Selection Statements and operators

CSC 1051 – Data Structures and Algorithms I

Dr. Mary-Angela Papalaskari
Department of Computing Sciences
Villanova University

Course website:
www.csc.villanova.edu/~map/1051/

Some slides in this presentation are adapted from the slides accompanying Java Software Solutions by Lewis & Loftus
Selection structures in Java

• Conditional statement:

```java
if (n>0)
    System.out.println(“positive”);
else
    System.out.println(“negative”);
```

• Other selection structures (Chapter 6 in text)
  – the `conditional` operator
  – the `switch` statement
The Conditional Operator

• Similar to an if-else statement, except that it is an expression that returns a value

• For example:

```java
String whatToPrint = (happy ? "happy" : "sad");
```

• If happy is true, then “happy” is assigned to whatToPrint; otherwise, “sad” is assigned to whatToPrint
The Conditional Operator

• Similar to an `if-else` statement, except that it is an expression that returns a value

• For example:

```java
String whatToPrint = (happy ? "happy" : "sad");
```

• The conditional operator requires three operands so it is sometimes called the `ternary` operator
The Conditional Operator Syntax

\[ \text{condition} \ ? \ \text{expression1} \ : \ \text{expression2} \]

- If the \textit{condition} is true, \textit{expression1} is evaluated; if it is false, \textit{expression2} is evaluated.

- The value of the entire conditional operator is the value of the selected expression.

- Remember, the \textit{conditional operator} is \textbf{not a statement}, it is an operator (can be part of an expression).
Another example

```java
System.out.println("Your change is " + count +
    ((count == 1) ? "Dime" : "Dimes");
```

- If `count` equals 1, "Dime" is printed
- If `count` is anything other than 1, then "Dimes" is printed
Quick Check

Rewrite this code using the conditional operator.

```java
if (val <= 10)
    System.out.println("It is not greater than 10.");
else
    System.out.println("It is greater than 10.");
```
Next: The switch Statement

- The switch statement provides a way to implement a multi-way choice (not just true/false)
Recall: Logic of an if-else statement

- condition evaluated
  - true: statement1
  - false: statement2
The **switch statement**: a way to implement a multi-way choice (not just true/false)

Note: this is a simplified flowchart of the logic of the switch statement
The switch Statement

• The *switch* statement provides a way to implement a multi-way choice (not just true/false)

• The *switch* statement evaluates an expression, then attempts to match the result to one of several possible cases

• Each case contains a value and a list of statements

• The flow of control transfers to the statement associated with the first case value that matches
The switch Statement

• An example of a switch statement:

    switch (value)
    {
        case 1:
            System.out.println("one");
            break;
        case 2:
            System.out.println ("two");
            break;
        case 3:
            System.out.println ("three");
            break;
    }
The switch Statement

• Another example – counting letters. Here option is a char

```java
switch (option) {
    case 'A':
        aCount++;
        break;
    case 'B':
        bCount++;
        break;
    case 'C':
        cCount++;
        break;
}
```
The switch Statement in general

- The general syntax of a `switch` statement is:

```
switch (expression)
{
    case value1 :
        statement-list1
    case value2 :
        statement-list2
    case value3 :
        statement-list3
    case ...
}
```

- `switch` and `case` are reserved words.
- If `expression` matches `value2`, control jumps to here.
The switch Statement

- Often a *break statement* is used as the last statement in each case's statement list.

- A `break` statement causes control to transfer to the end of the `switch` statement.

- If a `break` statement is not used, the flow of control will continue into the next case.

- Sometimes this may be appropriate, but often we want to execute only the statements associated with one case.
So… the logic of the switch is more like this:

```
So... the logic of the switch is more like this:

expression evaluated

1  2  3

statements1

statements2

statements3

Note: this is a still simplified flowchart of the logic of the switch statement
```
The switch Statement

- A `switch` statement can have an optional `default` case.

- The default case has no associated value and simply uses the reserved word `default`.

- If the default case is present, control will transfer to it if no other case value matches.

- If there is no default case, and no other value matches, control falls through to the statement after the switch.
The switch Statement

• The type of a switch expression must be integers, characters, or enumerated types

• As of Java 7, a switch can also be used with strings

• You cannot use a switch with floating point values

• The implicit boolean condition in a `switch` statement is equality

• You cannot perform relational checks with a `switch` statement

• See `GradeReport.java`
import java.util.Scanner;

public class GradeReport {
    //--
    // Reads a grade from the user and prints comments accordingly.
    //--
    public static void main (String[] args) {
        int grade, category;

        Scanner scan = new Scanner (System.in);

        System.out.print ("Enter a numeric grade (0 to 100): ");
        grade = scan.nextInt();

        category = grade / 10;

        System.out.print ("That grade is ");
        continue
continue

switch (category)
{
    case 10:
        System.out.println ("a perfect score. Well done.");
        break;
    case 9:
        System.out.println ("well above average. Excellent.");
        break;
    case 8:
        System.out.println ("above average. Nice job.");
        break;
    case 7:
        System.out.println ("average.");
        break;
    case 6:
        System.out.println ("below average. You should see the");
        System.out.println ("instructor to clarify the material "
                        + "presented in class.");
        break;
    default:
        System.out.println ("not passing.");
}
}
continue
switch {
    case 10:
        System.out.println("a perfect score. Well done.");
        break;
    case 9:
        System.out.println("well above average. Excellent.");
        break;
    case 8:
        System.out.println("above average. Nice job.");
        break;
    case 7:
        System.out.println("average.");
        break;
    case 6:
        System.out.println("below average. You should see the");
        System.out.println("instructor to clarify the material " + "presented in class.");
        break;
    default:
        System.out.println("not passing.");
}
Homework

• Review Sections 6.1 and 6.2
  • **Always** do all self-review exercises when you review

• Exercises
  1. Implement the Person class with instance variables for name, age and happiness state (a boolean). Create a driver to test it. Use the conditional operator in implementing the toString() method.
  2. Implement the Dog class, similar to the Person class. In addition to being happy or sad, the dog should also have another state symbolized by an integer: 1=sit; 2=sleep; 3=shake; 4=run

• Use a switch statement in toString() method
Another switch example

- SwitchExample.java