Selection
Control Structures

Chapter 5: Selection
Asserting Java

Chapter 5: Outline

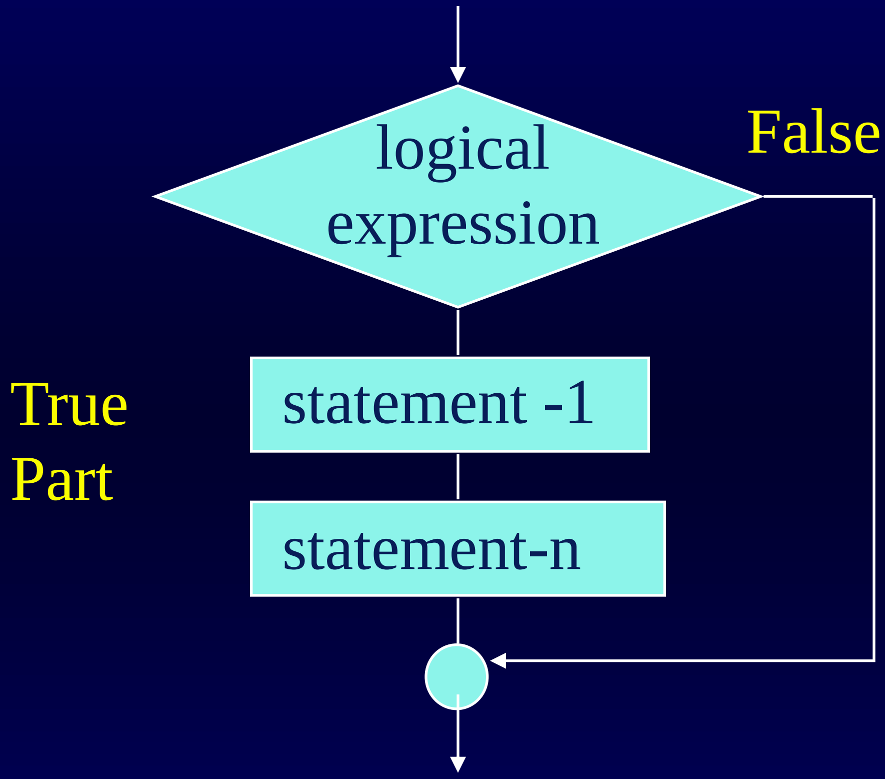
- This chapter presents algorithmic patterns that allow alternatives to straight sequential processing:
  - Guarded Action
    - execute an action only under certain conditions
  - Alternative Action
    - choose one action or another
  - Multiple Selection
    - choose from more than two sets of actions

The Guarded Action Pattern

Pattern:	Guarded Action
Problem:	Execute an action only under certain conditions
General Form	if (*true-or-false-condition* is true) execute this set of statements

Code Example:	`if(aStudent.getGPA() >= 3.5)`
	`    deansList.add(aStudent);`

Flowchart view of guarded action



– After the boolean expression of the if statement evaluates, the true-part executes only when the boolean expression is true.

The if statement (general form)

- General form:

```
    if (boolean-expression)
       true-part ;
```

- A *boolean-expression* is any expression that evaluates to true or false *three examples*

```
   sales < 15000.00
   hoursWorked > 40
   true || false  // read as true or false
```

Relational Operators

- boolean expressions often use these relational operators:

`>`	Greater than
`<`	Less than
`>=`	Greater than or equal
`<=`	Less than or equal
`==`	Equal
`!=`	Not equal

boolean expressions

- Answer true, or false

```
double n1 = 78.0;
double n2 = 80.0;
n1 < n2                          // _____
n1 >= n2                         // _____
(n1 + 35) > n2                   // _____
Math.abs(n1-n2) <= 0.001         // _____
n1 == n2                         // _____
n1 != n2                         // _____
```

Examples

```
if(grade >= 60)
  return "Passing";


if(name.indexOf(",") == -1)
  return "Missing comma";


if(str.length() < 2)
  return str;
```

Boolean Operators

- A logical operator (&& means AND) used in an if...else statement:

```
if((test >= 0) && (test <= 100))
   System.out.println("Test in range");
```

- The code evaluates an expression to see if test is in the range of 0 through 100 inclusive.

Truth Tables for Boolean Operators

- Truth tables for the Logical (Boolean) operators
  `!, ||, &&`

! (not)		|| (or)		&& (and)	
Expression	Result	Expression	Result	Expression	Result
`! false`	`true`	`true || true`	`true`	`true && true`	`true`
`! true`	`false`	`true || false`	`true`	`true && false`	`false`
		`false || true`	`true`	`false && true`	`false`
		`false || false`	`false`	`false && false`	`false`

The Alternative Action Pattern

- Situations arise that require a program to select between one set of actions or another
- Examples
  - withdraw or deposit money
  - pass or fail the entrance requirements
- This is the Alternative Action Pattern
  - choose between two alternate sets of actions

Alternative Action

Pattern:	Alternative Action
Problem:	Must choose one action from two alternatives

Outline:	if (*true-or-false-condition* is true)
	    execute action-1
	else
	    execute action-2

Code Example:	`if(finalGrade >= 60.0)`
	`   System.out.println("passing");`
	`else`
	`   System.out.println("failing");`

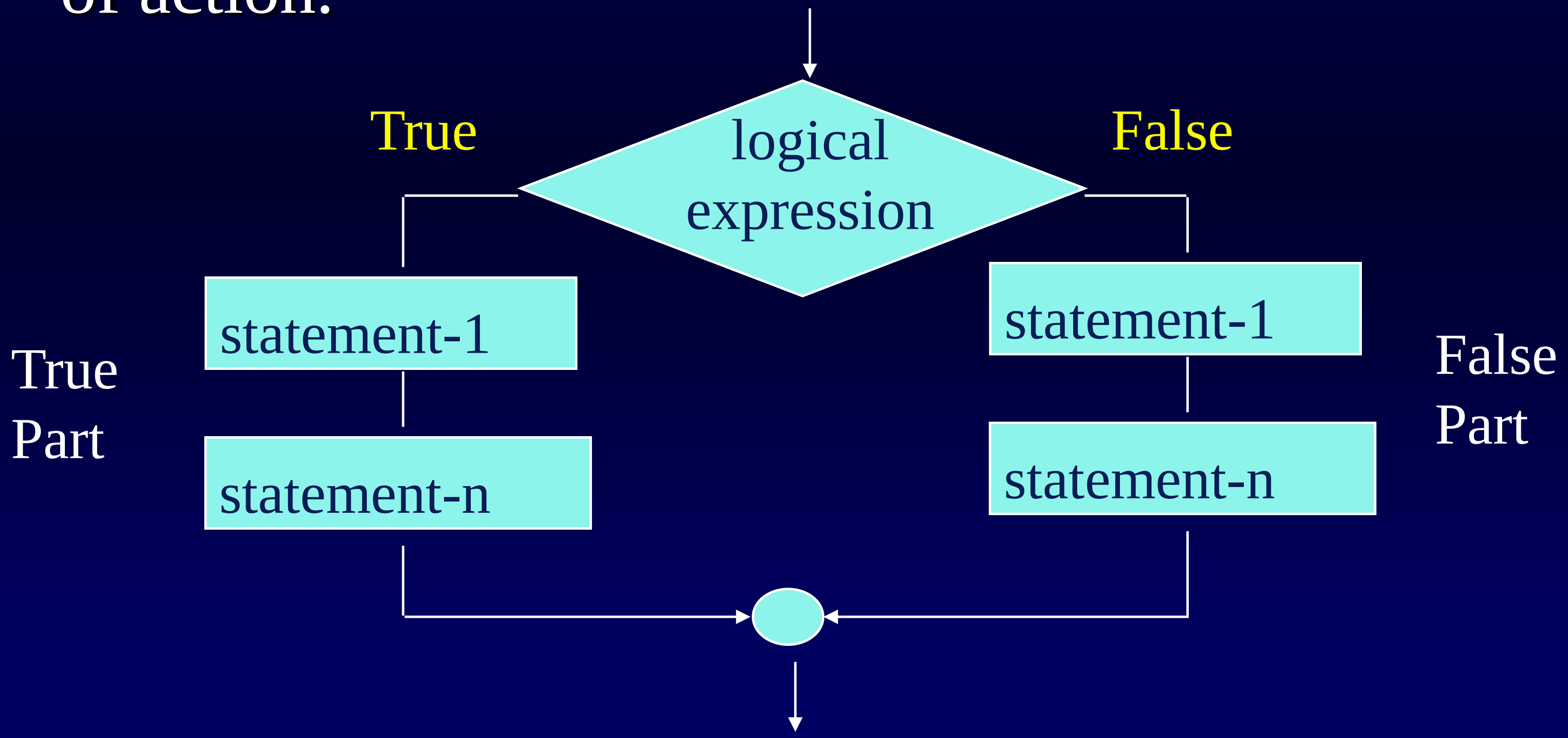
if-else General Form

```
if  ( boolean-expression )
    true-part ;
else
    false-part ;
```

– When the boolean expression evaluates to true, the true-part executes and the false-part is disregarded. When the boolean expression is false, only the false-part executes.

The if...else statement

- The if...else statement allows two alternate courses of action.



if...else Example

- Write the output below

```
if(sales >= 15000.00)
  System.out.println("Bonus="+((sales-15000.0)*0.05));
else
  System.out.println((15000.0-sales) + " short");
```

sales	Output
16000.00	______________________
2000.00	______________________
15000.00	______________________

More Precedence Rules

- The following slide summarizes all operators used in this textbook (we've seen 'em all now)
  - Precedence: most operators are evaluated (grouped) in a left-to-right order:

`a / b / c / d` is equivalent to `(((a/b)/c)/d)`

  - Assignment operators group in a right-to-left order so the expression

`x = y = z = 0.0` is equivalent to `(x=(y=(z=0.0)))`

Operators so far *chapter 7* += -= ++ --

Rank	Operator	Description	Grouping
1	`.`	field reference (field or method)	left to right
	`( )`	function call (message send)	
2	`! + -`	not, unary plus, unary minus	right to left
3	`new`	construct objects	
4	`*`	multiplication	left to right
	`/`	division	
	`%`	remainder	
5	`+ -`	addition (for int and double), subtraction	left to right
7	`< <=`	less than, less than or equal	left to right
	`> >=`	greater than, greater than or equal to	
8	`== !=`	equal not equal	left to right
12	`&&`	**`boolean`** and	left to right
13	`||`	**`boolean`** or	left to right
15	`=`	assignment	right to left

Short Circuit Boolean Evaluation

- Java boolean expressions evaluate subexpressions in a left to right order
- Sometimes the evaluation could stop early
  - This expression never evaluates the sqrt of a negative number (it only evaluates what is necessary) :

```
if((x >= 0.0) && (Math.sqrt(x) <= 2.5))
 // ...
```

  - and `test>100` is not evaluated when `test<0` is true

```
if(test < 0 || test > 100)
 // ...
```

Multiple Selection

- Nested logic:
  - one control structure contains another similar control structure.
    - an if...else inside another if...else.
    - allows selections from 3 or more alternatives
- We must often select one alternative from many

Pattern: Multiple Selection

Problem: Must execute one set of actions from three or more alternatives.

Outline:

```
if ( condition 1  is true)
    execute action 1
else  if( condition 2 is true)
   execute action 2
 // ...
else  if( condition n-1 is true)
   execute action n-1
else
   execute action n
```

Code Example:

```
if(grade < 60)
  result = "F";
else if(grade < 70)
  result = "D";
else if(grade < 80)
  result = "C";
else if(grade < 90)
  result = "B";
else
  result = "A";
```

Example of Multiple Selection

nested if...else

```
if(GPA < 3.5)
   System.out.println("Try harder");
else
   if(GPA < 4.0)
      System.out.println("Dean's List");
   else
      System.out.println("President's list");
```

The false part is another if...else

GPA	***Output:***
3.0	______________
3.6	______________
4.0	______________

A Greeting method

- According to the hour of day in European time, 0-23, Complete method `greeting` to return "Good Morning" (0..11), "Good Afternoon" 12..16), "Good Evening (17..19) and "Good Night (20..23)"
- Complete a test method for `String greeting(int)` in ControlFunTest
  - cover all branches (that will be 4)
  - cover all boundaries, which would be the lower bound of each branch (or the upper bound of each)
- Complete `String greeting(int)` in ControlFun

Multiple Returns

- It is possible to have multiple return statements in a method

terminate when the first return executes, BUT return something

```
public String letterGrade()   {
  if(my_percentage >= 90.0)
    return "A";
  if(my_ percentage >= 80.0)
    return "B";
  if(my_ percentage >= 70.0)
    return "C";
  if(my_ percentage >= 60.0)
    return "D";
  if(my_ percentage >= 0.0)
    return "F"; // OOPS! WRONG when percentage < 0
}
```

Error! You must return something

To fix, remove

`if(percentage >= 0.0)`