

Java Summer 16

Selection Statements



Objectives

After you have read and studied this chapter, you should be able to

- Implement a selection control using **if** statements
- Implement a selection control using **switch** statements
- Write boolean expressions using relational and boolean expressions
- Evaluate given boolean expressions correctly
- Nest an **if** statement inside another **if** statement
- Describe how objects are compared
- Choose the appropriate selection control statement for a given task
- Define and use enumerated constants



The if Statement

```
int testScore;  
  
testScore = //get test score input
```

```
if (testScore < 70)
```

```
    System.out.println("You did not pass" );
```

```
else
```

```
    System.out.println("You did pass" );
```

This statement is executed if the testScore is less than 70.

This statement is executed if the testScore is 70 or higher.



Syntax for the if Statement

```
if ( <boolean expression> )
```

```
    <then block>
```

```
else
```

```
    <else block>
```

Boolean Expression

Then Block

Else Block

```
if ( testScore < 70 )
```

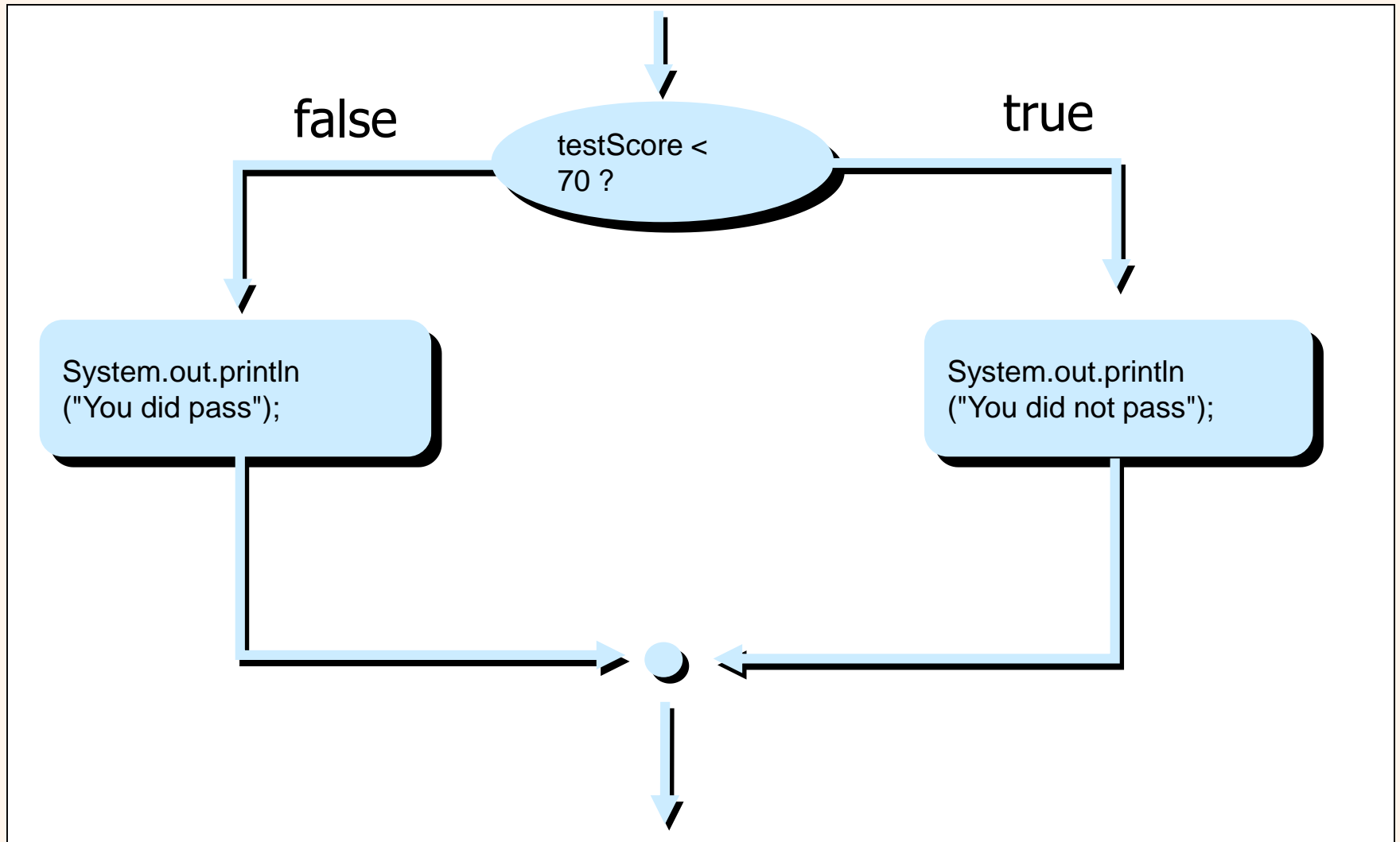
```
    System.out.println("You did not pass");
```

```
else
```

```
    System.out.println("You did pass ");
```



Control Flow





Relational Operators

<code><</code>	<code>//less than</code>
<code><=</code>	<code>//less than or equal to</code>
<code>==</code>	<code>//equal to</code>
<code>!=</code>	<code>//not equal to</code>
<code>></code>	<code>//greater than</code>
<code>>=</code>	<code>//greater than or equal to</code>

```
testScore < 80
testScore * 2 >= 350
30 < w / (h * h)
x + y != 2 * (a + b)
2 * Math.PI * radius <= 359.99
```



Compound Statements

- Use braces if the <then> or <else> block has multiple statements.

```
if (testScore < 70)
{
    System.out.println("You did not pass" );
    System.out.println("Try harder next time" );
}
else
{
    System.out.println("You did pass" );
    System.out.println("Keep up the good work" );
}
```

Then Block

Else Block



Style Guide

```
if ( <boolean expression> ) {  
    ...  
} else {  
    ...  
}
```

Style 1

```
if ( <boolean expression> )  
{  
    ...  
}  
else  
{  
    ...  
}
```

Style 2



The if-then Statement

```
if ( <boolean expression> )
```

```
<then block>
```

Boolean Expression

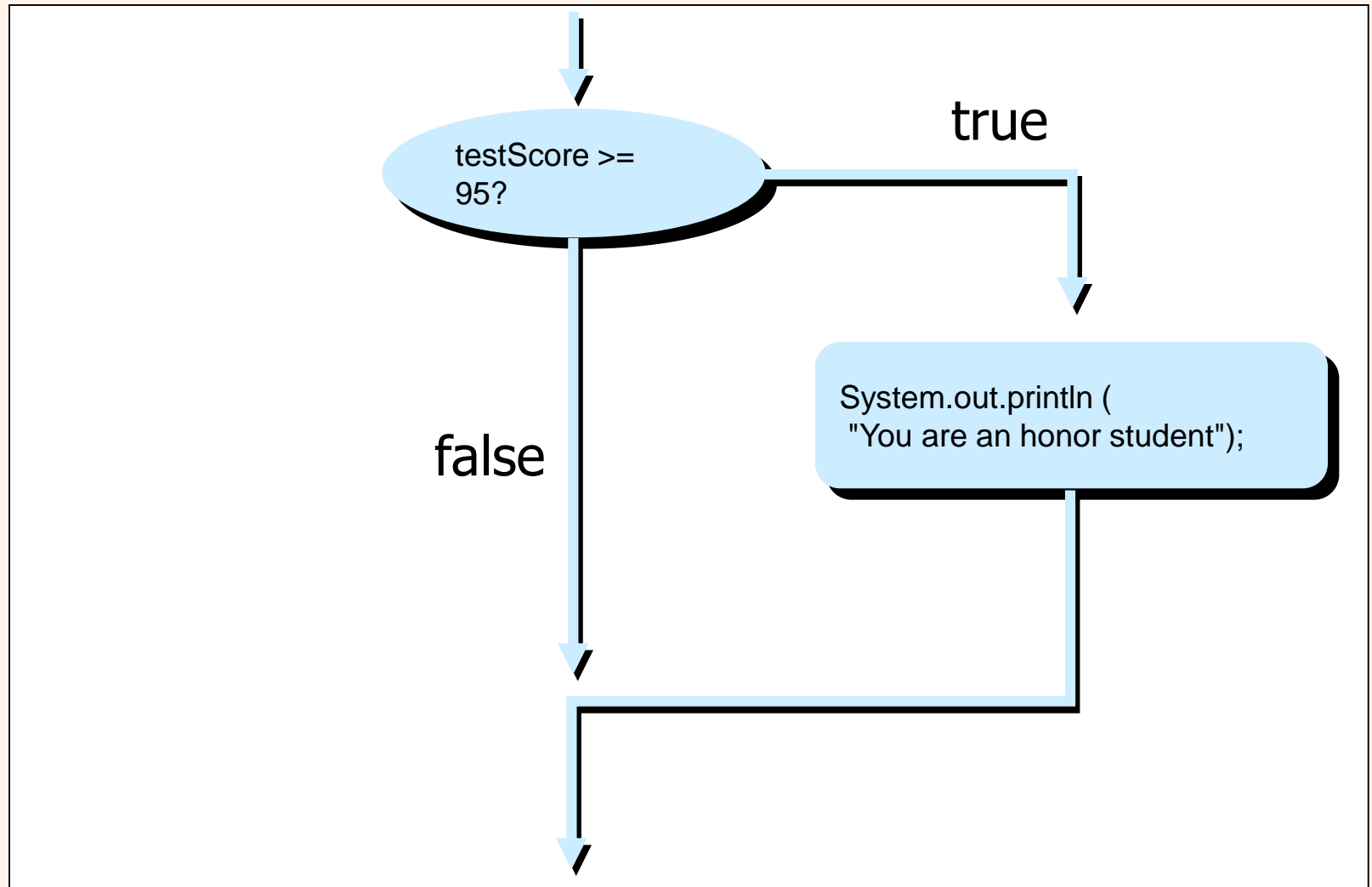
Then Block

```
if (     testScore >= 95     )
```

```
System.out.println("You are an honor student");
```



Control Flow of if-then





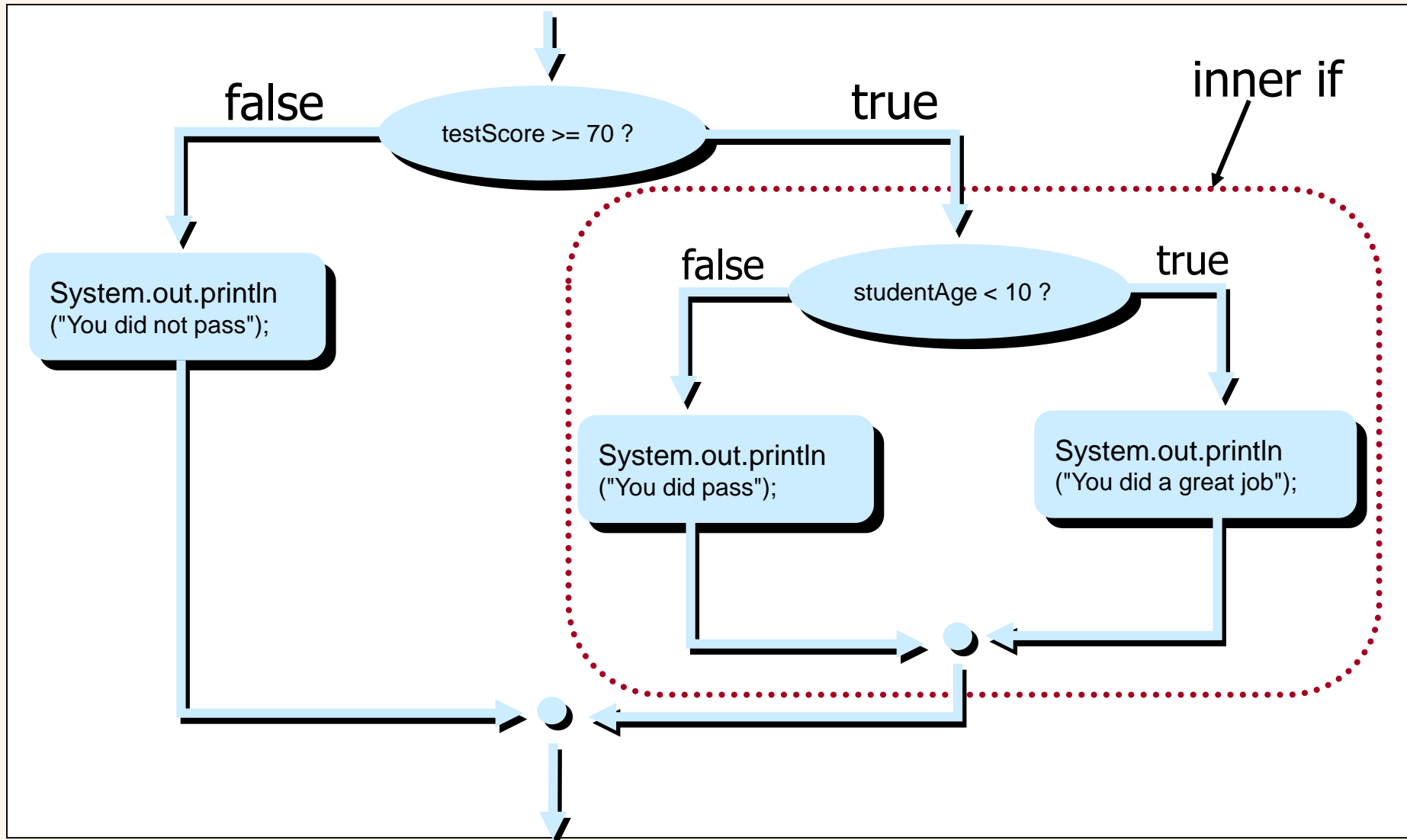
The Nested-if Statement

- The then and else block of an if statement can contain any valid statements, including other if statements. An if statement containing another if statement is called a nested-if statement.

```
if (testScore >= 70) {  
    if (studentAge < 10) {  
        System.out.println("You did a great job");  
    } else {  
        System.out.println("You did pass"); //test score >= 70  
                                              //and age >= 10  
    }  
} else { //test score < 70  
    System.out.println("You did not pass");  
}
```



Control Flow of Nested-if Statement





Writing a Proper if Control

```
if (num1 < 0)
    if (num2 < 0)
        if (num3 < 0)
            negativeCount = 3;
        else
            negativeCount = 2;
    else
        if (num3 < 0)
            negativeCount = 2;
        else
            negativeCount = 1;
else
    if (num2 < 0)
        if (num3 < 0)
            negativeCount = 2;
        else
            negativeCount = 1;
    else
        if (num3 < 0)
            negativeCount = 1;
        else
            negativeCount = 0;
```

```
negativeCount = 0;

if (num1 < 0)
    negativeCount++;
if (num2 < 0)
    negativeCount++;
if (num3 < 0)
    negativeCount++;
```

The statement

`negativeCount++;`

increments the variable by one



if – else if Control

Test Score	Grade
$90 \leq \text{score}$	A
$80 \leq \text{score} < 90$	B
$70 \leq \text{score} < 80$	C
$60 \leq \text{score} < 70$	D
$\text{score} < 60$	F

```
if (score >= 90)
    System.out.print("Your grade is A");

else if (score >= 80)
    System.out.print("Your grade is B");

else if (score >= 70)
    System.out.print("Your grade is C");

else if (score >= 60)
    System.out.print("Your grade is D");

else
    System.out.print("Your grade is F");
```



Matching else

Are **A** and **B** different?

A

```
if (x < y)
    if (x < z)
        System.out.print("Hello");
else
    System.out.print("Good bye");
```

B

```
if (x < y)
    if (x < z)
        System.out.print("Hello");
else
    System.out.print("Good bye");
```

Both **A** and **B** means...

```
if (x < y) {
    if (x < z) {
        System.out.print("Hello");
    } else {
        System.out.print("Good bye");
    }
}
```



Boolean Operators

- A *boolean operator* takes boolean values as its operands and returns a boolean value.
- The three boolean operators are
 - and: &&
 - or: ||
 - not !

```
if (temperature >= 65 && distanceToDestination < 2) {  
    System.out.println("Let's walk");  
} else {  
    System.out.println("Let's drive");  
}
```




Semantics of Boolean Operators

- Boolean operators and their meanings:

P	Q	P && Q	P Q	!P
false	false	false	false	true
false	true	false	true	true
true	false	false	true	false
true	true	true	true	false



De Morgan's Law

- De Morgan's Law allows us to rewrite boolean expressions in different ways

Rule 1: $\neg(P \ \&\& \ Q) \iff \neg P \ || \ \neg Q$

Rule 2: $\neg(P \ || \ Q) \iff \neg P \ \&\& \ \neg Q$

$\neg(\text{temp} \geq 65 \ \&\& \ \text{dist} < 2)$

$\iff \neg(\text{temp} \geq 65) \ || \ \neg(\text{dist} < 2)$ by Rule 1

$\iff (\text{temp} < 65 \ || \ \text{dist} \geq 2)$



Short-Circuit Evaluation

- Consider the following boolean expression:

$$x > y \ || \ x > z$$

- The expression is evaluated left to right. If $x > y$ is true, then there's no need to evaluate $x > z$ because the whole expression will be true whether $x > z$ is true or not.
- To stop the evaluation once the result of the whole expression is known is called *short-circuit evaluation*.
- What would happen if the short-circuit evaluation is not done for the following expression?

$$z == 0 \ || \ x / z > 20$$



Operator Precedence Rules

Group	Operator	Precedence	Associativity
Subexpression	()	10 (If parentheses are nested, then innermost subexpres- sion is evaluated first.)	Left to right
Postfix increment and decrement operators	++ --	9	Right to left
Unary operators	- !	8	Right to left
Multiplicative operators	* / %	7	Left to right
Additive operators	+ -	6	Left to right
Relational operators	< <= > >=	5	Left to right
Equality operators	== !=	4	Left to right
Boolean AND	&&	3	Left to right
Boolean OR		2	Left to right
Assignment	=	1	Right to left



Boolean Variables

- The result of a boolean expression is either **true** or **false**. These are the two values of data type **boolean**.
- We can declare a variable of data type **boolean** and assign a boolean value to it.

```
boolean pass, done;  
pass = 70 < x;  
done = true;  
if (pass) {  
    ...  
} else {  
    ...  
}
```



The switch Statement

```
Scanner scanner = new Scanner(System.in);

System.out.println( "Grade (Frosh-1,Soph-2,...):");
int gradeLevel = scanner.nextInt();

switch (gradeLevel) {
    case 1: System.out.print("Go to the Gymnasium");
            break;

    case 2: System.out.print("Go to the Science Auditorium");
            break;

    case 3: System.out.print("Go to Harris Hall Rm A3");
            break;

    case 4: System.out.print("Go to Bolt Hall Rm 101");
            break;

}
```

← This statement is executed if the gradeLevel is equal to 1.

← This statement is executed if the gradeLevel is equal to 4.



Syntax for the switch Statement

```
switch ( <arithmetic expression> ) {  
    <case label 1> : <case body 1>  
    ...  
    <case label n> : <case body n>  
}
```

```
switch ( gradeLevel ) {  
    case 1: System.out.print( "Go to the Gymnasium" );  
           break;  
    case 2: System.out.print( "Go to the Science Auditorium" );  
           break;  
    case 3: System.out.print( "Go to Harris Hall Rm A3" );  
           break;  
    case 4: System.out.print( "Go to Bolt Hall Rm 101" );  
           break;  
}
```

Arithmetic Expression

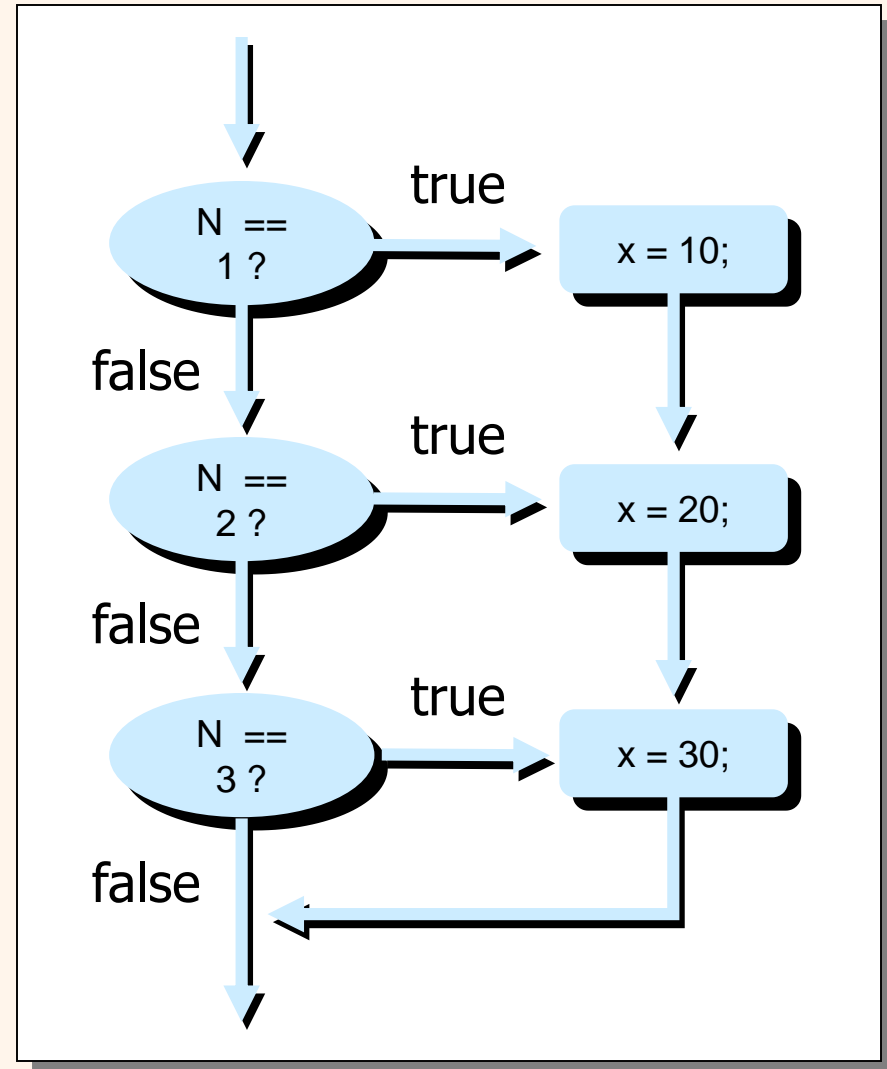
**Case
Label**

**Case
Body**



switch With No break Statements

```
switch ( N ) {  
    case 1: x = 10;  
    case 2: x = 20;  
    case 3: x = 30;  
}
```





switch With break Statements

```
switch ( N ) {  
    case 1: x = 10;  
            break;  
    case 2: x = 20;  
            break;  
    case 3: x = 30;  
            break;  
}
```

