Mat 2170
Week 5

Control Statements – Selection
Making Decisions

Spring 2014

Student Responsibilities

▶ Reading: Textbook, Chapter 4
▶ Lab 5
▶ Attendance
▶ EXAM 1
Thursday evening, 7:00pm
Other times on Thursday available

Chapter Four Overview – 4.3 – 4.4

Along with while and for, if and switch are types of control statements
They are used to make decisions or choices
The if and switch statements do NOT cause looping

Format: Simple if Statement

The simple if form is used when an operation is only performed when a particular condition is true, but processing otherwise is not changed.

```java
if (condition) {
    statements to be executed when condition is true
}
```

Flowchart: Simple if Statement

```
ENTER

if (Boolean Expression) true
ACTION
false
EXIT
```

if Statements and Blocks

▶ Single statement clause:

```java
if (condition) {
    statement
}
```

▶ Multi-line if statement requires curly braces

```java
if (condition) {
    statements to be executed when condition is true
    . . .more statements. . .
}
```
Example: Simple if Statement

```java
int sum = 0;
...
int n = readInt("Enter an integer: ");
if (n > 0) // if n is positive,
    sum += n; // add it to sum
```

When the clause is a single statement, braces are optional.

Examples: Simple if Statements

```java
if (n < 0)
    println("Negative number entered");

if (MyOval.getColor() == Color.BLUE)
    MyOval.setColor(Color.RED);

if (counter >= MAX)
{
    println("Maximum reached, resetting counter");
counter = 0;
}
```

Examples: if–else Statement

```java
GRect R = new GRect(x, y, width, height);
R.setFilled(true);
if ((count % 2) == 0)
    R.setColor(Color.RED);
else
    R.setColor(Color.BLUE);
add(R);
```

Format: if–else Statement

The if–else form is used when there are two alternatives: one clause for cases in which the condition is true and the other for cases in which it is false:

```java
if (condition)
{
    statements to be executed if condition is true
}
else
{
    statements to be executed if condition is false
}
```
Example: if–else Statement

```java
int Age = readInt("Enter age (whole years): ");
double TicketPrice;
if ((17 <= Age) && (Age <= 25))
    TicketPrice = 15.0;
else
{
    TicketPrice = 29.0;
    println("You receive no discount.");
}
```

Choosing Between if and if-else

▶ There is no hard-and-fast rule for choosing
▶ Best guideline: think about the problem description (in English) — if it uses else or otherwise, there’s a good chance you need to use if/else
▶ Example:
  ▶ suppose we want to change the AverageList program so it didn’t include any zero values in the average. A single test is needed to ensure zero scores aren’t added in or counted: if (value != 0) ...
  ▶ However, if we wish to keep a count of the zero values, we would need to add an else clause to increment a counter in the case the value entered was equal to 0: if (value == 0) ... else ...

The AverageList Program - Checking For Zeroes

```java
// Initializations
int total = 0; // sum of input values
int count = 0; // count of input values

// Display program information
println("Average a list of non-negative integers.\n");
println("Enter one value per line, " + SENTINEL + " to end\n");
// where SENTINEL is declared below as:
// private static final int SENTINEL = -1;

// get initial input, aka “prime the pump"
int value = readInt("Enter number, " + SENTINEL + " to end: ");

// repeat-until-sentinel pattern
while (value != SENTINEL)
{
    // process only non-zero data
    if (value != 0)
    {
        total += value;
        count++;
    }

    // get next value from user
    value = readInt("Enter number, " + SENTINEL + " to end.");
}

// if list wasn’t empty, display the average
if (count != 0)
    println("Average is: " + (double) total / (double) count);
else
    println("No valid values were entered.");
// end of run()

// constant declaration section
private static final int SENTINEL = -1;
```

Examples: if-else Statement

```java
if (n <= 0)
    println("Non-positive number entered");
else
    sum += n;

MyOval.setFilled(true);
if (((row + col) % 2) == 0)
{
    MyOval.setFillColor(Color.RED);
    MyOval.setColor(Color.YELLOW);
}
else
{
    MyOval.setFillColor(Color.BLUE);
    MyOval.setColor(Color.CYAN);
}
```
The AverageList Program - Counting Zeroes

// Initializations
int total = 0; // sum of input values
int count = 0; // count of input values
int zeroCount = 0; // count of zeroes entered

// get initial input, aka “prime the pump”
int value = readInt("Averaging list, " + SENTINEL + " to end: ");

// repeat-until-sentinel pattern
while (value != SENTINEL)
{
// sum only non-zero data
if (value != 0)
{ total += value;
  count++;
}
// otherwise, count any zeroes entered
else
  zeroCount++;

// get next value from user
value = readInt("Enter number, " + SENTINEL + " to end: ");
}

// if list wasn’t empty, display the average
if (count != 0)
print("Average is: " + (double) total / (double) count + 
"\n");
// otherwise, acknowledge no data was processed
else
print("No valid values were entered.\n");

// display number of zeroes in data entered
println("There were " + zeroCount + " zeroes in data.");

// Summing positive and negative values separately
int posSum = 0;
int negSum = 0;
int zeroCnt = 0;
int n = readInt("Enter value, " + SENTINEL + " to stop: ");
while (n != SENTINEL)
{
if (n > 0) // if n is positive,
posSum += n; // add to positive sum
else if (n < 0) // if n is negative
  negSum += n; // add to negative sum
else // if n is zero, count it
  zeroCnt++;

n = readInt("Enter value, " + SENTINEL + " to stop: ");
}

// repeat-until-sentinel pattern
while (value != SENTINEL)
{
// sum only non-zero data
if (value != 0)
{ total += value;
  count++;
}
// otherwise, count any zeroes entered
else
  zeroCount++;

// get next value from user
value = readInt("Enter number, " + SENTINEL + " to end: ");
}

// if list wasn’t empty, display the average
if (count != 0)
print("Average is: " + (double) total / (double) count + 
"\n");
// otherwise, acknowledge no data was processed
else
print("No valid values were entered.\n");

// display number of zeroes in data entered
println("There were " + zeroCount + " zeroes in data.");
Do Placement of `else`s Matter? **YES!**

```java
int posSum = 0;
int negSum = 0;
int zeroCount = 0;
int n = readInt("Enter value, " + SENTINEL + " to stop: ");
while (n != SENTINEL) {
    if (n > 0) // if n is positive,
        posSum += n; // add to positive sum
    if (n < 0) // if n is negative
        negSum += n; // add to negative sum
    else // else
        zeroCount++; // count it
    n = readInt("Enter value, " + SENTINEL + " to stop: ");
}
```

An else is paired with the nearest available, unmatched `if` above it.

As Do the Placement of Braces — **Logic Error!**

```java
int posSum = 0;
int negSum = 0;
int zeroCount = 0;
int n = readInt("Enter value, " + SENTINEL + " to stop: ");
while (n != SENTINEL) {
    if (n > 0) // if n is positive,
        { // add to positive sum
            posSum += n;
            if (n < 0) // if n is negative
                negSum += n; // add to negative sum
        }
    else // else
        zeroCount++; // count it
    n = readInt("Enter value, " + SENTINEL + " to stop: ");
}
```

And Then There’s… **Inefficiency**

```java
// Summing positive and negative values separately
// and counting number of times zero is entered
int posSum = 0;
int negSum = 0;
int zeroCount = 0;
int n = readInt("Enter value, " + SENTINEL + " to stop: ");
while (n != SENTINEL) {
    if (n > 0) // if n is positive,
        posSum += n; // add to positive sum
    if (n < 0) // if n is negative
        negSum += n; // add to negative sum
    else // else
        zeroCount++; // count it
    n = readInt("Enter value, " + SENTINEL + " to stop: ");
}
```

Which Is Required: **while or if Statement?**
I.e., a loop or a selection statement?

```java
while / loop – or – if / selection?

- to determine whether \( x \) is positive or not
- to determine the sum of a list of values entered by a user
- to eat cookies as long as the cookie jar is not empty
- to find whether there is sufficient money to buy that CD
- to balance a checkbook over several statements
- to determine whether \( GOval \ C \) is Cyan
```

**Format: switch Statement**

The `switch` statement provides a convenient syntax for choosing from among a small set of possible paths:

```java
switch ( expr ) {
    case \( v_1 \) :  // statements executed if \( expr = v_1 \)
        break;
    case \( v_2 \) :  // statements executed if \( expr = v_2 \)
        break;
    ...more case clauses if needed...
    default:          // statements executed if no values match
        break;
}
```
Example: `switch` Statement

```java
int month = readInt("Enter month (Jan=1): ");
switch ( month ) {
    case 2:
        println("28 days, 29 in leap years");
        break;
    case 4: case 6: case 9: case 11:
        println("30 days");
        break;
    case 1: case 3: case 5: case 7:
            case 8: case 10: case 12:
        println("31 days");
        break;
    default:
        println("Illegal month number");
        break;
}
```

Writing Boolean Expressions — Consider the Red Region

We want a boolean expression that is true exactly when \( x \) is in the given interval, i.e., between 5 and 10, inclusive.

Mathematically:
\[ x \in [5, 10] \]

Java Expression:
\((5 \leq x) \land (x \leq 10)\)

An Expression Which Is True for Values in the Red Regions:

Mathematically:
\[ x \in [-10, -5] \cup [5, 10] \]

Java Expression:
\((-10 \leq x) \land (x \leq -5) \lor ((0 \leq x) \land (x \leq 10))\)

An Expression Which Is True for Values in the Red Regions:

Mathematically:
\[ x \in [-10, -5] \cup [0, 5] \cup [10, 15] \]

Java Expression:
\((-10 \leq x) \land (x \leq -5) \lor ((0 \leq x) \land (x \leq 5)) \lor ((10 \leq x) \land (x \leq 15))\)