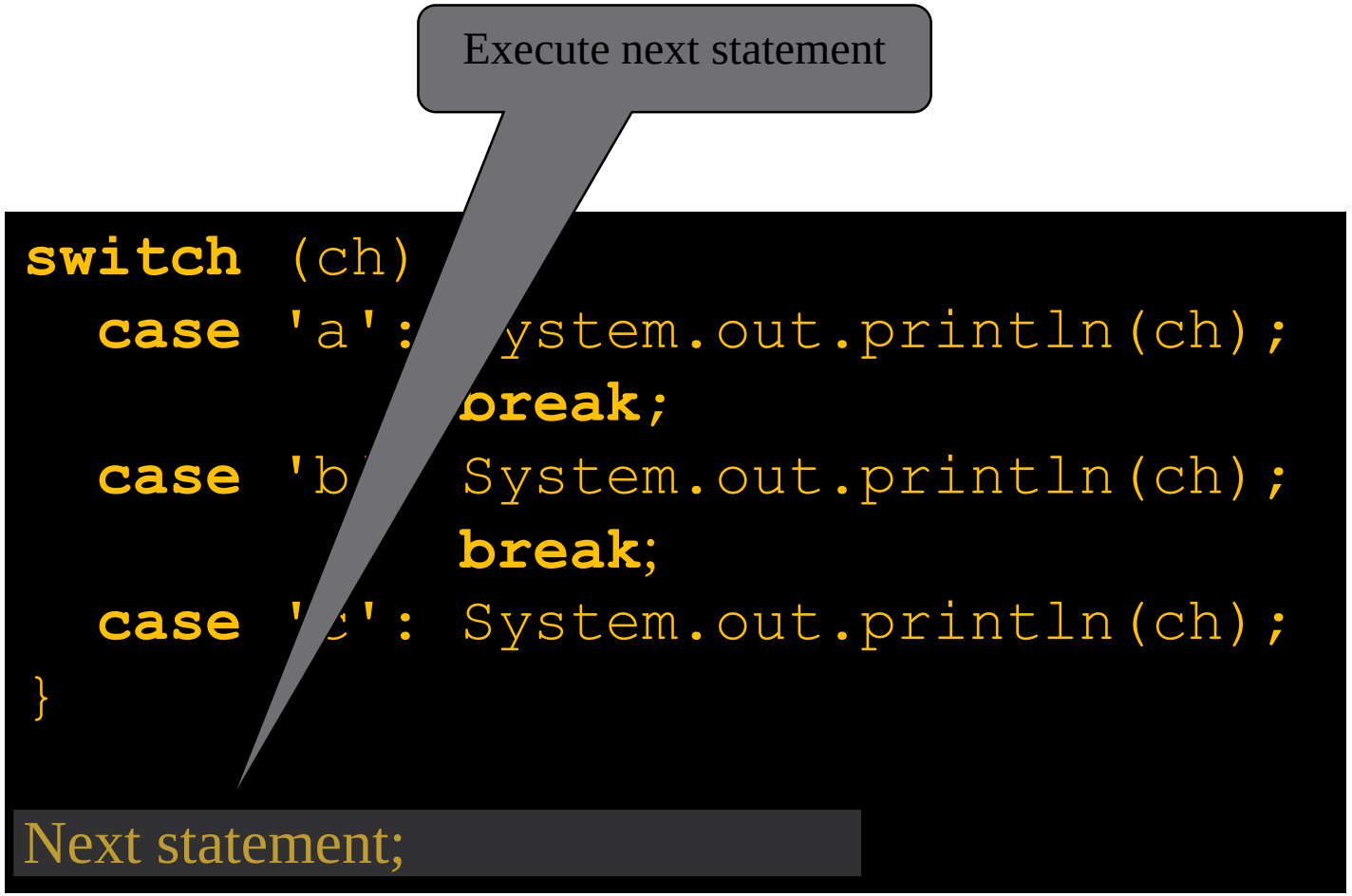
Trace switch statement



Trace switch statement



Trace switch statement



for Statement

- A for statement should have the following form:

```
for (initialization; condition; update) {
        statements;
}
for (k = 0, flag; k < 10 && flag; k++ ) {
        . . .
}
Enhanced for loop
        for (variable : Collection ) {
                . . .
        }
```

while, do - while Statements

- while, do-while and for control looping are classified as iteration statements.

```
while (condition) {
        statements;
}
```

```
do {
        statements;
} while (condition);
```

break - Labeled break

- A break “drops out of the bottom” of the loop. The break statement with no label attempts to transfer control to the innermost enclosing ***switch, for, while*** or ***do-while*** of immediately enclosing statement.
- A labeled break drops out of the bottom of the end of the loop denoted by the label.

Ex:

```
out:    for (int i = 0; i < 10; i++ ) {
                    for (int k = 0; k < 10; k++) {
                            if (i == k)
                            break out;
                    }
        System.out.println(i);
        }
```

continue – Labeled continue

- A plain continue goes to the top of the innermost loop and continues.
- A labeled continue goes to the label and re-enters the loop right after that label

Ex: Calculates the factorials of odd number

```
outerLoop: for (int i = 0; i < limit; i++ ) {
                    for (int k = 2; k < i; k++) {
                        if (i % 2)
                            continue outerLoop;
                    factory *= i;
                }
        }
```

Type Conversions

- Java is a strong typed language
- Implicit conversion for primitive value (automatic conversion): any numeric value can be assigned to any numeric value whose type supports a larger range of values.

  byte → short → int → long → float → double

- Explicit conversion – casting.
  - boolean type doesn't allow any casting at all.
  - A char can be cast to any integer type and vice versa excepting to a short type. When chart is cast to int type upper bits are filled with zeros.
  - Attention: interger types are converted by chopping off the upper bits. If the larger integer has a value outside the range off the smaller type, dropping the upper bits changes the value, including possibly changing sign.

    What is the value of y ???

    Ex: short x = -129;

    byte y = (byte)x;