

Chapter 6 - Control Statements

Section 6.1 - Logical Operators

No program is perfect! - read page 199

Java includes three logical operators along with the previously mentioned relational operators.

Relational Operators

- 1) >
- 2) <
- 3) >=
- 4) <=
- 5) ==
- 6) !=

Logical Operators

- 1) && - and
- 2) || - or
- 3) ! - not

- 1) If the sun is shining AND it is 8 a.m. then let's go for a walk else let's stay home.
- 2) If the sun is shining OR it is 8 a.m. then let's go for a walk else let's stay home.
- 3) If NOT the sun is shining then let's go for a walk else let's stay home.

1)

THE SUN IS SHINING	IT IS 8 A.M.	THE SUN IS SHINING AND IT IS 8 A.M.	ACTION TAKEN
true	true	true	go for a walk
true	false	false	stay at home
false	true	false	stay at home
false	false	false	stay at home

2)

THE SUN IS SHINING	IT IS 8 A.M.	THE SUN IS SHINING OR IT IS 8 A.M.	ACTION TAKEN
true	true	true	go for a walk
true	false	true	go for a walk
false	true	true	go for a walk
false	false	false	stay at home

3)

THE SUN IS SHINING	NOT THE SUN IS SHINING	ACTION TAKEN
true	false	stay at home
false	true	go for a walk

Truth Tables for Logical Operators

and	T	F
T	T	F
F	F	F

or	T	F
T	T	T
F	T	F

not (T) = F
not (F) = T

P	Q	P AND Q	P OR Q	NOT P
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

Order of Operations (Precedence Rules)

OPERATION	SYMBOL	PRECEDENCE (FROM HIGHEST TO LOWEST)	ASSOCIATION
Grouping	()	1	Not applicable
Method selector	.	2	Left to right
Unary plus	+	3	Not applicable
Unary minus	-	3	Not applicable
Not	!	3	Not applicable
Multiplication	*	4	Left to right
Division	/	4	Left to right
Remainder or modulus	%	4	Left to right
Addition	+	5	Left to right
Subtraction	-	5	Left to right
Relational operators	< <= > >= == !=	6	Not applicable
And	&&	8	Left to right
Or		9	Left to right
Assignment operators	= *= /= %= += -=	10	Right to left

Complex Boolean Expressions

```
if (the sun shines && (you have the time || it is Sunday))  
    let's go for a walk;  
else  
    let's stay home;
```

P: THE SUN SHINES	Q: YOU HAVE TIME	R: IT IS SUNDAY	P && (Q R)	ACTION TAKEN
true	true	true	true	walk
true	true	false	true	walk
true	false	true	true	walk
true	false	false	false	stay home
false	true	true	false	stay home
false	true	false	false	stay home
false	false	true	false	stay home
false	false	false	false	stay home

- 1) if (the sun is shining AND it is 8 a.m.) OR (NOT your brother is visiting) then
 let's go for a walk
 else
 let's stay at home

P	Q	R	$(P \wedge Q) \vee (\neg R)$	
T	T	T	T	walk
T	T	F	T	walk
T	F	T	T	home
T	F	F	T	walk
F	T	T	T	home
F	T	F	T	walk
F	F	T	T	home
F	F	F	T	walk

- 2) if the sun is shining AND (it is 8 a.m. OR (NOT your brother is visiting)) then
 let's go for a walk
 else
 let's stay at home

P	Q	R	$P \wedge (Q \vee \neg R)$	
T	T	T	T	w
T	T	F	T	w
T	F	T	F	
T	F	F	T	w
F	T	T	F	
F	T	F	T	
F	F	T	F	
F	F	F	F	

Using Logical Operators

```
if (score1 >= 90 && score2 >= 90)
    System.out.println("Qualified to be a manager.");

if (score1 >= 90 || score2 >= 90)
    System.out.println("Qualified to be a supervisor.");

if (score1 >= 70 || score2 >=70) && !(score1 < 50 || score2 < 50))
    System.out.println("Qualified to be a clerk.");
```

What happens for the following test scores:

- 1) score1 = 100, score2 = 100
- 2) score1 = 100, score2 = 80
- 3) score1 = 80, score2 = 60
- 4) score1 = 80, score2 = 40
- 5) score1 = 40, score2 = 40

Previous statements rewritten

```
boolean bothHigh           = (score1 >= 90 && score2 >= 90);
boolean atLeastOneHigh     = (score1 >= 90 || score2 >= 90);
boolean atLeastOneModerate = (score1 >=70 || score2 >=70);
boolean noLow              = !(score1 < 50 || score2 <50);
```

```
if (bothHigh == True)
    System.out.println("Qualified to be a manager.");

if (atLeastOneHigh)
    System.out.println("Qualified to be a supervisor.");

if (atLeastOneModerate && noLow)
    System.out.println("Qualified to be a clerk.");
```

Boolean Equivalences (De Morgan's Laws)

1) $\overline{(p \vee q)} \iff \overline{p} \wedge \overline{q}$

2) $\overline{(p \wedge q)} \iff \overline{p} \vee \overline{q}$

3) $p \vee (q \wedge r) \iff (p \vee q) \wedge (p \vee r)$

4) $p \wedge (q \vee r) \iff (p \wedge q) \vee (p \wedge r)$

Prove #1

	p	q	$\overline{(p \vee q)}$	
	T	T	F	
	T	F	F	
	F	T	F	
	F	F	T	

	p	q	$\overline{p} \wedge \overline{q}$	
	T	T	F	
	T	F	F	
	F	T	F	
	F	F	T	

Prove #3

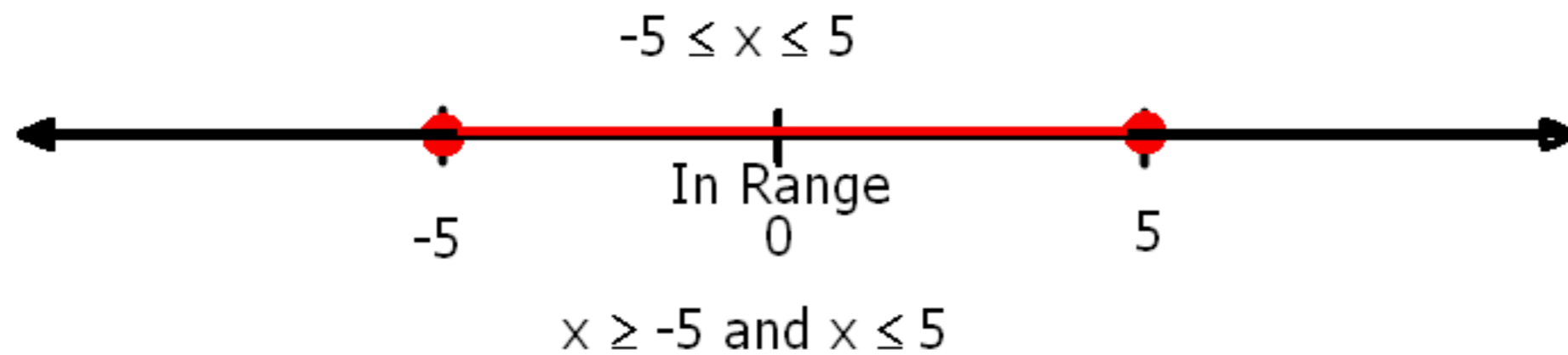
$p \parallel (q \ \&\& \ r)$

p	q	r	
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	T
F	T	T	T
F	T	F	F
F	F	T	F
F	F	F	T

$(p \parallel q) \ \&\& \ (p \parallel r)$

p	q	r	
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	T
F	T	T	T
F	T	F	F
F	F	T	F
F	F	F	F

Algebraic Example



Indicate when out of range.

- 1) `if (!(x >= -5 && x <= 5)) System.out.println("Out of range.");`
- 2) `if (!(x >= -5) || !(x <= 5)) System.out.println("Out of range.");`
- 3) `if (x < -5 || x > 5) System.out.println("Out of range.");`

Short-circuit Evaluaton

1) Read page 205

2) if (age == 25 && school.equals("Greenwood")) System.out.println("Give \$1000000.");
F

3) if (age <= 21 || school.equals("Central")) Sytem.out.println("Give \$1000000.");
T

4) if (count > 0 && sum/count > 10)
 System.out.println("average > 10");
else
 System.out.println("count = 0 or average <= 10");

String Methods

String x = "Gators";
String y = " Gators ";
String z;

1) equals ✓
if (x.equals("CatDog")) System.out.println("Password Correct!");

2) trim ✓
z = y.trim();

3) toUpperCase
z = x.toUpperCase();

4) toLowerCase
z = x.toLowerCase();

5) substring
z = x.substring(2,5); z oh

6) length
System.out.println(x.length()); 6

(2, 5)

Case Study #1 - Compute Weekly Pay - Server Program ("Employee.java")

```
/* Employee.java
```

```
1. Instance variables: name, type, rate, hours
```

```
2. Methods to
```

```
get data validation rules
```

```
set instance variables if data is valid
```

```
get name and pay */
```

```
public class Employee {
```

```
    // Private Instance Variables:
```

```
    private String name;
```

```
    private int type;
```

```
    private double rate;
```

```
    private int hours;
```

```
    // Public Methods:
```

```
    public Employee(){
```

```
        name = "";
```

```
        type = 0;
```

```
        rate = 0;
```

```
        hours = 0;
```

```
    }
```

```
    public String getNameRules(){
```

```
        return "nonblank";
```

```
    }
```

```
    public String getTypeRules(){
```

```
        return "1 or 2";
```

```
    }
```

```
    public String getRateRules(){
```

```
        return "between 6.75 and 30.50, inclusive";
```

```
    }
```

```
    public String getHoursRules(){
```

```
        return "between 1 and 60, inclusive";
```

```
    }
```

```
    public boolean setName(String nm){
```

```
        if (nm.equals(""))
```

```
            return false;
```

```
        else{
```

```
            name = nm;
```

```
            return true;
```

```
        }
```

```
    }
```

```
    public boolean setType(int tp){
```

```
        if (tp != 1 && tp != 2)
```

```
            return false;
```

```
        else{
```

```
            type = tp;
```

```
            return true;
```

```
        }
```

```
    }
```

```
    public boolean setRate(double rt){
```

```
        if (!(6.75 <= rt && rt <= 30.50))
```

```
            return false;
```

```
        else{
```

```
            rate = rt;
```

```
            return true;
```

```
        }
```

```
    }
```

```
    public boolean setHours(int hrs){
```

```
        if (!(1 <= hrs && hrs <= 60))
```

```
            return false;
```

```
        else{
```

```
            hours = hrs;
```

```
            return true;
```

```
        }
```

```
    }
```

```
    public String getName(){
```

```
        return name;
```

```
    }
```

```
    public double getPay(){
```

```
        double pay;
```

```
        if (hours <= 40 || type == 2)
```

```
            pay = rate * hours;
```

```
        else
```

```
            pay = rate * 40 + rate * 2 * (hours - 40);
```

```
        return pay;
```

```
    }
```

```
}
```

Case Study #1 - Compute Weekly Pay - Client Program ("PayrollSystemApp")

```
/* Case Study 6.1: PayrollSystemApp.java
```

1. Request employee name, type, pay rate, and hours.
2. Print employee name and pay.
3. Repeat until the name is blank.*/

```
import java.util.Scanner;
```

```
public class PayrollSystemApp {
```

```
    public static void main(String [] args) {
```

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

```
        Employee emp; // employee
```

```
        String name; // name
```

```
        int type; // type
```

```
        double rate; // hourly pay rate
```

```
        int hours; // hours worked
```

```
        String prompt; // user prompt;
```

```
        while (true){
```

```
            // Get the name and break if blank
```

```
            System.out.println("Enter employee data");
```

```
            System.out.print(" Name (or blank to quit): ");
```

```
            name = reader.nextLine();
```

```
            name = name.trim(); // Trim off leading and trailing spaces
```

```
            if (name.length() == 0) break;
```

```
            emp = new Employee();
```

```
            emp.setName(name);
```

```
            // Get the type until valid
```

```
            while (true){
```

```
                prompt = " Type (" + emp.getTypeRules() + "): ";
```

```
                System.out.print(prompt);
```

```
                type = reader.nextInt();
```

```
                if (emp.setType(type)) break;
```

```
            }
```

```
            // Get the hourly pay rate until valid
```

```
            while (true){
```

```
                prompt = " Hourly rate (" + emp.getRateRules() + "): ";
```

```
                System.out.print(prompt);
```

```
                rate = reader.nextDouble();
```

```
                if (emp.setRate(rate)) break;
```

```
            }
```

```
            // Get the hours worked until valid
```

```
            // To illustrate the possibilities we compress this code
```

```
            // into a single hard-to-read statement.
```

```
            System.out.print("Hours worked (" +
```

```
                emp.getHoursRules() + "): ");
```

```
            while (!emp.setHours(reader.nextInt())) {
```

```
                System.out.print("Hours worked (" +
```

```
                    emp.getHoursRules() + "): ");
```

```
            }
```

```
            // Consume the trailing newline
```

```
            reader.nextLine();
```

```
            // Print the name and pay
```

```
            System.out.println(" The weekly pay for " + emp.getName() +
```

```
                " is $" + emp.getPay());
```

```
        }
```

```
    }
```

```
}
```

Section 6.2 - Testing If Statements

- 1) Read section 6.2 pages 212-213
- 2) Test data for the payroll program

TYPE OF TEST	DATA USED
Code coverage	employee type: 1 hourly rate: 10 hours worked: 30 and 50
Boundary conditions	employee type: 1 hourly rate: 10 hours worked: 39, 40, and 41
Extreme conditions	employee type: 1 hourly rate: 10 hours worked: 0 and 168
Tests when the employee type is 2	employee type: 2 hourly rate: 10 hours worked: 30 and 50
Data validation rules	type: 0, 1, 2, and 3 hourly rate: 6.74, 6.75, 10, 30.50, and 30.51 hours worked: 0, 1, 30, 60, and 61

Example: Describe appropriate test data for the following code:

```
System.out.println("Enter a number:");  
x = reader.nextDouble();
```

```
if (x >= -5 && x <= 5)  
    System.out.println("In Range.");  
else  
    System.out.println("Out of Range.");
```

Type of Test	Data Used
Code Coverage	-10, 10, 0
Boundary Conditions	-6, -5, -4 4, 5, 6
Extreme Conditions	100, -100
Data Validation Rules	String , Double, Int

Section 6.3 - Nested if Statements

A program's logic can become complex. The logical operators (&&, ||, and !) provide one way to deal with this complexity. Also nested if statements can be used as an alternative to the logical operators.

Example #1 (using logical operators)

```
if (age==18 && school.equals("Greenwood")) {
    System.out.println("You get a million dollars.");
} else {
    System.out.println("You do not get a million dollars.");
}
```

Example #2 (using nested if statements)

```
if (age==18) {
    if (school.equals("Greenwood")) {
        System.out.println("You get a million dollars.");
    } else {
        System.out.println("You do not get a million dollars.");
    }
} else {
    System.out.println("You do not get a million dollars.");
}
```

Example #3 (logical operators and no nesting)

```
if (average>=90) System.out.println("A");
if (average>=80 && average<90) System.out.println("B");
if (average>=70 && average<80) System.out.println("C");
if (average>=60 && average<70) System.out.println("D");
if (average<60) System.out.println("F");
```

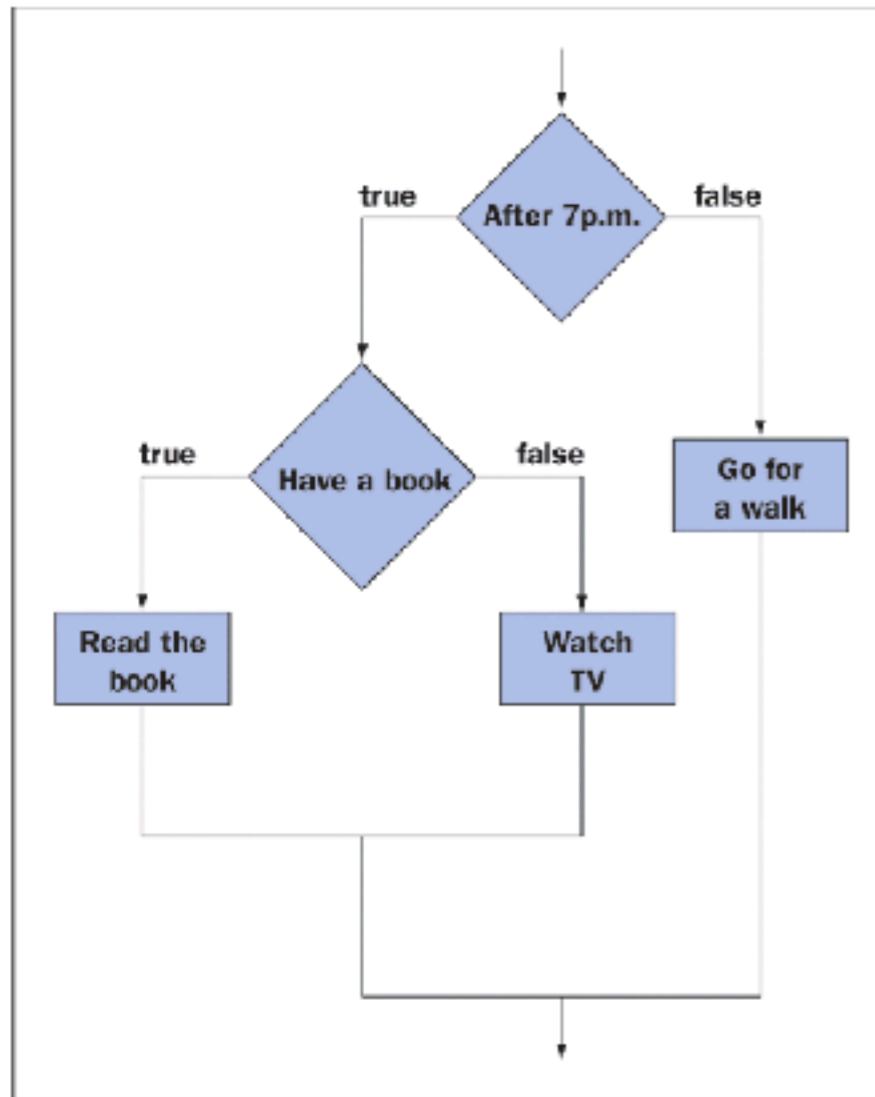
Example #4 (nesting with braces)

```
if (average>=90)
    System.out.println("A");
else {
    if (average>=80)
        System.out.println("B");
    else {
        if (average>=70)
            System.out.println("C");
        else {
            if (average>=60)
                System.out.println("D");
            else {
                System.out.println("F");
            }
        }
    }
}
```

Example #5 (nesting without braces - more compact statement)

```
if (average>=90)
    System.out.println("A");
else if (average>=80)
    System.out.println("B");
else if (average>=70)
    System.out.println("C");
else if (average>=60)
    System.out.println("D");
else
    System.out.println("F");
```

Example #6 - Can you write an if statement for the following?



AFTER 7 P.M.	HAVE A BOOK	ACTION TAKEN
true	true	read book
true	false	watch TV
false	true	walk
false	false	walk

Can it be written with logical operators or must it be nested?

Nested if statement

```
if (the time is after 7 PM) {  
    if (you have a book)  
        read the book;  
    else  
        watch TV;  
}else  
    go for a walk;
```

Section 6.4 - Logical Errors in Nested if statements

Example #1 (original)

```
if (the weather is wet) {  
    if (you have an umbrella)  
        walk;  
    else  
        run;  
}
```

Example #2

```
if (the weather is wet) {  
    if (you have an umbrella)  
        walk;  
}else  
    run;
```

Do these examples show the same logic?

THE WEATHER IS WET	YOU HAVE AN UMBRELLA	VERSION 1 OUTCOME	VERSION 2 OUTCOME
true	true	walk	walk
true	false	run	none
false	true	none	run
false	false	none	run

NO!

What happens if all braces are removed?

Example #3

```
if (the weather is wet)
  if (you have an umbrella)
    walk;
else
  run;
```

Example #4

```
if (the weather is wet)
  if (you have an umbrella)
    walk;
else
  run;
```

Where does the else go? With the first if or the second?

In the absence of braces the else will go with the nearest preceding if.

Indentation does not affect the outcome.

What happens in this example?

Example #5

```
if (the weather is wet)
  if (you have an umbrella)
    open umbrella;
    walk;
  else
    run;
```

There is a compile error. You must use braces to correct.

Example #6

```
if (the weather is wet) {
  if (you have an umbrella) {
    open umbrella;
    walk;
  } else {
    run;
  }
}
```

Sometimes logical errors occur in nested if statements.

Example #7

- You make a "B" if you get at least 80%.
- You make a "A" if you get at least 90%.

```
if (average>=80)
    grade = "B";
else if (average>=90)
    grade = "A";
```

This logic error can be corrected with a nested if statement. Sometimes getting rid of nested ifs as a sequence of independent if statements can also correct the problem.

Example #8

```
if (average>=90)
    grade ="A";
else if (average>=80)
    grade = "B";
```

Example #9

```
if (average>=90)
    grade ="A";

if (average>=80 && average<90)
    grade = "B";
```

Section 6.5 - Nested Loops

Def.

Nested Loop is a programming situation where a loop is placed inside another loop.

Example #1 - Multiplication Table (for statements)

```
for (int x=1; x<=10; x++){
    for (int y=1; y<=10; y++){
        System.out.print(x*y + " ");
    }
    System.out.println();
}
```

Example #2 - Multiplication Table (while statements)

```
int x=1;
while (x<=10){
    int y=1;
    while (y<=10){
        System.out.print(x*y + " ");
        y++;
    }
    System.out.println();
    x++;
}
```

Example #3 - Determine if a Number is Prime

```
Scanner reader = new Scanner(System.in);
System.out.print("Enter a positive integer: ");
int n = reader.nextInt();

boolean prime = true;

for (int x=2; x<=Math.sqrt(n); x++){
    if (n%x==0) {
        prime=false;
        break;
    }
}

if (prime)
    System.out.print(n + " is a prime number.");
else
    System.out.print(n + " is a not a prime number.");
```

Example #4 - Finding All Prime Numbers Between 1 and 1000

```
for (int n=2; n<=1000; n++){  
  
    boolean prime = true;  
  
    for (int x=2; x<=Math.sqrt(n); x++){  
        if (n%x==0) {  
            prime=false;  
            break;  
        }  
    }  
  
    if (prime)  
        System.out.print(n + " is a prime number.");  
}
```

Section 6.6 - Testing Loops

When designing test data for loops, we want to cover all the possibilities.

Read pages 224-225.

```
//Display the proper divisors of a number
System.out.print("Enter a positive integer: ");
int n = reader.nextInt();

int limit = n/2;

for (int d=2; d<=limit; d++){
    if (n%d == 0)
        System.out.print(d + " ");
}
```

Test data for this loop:

TYPE OF TEST	DATA USED
No iterations	0, 1, 2, and 3
One iteration	4 and 5
Multiple iterations for a number with divisors	24
Multiple iterations for a number without divisors	29

Case Study 2: Fibonacci Numbers

Read pages 226-228

```
import java.util.Scanner;
public class Fibonacci {
    public static void main (String [] args) {
        Scanner reader = new Scanner(System.in);
        int n;
        int fib;
        int a,b,count;
        while (true){
            System.out.print("Enter a positive integer or -1 to quit: ");
            n = reader.nextInt();
            if (n == -1) break;
            else if (n >= 1){
                fib = 1;
                a = 1;
                b = 1;
                count = 3;
                while (count <= n){
                    fib = a + b;
                    a = b;
                    b = fib;
                    count = count + 1;
                }
                System.out.println ("Fibonacci of " + n + " is " + fib);
            }
        }
    }
}
```

