Lecture 19:
Selection, Part I

Sierra College
CSCI-12
Spring 2015
Weds 04/08/15

Announcements

- **General**
  - Midterms were returned on Monday, and all scores are posted in Canvas
  - See me if you didn't get yours back, or if you have any grading/scoring concerns
- **Schedule**
  - Spring withdraw deadline is Thurs 4/16
    - Final off-ramp: after that point, you will receive a letter grade for this class
    - Please check your grades in Canvas, and assess where you stand ("gut check")
      - Let's talk if any concerns…
- **New assignments**
  - PRGM19: Age Utils (due Sunday 04/19 @ 11pm) lab time this wk AND next wk
    - 2 Java classes: **Utils** and **AgeClient** (some starter code is provided)
    - **Utils**: beginnings of a static utilities file, get an int from Scanner ***or*** JOptionPane, and calculate an accurate age (we will keep adding to this file over next programs)
    - **AgeClient**: prompt user for BD data, call age method(s) 3x, and display correct ages

Lecture Topics

- **Last time**:
  - Conditions: equality, relational, and logical operators
- **Today**:
  - Finish up conditional operators
  - Selection: various forms of ***if*** logic

For Next Time

- **Lecture Prep**
  - Text readings and lecture notes
- **Lab**
  - Start making headway on the next assignment
  - At a minimum, get starter versions of both classes in place
  - Suggestion: write your client class first, using the *existing* version of the starter UtilsFL class

Selection Recap

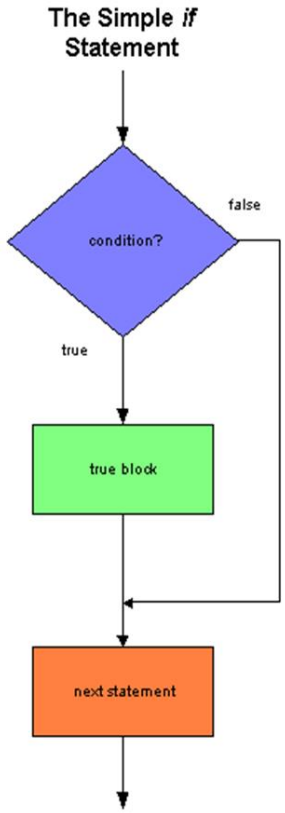
- In the previous lecture, we began talking about **selection**
  - Selection ← → "decision making"
  - 3rd of the 4 basic **flows of control** (Ch.5)
    - Others: sequential execution, method calls, looping (Ch.6)
- Two basic components to selection
  - **Conditions**
    - "Forks in the road" of execution paths
    - Allow us to express decision-making programatically
    - Formed from 3 types of operators:
      - 2 **equality operators**: **==, !=**
      - 4 **relational operators**: **> , >= , < , <=**
      - 3 **logical operators:** **&& , || , ! (AND, OR, NOT)**
  - **Selection structures**
    - The logic structures which provide the code framework to implement the desired selection logic
    - Topic of today's lecture

Selection Structures

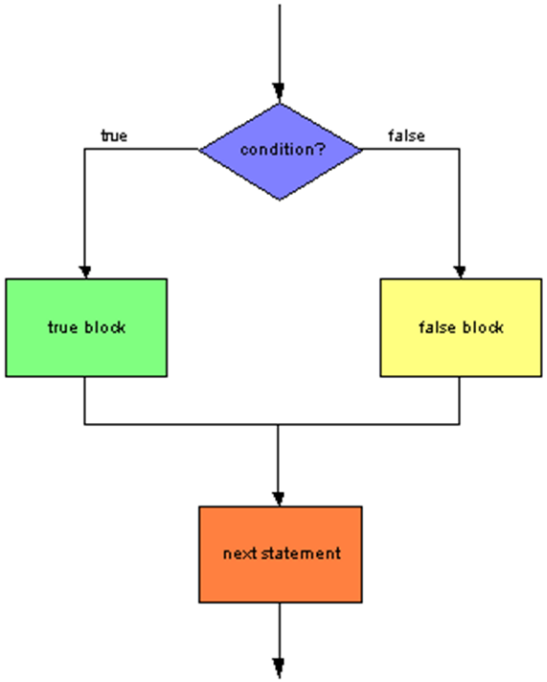
- **Selection structures** are standard decision-making frameworks in code
  - Certain statement branches within them are only executed if their associated conditions are **true**
  - If their associated conditions are false, those statement are ignored and not executed
- The execution path thru selection structures is often expressed visually using **flowcharts**
- There are several selection structures, each useful in different program logic situations:
  - ***if***
  - ***if-else***
  - ***if-else if***
  - the **conditional operator**
  - ***switch***

if Selection Structures Compared

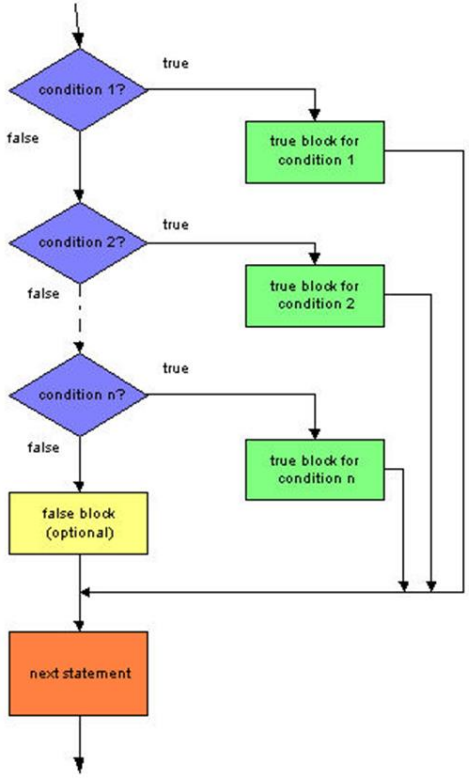
if structure



if-else structure



if-else if structure



if Selection Structures Compared

- Use the ***if*** structure for:
  - Executing statements under one condition only, but not otherwise
  - "*do this, but only if…*"
- Use ***if-else*** structure for:
  - Executing statements for either one of two mutually exclusive conditions
  - "*do this or that*"
- Use ***if-else if*** structure for:
  - Executing statements under multiple, mutually exclusive conditions
  - "*do this or that or that or…*"

The Simple *if* Structure

- This is the simplest selection form
- Used when a program should execute statement(s) for a certain condition, but not at all otherwise

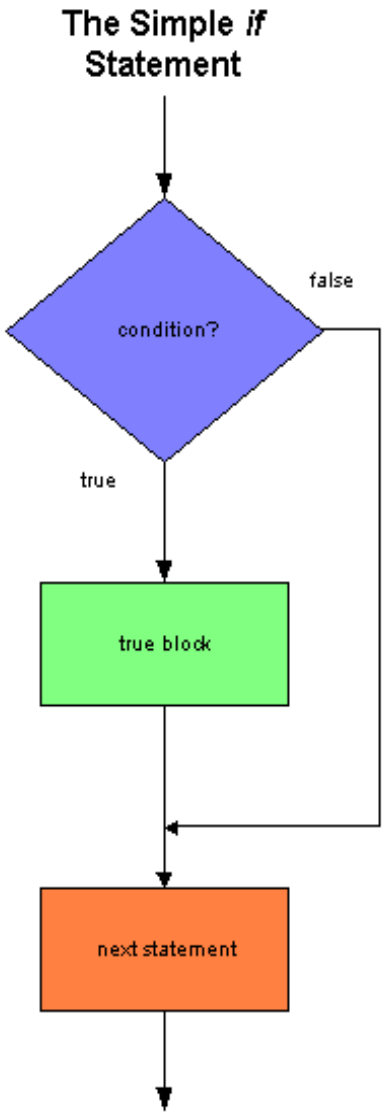
- Notes:
  - The braces {…} are **optional** if the block contains only one statement, **required** otherwise
  - Indentation aids readability
  - Easier to add more statements later
  - Consider the braces and indentation as the REQUIRED coding style for this course

- General form:

```
if ( condition ) {
   // true block statement(s)
   // executed only if condition = true
}

// next statement(s)
// always executed, whether condition T/F
```

Simple *if* Example




```
13 import java.util.Scanner;
14 import java.text.DecimalFormat;
15
16 public class SelectionIf {
17
18     public static void main(String [] args) {
19
20         // object declarations
21         Scanner input = new Scanner(System.in);
22         DecimalFormat fmt = new DecimalFormat("$#0.00");
23
24         // data declarations
25         int age;
26         double ticketPrice = 10.00;
27
28         // prompt for inputs
29         System.out.print("Enter age: ");
30         age = input.nextInt();
31
32         // check for 25% senior discount
33         if (age >= 65) {
34             ticketPrice *= 0.75;
35         }
36
37         // following statement(s) are always done
38         System.out.println("Please pay: " +
39                            fmt.format(ticketPrice));
40
41     } // end main
42
43 } // end class
```

See ***SelectionIf.java*** in **Example Source Code**

Some *if* Finer Points

- Some things to demonstrate (or try yourself):
  - Do NOT place a semicolon after the condition statement
    - This results in a condition with an EMPTY *if*-block
    - It gives no compiler error
    - But it can give unintended logic errors (statements always executed)
  - If there are no braces around the if block, any statements past the first one will ALWAYS get executed
    - Can give unintended logic errors
    - For this course, ALWAYS use braces after a condition, even if not required for a single-line statement
  - The debugger is your friend!
    - Use it to trace the actual path of execution thru your code
    - Otherwise, you must rely on print() statements to debug your code

The *if-else* Structure

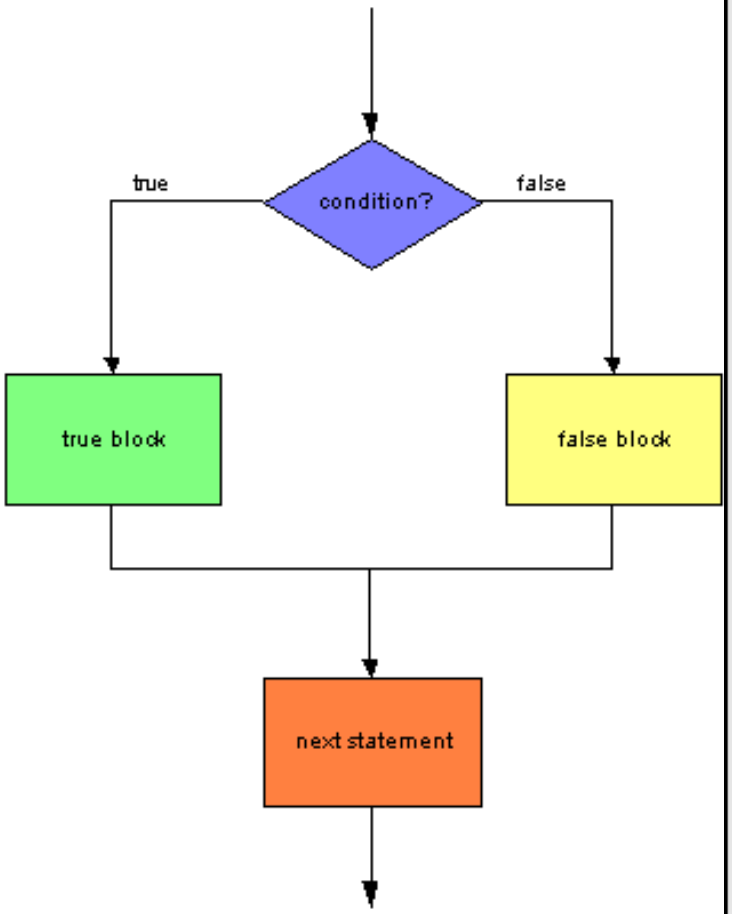
- Next simplest selection form
- Used when a program should execute **one of two mutually exclusive** sets of statements
- Notes:
  - The braces {…} are **optional** if either block contains only one statement, **required** otherwise
  - Indentation aids readability
  - Easier to add more statements later
  - Consider the braces and indentation as the REQUIRED coding style for this course

- General form:

```
if (condition) {
   // true block
   // executed only if condition = true
}
else {
   // false block
   // executed only if condition = false
}

// next statement(s)
// always executed, whether condition T/F
```

if-else Example




```
14 public class SelectionIfElse {
15
16     public static void main(String [] args) {
17
18         // data initializations
19         int score;
20         boolean status;
21         final int PASSING_SCORE = 70;
22
23         String passMessage = "You passed, you da man!";
24         String failMessage = "You didn't pass, keep trying";
25         String message = new String();
26
27         // read input using utility method statically
28         // encapsulates details of setting up Scanner/JOptionPane
29         score = UtilsFL.readInt("Enter test score: ");
30
31         // decide outcome and set flag
32         if (score >= PASSING_SCORE) {
33             status = true;
34         }
35         else {
36             status = false;
37         }
38
39         // a boolean flag by itself can represent a condition
40         // a bit contrived, but just to illustrate a point
41         if (status) {
42             message = passMessage;
43         }
44         else {
45             message = failMessage;
46         }
47         System.out.println(message);
48
49     } // end main
50
51 } // end class
```

See ***SelectionIfElse.java*** in **Example Source Code**

The *if-else if* Structure

- Most general selection form
  - Represents a "stacked", or cascaded, set of *if-else* constructs
- Used when a program should execute **one of several mutually exclusive** sets of statements
- Note:
  - The final *else* block is **optional**
  - The braces {…} are **optional** if either block contains only one statement, **required** otherwise
  - Indentation aids readability
  - Easier to add more statements later
  - Consider the braces and indentation as the REQUIRED coding style for this course

- General form:

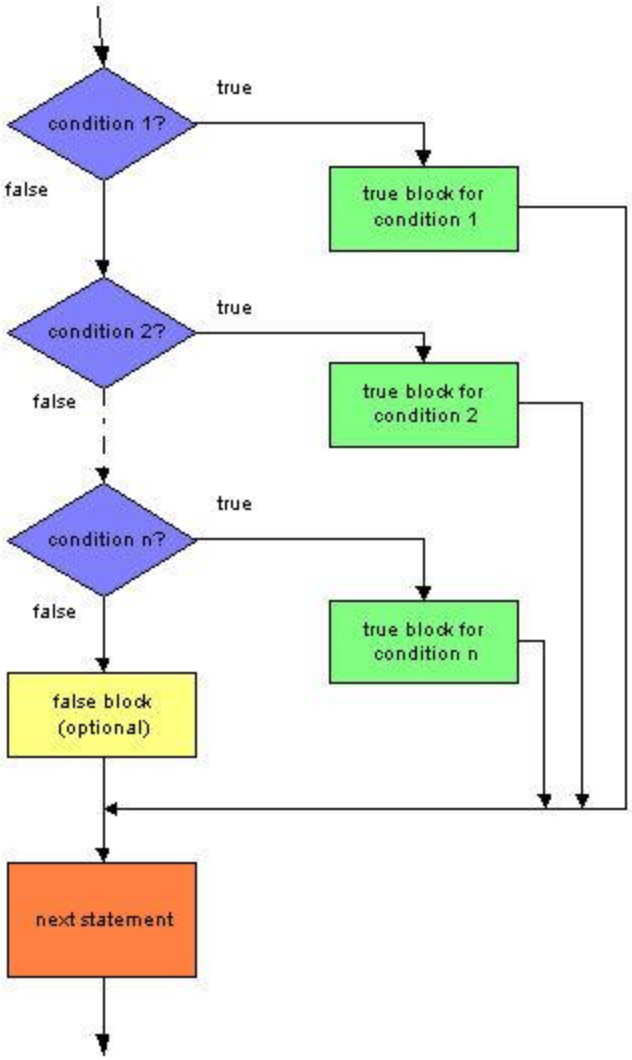
```
if (condition 1) {
   // true block for condition 1
}
else if (condition 2) {
   // true block for condition 2
}

 . . .

else {
   // false block for all other conditions
   // (none of the other conditions are met)
}

// next statement(s), always executed
```

if-else if Example




```
14 public class SelectionIfElseIf {
15
16     public static void main(String [] args) {
17
18         String message = new String();
19         int age;
20
21         // prompt user for age
22         age = UtilsEL.readInt("Enter your age: ");
23
24         // determine age-specific category
25         // note that ages are mutually exclusive,
26         //  earlier ones weed out ages for later ones
27         if (age < 13) {
28             message = "just a kid!";
29         }
30         else if (age < 18) {
31             message = "uh-oh, a teenager";
32         }
33         else if (age < 30) {
34             message = "time for college";
35         }
36         else if (age < 50) {
37             message = "time for a mortgage";
38         }
39         else if (age < 70) {
40             message = "how's the 401(K)?";
41         }
42         else {
43             message = "the golden years";
44         }
45
46         // print demographic message
47         System.out.println(message);
48
49     } // end main
50
51 } // end class
```

See ***SelectionIfElseIf.java*** in **Example Source Code**

Some *if-else if* Points

- Things to note in the previous example:
  - It is not necessary to test the age at both endpoints, using a compound condition
    - Prior *if* blocks already eliminate these possibilities
  - ALL *if* blocks are checked in order, until a matching case is found (or not found)
    - Easiest to see using the debugger

```
14 public class SelectionIfElseIf {
15
16     public static void main(String [] args) {
17
18         String message = new String();
19         int age;
20
21         // prompt user for age
22         age = UtilsEL.readInt("Enter your age: ");
23
24         // determine age-specific category
25         // note that ages are mutually exclusive,
26         //  earlier ones weed out ages for later ones
27         if (age < 13) {
28             message = "just a kid!";
29         }
30         else if (age < 18) {
31             message = "uh-oh, a teenager";
32         }
33         else if (age < 30) {
34             message = "time for college";
35         }
36         else if (age < 50) {
37             message = "time for a mortgage";
38         }
39         else if (age < 70) {
40             message = "how's the 401(K)?";
41         }
42         else {
43             message = "the golden years";
44         }
45
46         // print demographic message
47         System.out.println(message);
48
49     } // end main
50
51 } // end class
```

See ***SelectionIfElseIf.java*** in **Example Source Code**

The Conditional Operator

- The **conditional operator (?:)** is a shortcut operator which simplifies some specific *if-else* logic
  - It is not a full statement by itself
  - But, it can be used as part of larger expressions, or for variable assignment
  - Java's only ternary (3 operand) operator
- Syntax:

  (condition ? trueExpression : falseExpression)
- Operation:
  - Evaluate the condition
  - If condition == *true*, use *trueExpression* for the expression value
  - If condition == *false*, use *falseExpression* for the expression value

Conditional Operator Equivalence

- Both versions of code at right are functionally identical
  - Some prefer the compact style of the conditional operator
  - Others prefer the if-else format as more readable
- Some typical uses for the conditional operator:
  - Handling invalid input
  - Output of similar messages
  - Two possible variable values
- Conditional operator precedence is very low
  - See next slide, or Appendix B

```
// if-else logic:
if (condition) {
   var  =  trueExpr;
}
else {
   var = falseExpr;
}

// conditional operator:
var = (condition ? trueExpr : falseExpr);
```

Conditional Operator Precedence

Operator Precedence

Operators	Precedence
postfix	`expr++ expr--`
unary	`++expr --expr +expr -expr ~ !`
multiplicative	`* / %`
additive	`+ -`
shift	`<< >> >>>`
relational	`< > <= >= instanceof`
equality	`== !=`
bitwise AND	`&`
bitwise exclusive OR	`^`
bitwise inclusive OR	`|`
logical AND	`&&`
logical OR	`||`
ternary	`? :`
assignment	`= += -= *= /= %= &= ^= |= <<= >>= >>>=`

Conditional Operator Examples

- See the following examples:
  - **Example Source Code**, ***SelectionConditional1.java***
    - Grade calculation message, done 2 ways (if-else, conditional)
  - **Example Source Code**, ***SelectionConditional2.java***
    - Absolute value, done 3 ways (if-else, conditional. Math.abs())

Variable Block Scope

- **Block scope** refers to:
  - The extent within a program where some variable can be **referenced**
  - Where it can be “seen”, or used, by other code
- Some block examples:
  - Selection structure *true* and *false* blocks
  - Within methods
  - Within classes
- The **scope** of a program variable extends
  - From the point at which it is declared
  - To the end of the block in which it was declared
- A variable is only **valid** (**visible**) within the braces of the innermost block in which it was declared
  - **To access a variable outside a block, it must also be declared outside that block**
  - **Declaring all variables near beginning of programs helps avoid this problem in the first place!**

Block Scope Example

- The scope of *letterGrade* is line 27 only, the *true* block.
- The compiler will generate an error message for lines 30 and 32 (“error: cannot find symbol”)
- To fix this problem, declare letterGrade as shown in line 20
  - As should be done per good practice anyway!

```
13 public class SelectionScope {
14
15     public static void main(String [] args) {
16
17         // declarations
18         final int PASS_MINIMUM = 60;
19         int grade;
20         //char letterGrade;
21
22         // obtain input from user
23         grade = UtilsFL.readInt("Enter numerical test score: ");
24
25         // display outcome
26         if (grade >= PASS_MINIMUM) {
27             char letterGrade = 'P';
28         }
29         else {
30             letterGrade = 'F';
31         }
32         System.out.println("Your grade is: " + letterGrade);
33
34     } // end main
35
36 } // end class
37
```

See ***SelectionScope.java*** in **Source Code Examples**