The Conditional Operator

- Can use to create short if/else statements
- Format: `expr ? expr : expr;`

```
x<0 ? y=10 : z=20;
```

First Expression: Expression to be tested
2nd Expression: Executes if first expression is true
3rd Expression: Executes if the first expression is false

The value of a conditional expression is
- The value of the second expression if the first expression is true
- The value of the third expression if the first expression is false
Parentheses () may be needed in an expression due to precedence of conditional operator

The switch Statement

- Used to select among statements from several alternatives
- In some cases, can be used instead of if/else if statements
**switch Statement Format**

```java
switch (expression) //integer
{
    case exp1: statement1;
    case exp2: statement2;
    ...
    case expn: statementn;
    default: statementn+1;
}
```

**The switch Statement in Program 4-23**

```java
switch (expression) //integer
{
    case exp1: statement1;
    case exp2: statement2;
    ...
    case expn: statementn;
    default: statementn+1;
}
```

**switch Statement Requirements**

1. *expression* must be an integer variable or an expression that evaluates to an integer value
2. *exp1* through *expn* must be constant integer expressions or literals, and must be unique in the switch statement
3. default is optional but recommended

**switch Statement—How it Works**

1. *expression* is evaluated
2. The value of *expression* is compared against *exp1* through *expn*
3. If *expression* matches value *expi*, the program branches to the statement following *expi* and continues to the end of the switch
4. If no matching value is found, the program branches to the statement after default:

**break Statement**

- Used to exit a switch statement
- If it is left out, the program "falls through" the remaining statements in the switch statement

**break and default statements in Program 4-25**

```java
// This program is carefully constructed to use the "fall through"
// behavior of the switch statement.
// Include whatever line spacing changes needed.

switch (expression) //integer
{
    case exp1: statement1;
    case exp2: statement2;
    ...
    case expn: statementn;
    default: statementn+1;
}
```
break and default statements in Program 4-25

Using switch in Menu Systems

- switch statement is a natural choice for menu-driven program:
  - display the menu
  - then, get the user's menu selection
  - use user input as expression in switch statement
  - use menu choices as expr in case statements

More About Blocks and Scope

- **Scope** of a variable is the block in which it is defined, from the point of definition to the end of the block
- Usually defined at beginning of function
- May be defined close to first use

Variables with the Same Name

- Variables defined inside { } have local or block scope
- When inside a block within another block, can define variables with the same name as in the outer block.
- When in inner block, outer definition is not available
- Not a good idea

Variables with the Same Name

- Variables defined inside { } have local or block scope
- When inside a block within another block, can define variables with the same name as in the outer block.
- When in inner block, outer definition is not available
- Not a good idea
Two Variables with the Same Name in Program 4-30

Program 4-30

// This program uses two variables with the same name.
// Enter a number greater than 5.
// Enter another number.
int main()
{
    // Define a variable named number.
    int number;
    cout << "Enter a number greater than 5: ">
    cin >> number;
    cout << "Enter another number: ">
    cin >> number;
    cout << "The second number you entered was ">
    << number << endl;
    return 0;
}