Object-Oriented Programming I

Advanced Branching: if and switch

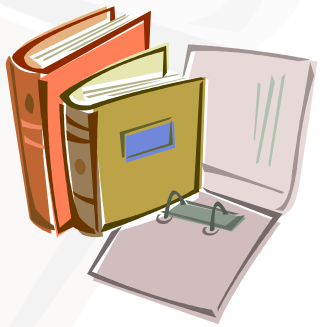
Slides by Magdin Stoica
Updates by Georg Feil

Learning Outcomes

1. Define more advanced control structures using nested and alternative if-else statements
2. Define control structures using the switch statement
3. Explain how switch statements control the flow of the program
4. Create programs that using flow control structures to achieve different outcomes in the program based on program conditions
5. Understand how to choose between ‘switch’ and ‘if-else’

Reading Assignment

- Introduction to Java Programming (required)
  - Chapter 3: Selections, section 3.14



Nested ‘if’ statements

- Putting an ‘if’ statement in the body of another ‘if’ statement is called nesting
  - Nested ‘if’ statements may appear either in the “if” block or the “else” block of another ‘if’ statement
  - There is no practical limit to the number of levels of nesting, but try to keep your program readable!

Nested ‘if’ statements (pseudocode)

```
if (<condition expression 1>)
{
        …
        if (<condition expression 2>)
        {
                <positive outcome statement 1 && 2>;
        }
        …
}
else
{…}
```

Nested ‘if’ statements (pseudocode)

```
if (<condition expression 1>)
{…}
else
{
        …
        if (<condition expression 2>)
        {
                <outcome statement !1 && 2>;
        }
        …
}
```

Nested ‘if’ statements (pseudocode)

```
if (<condition expression 1>)
{
        if (<condition expression 2>)
        {
                <positive outcome statement 1 && 2;
        }
}
else
{
        if (<condition expression 3>)
        {
                <positive outcome statement !1 && 3>;
        }
}
```

Example Prog 1: Nested ‘if’ statement

```
...

    Scanner input = new Scanner(System.in);   // Scanner to ask for input
    String line1 = input.nextLine();
    String line2 = input.nextLine();

    if (line1.equals("AA")) {
        if (line2.equals("BB")) {
           System.out.println("Line 1 is AA and line 2 is BB");
        }
    }
    else {
        if (line2.equals("BB")) {
           System.out.println("Line 1 is not AA and line 2 is BB");
        }
    }
    System.out.println("This runs no matter what");

...
```

Alternative ‘if’ statements

- Put an ‘if’ statement immediately following the ‘else’ keyword of another ‘if’ statement to test more than one alternative or condition
  - Note this is not considered nesting

- An ‘else’ block at the very end will run if none of the ‘if’ conditions was true

Alternative ‘if’ Statements (pseudocode)

```
if (<condition expression 1>) {
  <outcome statement 1;
}
else if (<condition expression 2>) {
  <outcome statement !1 && 2>;
}
else if (<condition expression 3>) {
  <outcome statement !1 && !2 && 3>;
}
else {
  <outcome statement !1 && !2 && !3>;
}
```

Exercise 1: Alternatives (else-if)

- Write a program that inputs a student grade as a percentage (0 – 100) and converts the numeric grade into a letter grade
  - Use the Sheridan grade conversion rules
- Hint: Use nested if statements to combine conditions, or use alternative conditions (else-if)
- How would you write the program that does the opposite conversion (from letter grade to number)?

Example Prog 2: ‘if’ statement with ‘else-if’

```
...
    Scanner input = new Scanner(System.in); // For user input
    String line = input.nextLine();

    if (line.equals("AA")) {
       System.out.println("Line is AA");
    } else if (line.equals("BB")) {
       System.out.println("Line is BB");
    } else if (line.equals("CC")) {
       System.out.println("Line is CC");
    } else {
       System.out.println("Line is NOT AA, BB or CC");
    }
    System.out.println("This runs no matter what");
...
```

‘if’ examples from Head First Java

- See page 13: http://proquestcombo.safaribooksonline.com.library.sheridanc.on.ca/book/programming/java/0596009208/dive-in-a-quick-dip-breaking-the-surface/13

When there is more choice

- Sometimes we have to choose between more than true and false
  - Work schedules may be different depending on the day of the week
    - (i.e. Monday we work 9 to 5, Saturday we work 11 to 3, Sunday we don't work)
  - Types of accounts may have different business rules associated with them
    - Chequing Account, Savings Accounts, Tax-free Accounts, RRSP accounts
  - Status information can be represented better if not restricted to only two values (true / false)
    - Marital Status: Married, NotMarried, CommonLaw, Divorced, Widowed
    - Work Status: FullTime, Contract, Occasional, Unemployed, Retired
    - Student Status: FullTime, PartTime, ConEd
    - Month: Jan, Feb, March…
    - Colour: Red, Blue, Green
- What if the program needs to perform different actions depending on each choice?

A big fork in the road

…

Day of the week (1-7)

1: “Monday”

2: “Tuesday”

3: “Wednesday”

4: “Thursday”

5: “Friday”

Any other day…

…

Numeric expression

The switch Control Structure

- The switch keyword introduces the condition
  - Must be numerical expression that evaluates to a whole number
  - Some languages allow string expressions that evaluate to a string (Java allows that starting with Java 7)
- The case keyword introduces each case and its statements
- The default keyword is the ultimate “else”, when no case applies.

The switch Control Structure (pseudocode)

```
switch(<expression>) {

    case <value-1>:
            <statement 1.1>;
            <statement 1.2>;
            …
            break;

    case <value-2>:
            <statement 2.1>;
            <statement 2.2>;
            …
            break;

    … // as many cases as you need

    default:
            <default statement 1>;
            <default statement 2>;
            …
            break;
}
```

Keyword to introduce the numerical / text expression

Keyword to introduce each case, each possible value and its outcomes

Keyword to define what happens if expression value does not match any case

Keyword to end the case

Notice the colon

‘switch’ requires
a numeric expression,
or String in Java 7

Example: switch by month

```
int month = …;
switch (month) {

        case 1:
                System.out.println("January");
                break;

        case 2:
                System.out.println("February");
                break;

        case 3:
                System.out.println("March");
                break;

        …

        case 12:
                System.out.println("December");
                break;

        default:
                System.out.println("Invalid month number");
                break;
}
```

Example: switch with Strings (Java 7)

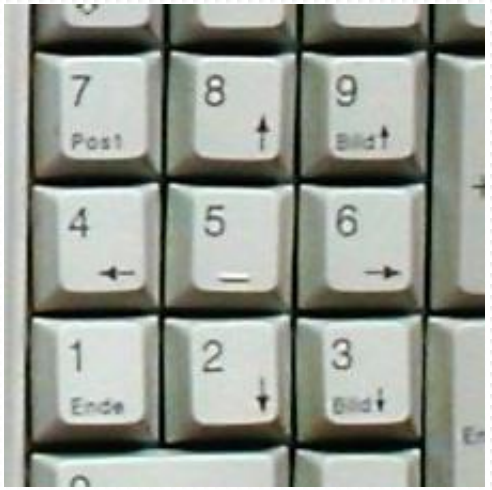
```
String month = …;
switch (month) {

        case "January":
                System.out.println("January has 31 days");
                break;

        case "February":
                System.out.println("February has 28 or 29 days");
                break;

        case "March":
                System.out.println("March has 31 days");
                break;

        ...

        case "December":
                System.out.println("December has 31 days");
                break;

        default:
                System.out.println("Invalid month name");
                break;
}
```

How to choose between if-else and switch

- Use a ‘switch’ statement in the following situations:
  - You are testing for specific numbers or characters, for example letters of the alphabet
  - You are testing for specific strings
  - There are many values to test (more than 3)
- Use if-else in the following situations:
  - You are testing conditions that involve >, <, or !=
  - The same action is needed for a range of values, for example all integers between 100 and 200
  - You are testing data types which can’t be used in switch
    - e.g. double, float, long, any class type except String
  - There are only a few values to test (3 or fewer)

Exercise 2: Alternatives (switch)

- Write a Java program that handles keyboard input similar to a game:
  - 2 – print “move down”
  - 4 – print “move left”
  - 6 – print “move right”
  - 8 – print “move up”
  - 1, 3, 7, 9 – print “diagonal move not allowed”
  - Any other number – print “bad input”
- Your program should use the ‘switch’ statement



Wikimedia Commons

Fall-Through Cases

- Fall-through cases are cases that are not terminated by a “break” statement
- The statements execute starting with a matching case and continue until the first “break” statement is encountered
- If a “case” block doesn’t contain a break, the program flow will continue with the statements of the next case in the sequence
  - Watch out, don’t do this by accident!

Example: Fall-through by Quarter

```
int month = …;
switch (month)
{
        case 1:
        case 2:
        case 3:
                System.out.println("First Quarter");
                break;

        case 4:
        case 5:
        case 6:
                System.out.println("Second Quarter");
                break;

        …

        default:
                System.out.println("Invalid month number");
                break;
}
```

Exercise 3: Fall-through

- Write a class called VowelFinder that inputs a character from the user
- The program should determine whether the character is a vowel or not, then print a message
- Hints
  - It's hard to input a 'char' using Scanner so you should input the character as a string instead
  - Your program should work for upper & lower case

Common switch Errors

- Using a Boolean expression instead of a numerical (or String in Java 7) expression
  - Causes a syntax error
- Forgetting a “break” for a case
  - Causes a logical error since the program flow continues with the next case
- Not having a “default” case
  - Errors happen! If there is no “default” then no statements inside the switch will execute when no case matches, hiding the error
  - Always have a default even if all it does it prints an error message or signals an error through other means (asserts, exceptions)

Always define a
"default" case

Commenting Control Structures

- All control structures should contain at least a comment above explaining WHY is the sequence branched
  - Explain the reason in your own words, don't just state the obvious
  - Good Example: "test the number of hours worked to check for overtime"
  - Bad Example: "compare _hoursWorked with 40"
- Comment inside the if / else / case blocks
  - Each case can have a comment to reiterate what is the condition the statements are executing in
  - Examples: "regular hours, paid according to hourly pay", "overtime hours, paid 1.5 regular hourly pay".
- Control structure comments are mandatory (not by the language but by me)