Chapter 5

Control Structures II

Chapter Objectives

- Learn about repetition (looping) control structures
- Explore how to construct and use count-controlled, sentinel-controlled, flag controlled, and EOF-controlled repetition structures
- Examine break and continue statements
- Discover how to form and use nested control structures

Why Is Repetition Needed?

- There are many situations in which the same statements need to be executed several times
- Example
  - Formulas used to find average grades for students in a class
The while Looping (Repetition) Structure

- Syntax:
  ```
  while(expression)
  statement
  ```
- Expression is always true in an infinite loop
- Statements must change value of expression to false

Counter-Controlled while Loop

- Used when exact number of data or entry pieces is known
- Syntax:
  ```
  int N = //value input by user or specified in program;
  int counter = 0;
  while(counter < N){
    statement(s);
    counter++;
  }
  ```

Sentinel-Controlled while Loop

- Used when exact number of entry pieces is unknown but last entry (special/sentinel value) is known
- Syntax:
  ```
  input first data item into variable;
  while(variable != sentinel){
    statement(s);
    input a data item into variable;
  }
  ```
Flag-Controlled while Loop

- Boolean value used to control loop
- Syntax:
  ```java
  boolean found = false;
  while(!found){
    statement(s);
    if(expression)
      found = true;
  }
  ```

EOF(End of File)-Controlled while Loop

- Used when input is from files
- Sentinel value is not always appropriate
- Syntax:
  ```java
  input the first data item;
  while(not_end_of_input_file){
    statement(s);
    input a data item;
  }
  ```
  - read method with value –1 can be used as expression in while

Programming Example: Checking Account Balance

- Input file: customer’s account number, account balance at beginning of month, transaction type (withdrawal, deposit, interest), transaction amount
- Output: account number, beginning balance, ending balance, total interest paid, total amount deposited, number of deposits, total amount withdrawn, number of withdrawals
Programming Example: Checking Account Balance

• Solution:
  – Read data
  – EOF-controlled loop
  – switch structure of transaction types
  – Determine action (add to balance or subtract from balance depending on transaction type)

Programming Example: Fibonacci Number

• Fibonacci formula for any Fibonacci sequence:
  \[ a_n = a_{n-1} + a_{n-2} \]

• Input: first two Fibonacci numbers in sequence, position in sequence of desired Fibonacci number (n)
  – int previous1 = Fibonacci number 1
  – int previous2 = Fibonacci number 2
  – int nthFibonacci = position of nth Fibonacci number

• Output: nth Fibonacci number

Programming Example: Fibonacci Number (Solution)

• code for while loop based on inputs:
  ```
  int counter = 3; // Fibonacci 1 and 2 already known
  while(counter <= nthFibonacci){
    current = previous2 + previous1;
    previous1 = previous2;
    previous2 = current;
    counter++;
  }
  ```

• Final result found in last value of current
The for Looping (Repetition) Structure

- Specialized form of while loop
- Simplifies the writing of count-controlled loops
- Syntax
  
  ```
  for(initial statement; loop condition; update statement)
  {
    statement(s);
  }
  ```

Execution:
- initial statement executes
- loop condition evaluated
- If loop condition evaluates to true, execute for loop statement and execute update statement
- Repeat until loop condition is false

- Does not execute if initial condition is false
- Update expression changes value of loop control variable, eventually making it false
- If loop condition is always true, result is an infinite loop
- Infinite loop can be specified by omitting all three control statements
- If loop condition is omitted, it is assumed to be true
- for statement ending in semicolon is empty; does not effect program
Programming Example: Classify Numbers

- Input: N integers (positive, negative, and zeros)
  
  ```
  int N = 20; // N easily modified
  ```
- Output: number of 0s, number of even integers, number of odd integers

Programming Example: Classify Numbers (Solution)

- Use a for loop to get and evaluate 20 numbers from the user and input into switch structure
- `switch(number % 2)`
- `case 0:
  - if(number == 0) increment number of zeros;
  - if(number != 0) increment number of even integers;
- case 1:
  - increment number of odd integers

The do…while Loop (Repetition) Structure

- Syntax
  ```
  do{
  statement(s);
  }
  while(expression);
  ```
- Statements executed first, then expression evaluated
- Statement(s) executed at least once then continued if expression is true
do…while Loop (Post-test Loop)

- Used to exit early from a loop
- Used to skip remainder of switch structure
- Can be placed within if statement of a loop
  - If condition is met, loop exited immediately

break Statements

- Used to exit early from a loop
- Used to skip remainder of switch structure
- Can be placed within if statement of a loop
  - If condition is met, loop exited immediately

continue Statements

- Used in while, for, and do…while structures
- When executed in a loop, the remaining statements in the loop are skipped; proceeds with the next iteration of the loop
- When executed in a while/do…while structure, expression evaluated immediately after continue statement
- In a for structure, the update statement is executed after the continue statement; then the loop condition executes
Nested Control Structures

• Provides new power, subtlety, and complexity
• if, if…else, and switch structures can be placed within while loops
• for loops can be found within other for loops

Nested Control Structures (Example)

• for(int i = 1; i <= 5; i++)
  for(int j = 1; j <= i; j++)
    System.out.print("*" + i);
    System.out.println();
} 
• Output:
  *
  **
  ***
  ****
  *****

Chapter Summary

• Looping Mechanisms
  – Counter-controlled while loop
  – Sentinel-controlled while loop
  – Flag-controlled while loop
  – EOF-controlled while loop
  – for loop
  – do…while loop
• break statements
• continue statements
• Nested control structures