# Java Programming: Guided Learning with Early Objects
## *Chapter 5*
## *Control Structures II: Repetition*

- Learn about repetition (looping) control structures
- Explore how to construct and use:
  - Counter-controlled repetition structures
  - Sentinel-controlled repetition structures
  - Flag-controlled repetition structures
  - EOF-controlled repetition structures
- Examine break statements in loops
- Discover how to form and use nested control structures
- Learn how to avoid bugs by avoiding patches

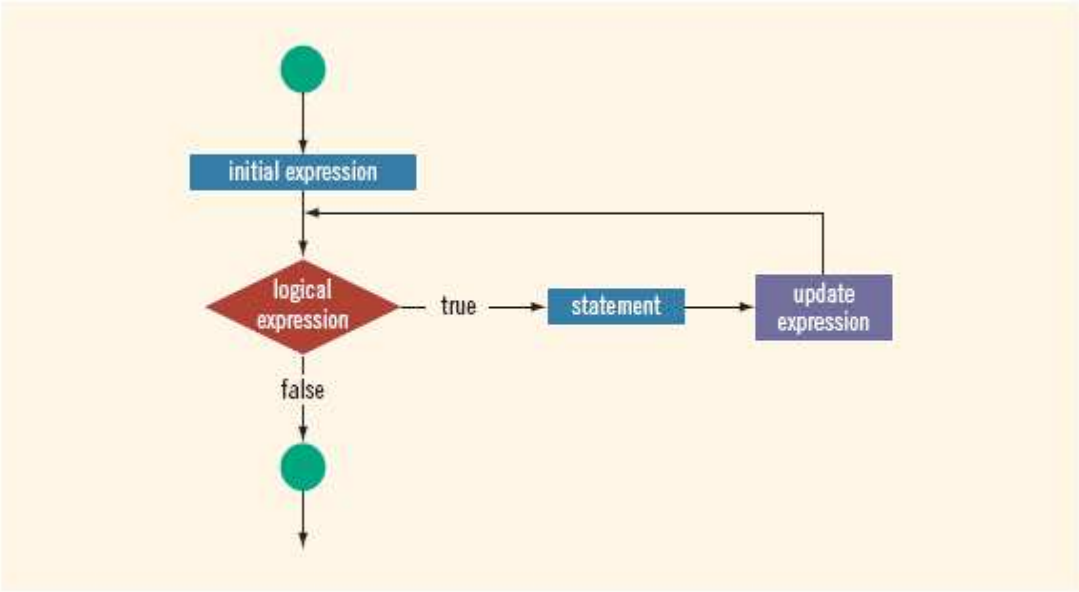
## Why Is Repetition Needed?

- Many situations in which it is necessary to repeat a set of statements
- Three looping (repetition) structures:
  - `while`
  - `for`
  - `do…while`
- Allow repeating statements over and over until certain conditions are met



## ***Learning about the Loop Structure***

- Loop- A structure that allows repeated execution of a block of statements
- Loop body- A block of statements; as long as the expression is true, the loop body executes
- Iteration- One execution of any loop
- logical expression is called a **loop condition**
- statement may be simple or compound
- If logical expression evaluates to true the statement executes
- **Infinite loop**: executes indefinitely
- Loop control variable must be initialized before executing the loop
- If semicolon is at the end of the loop, the action of the loop is empty or null



## *Using Shortcut Arithmetic Operators*

- To increase a variable's value by exactly one:
  - **prefix ++**
    - Used before the variable name
      - ++someValue;
  - **postfix ++**
    - Used after the variable name
      - anotherValue++;

## *Example :-*

```
public class MyLoop {

public static void main(String[] args) {
          int v=4;
           int abc= ++v; // before execution, v is incremented by one.
           int xyz= v++; // after execution , v is incremented by one.
           System.out.println("v is "+v);
           System.out.println("++v is "+abc);
           System.out.println("v++ is "+xyz);
}        }
```

## *Using a for Loop*

- for loop- A special loop that is used when a definite number of loop iterations is required
- Keyword **for**
- Set of parentheses
- Three sections within parentheses
  - Initializing the loop control variable
  - Testing the loop control variable
  - Updating the loop control variable



## *for loop syntax:*

- `for (initial expr;logical expr;update expr)` statement
- logical expr is the **loop condition**
- All three expressions are for loop control expressions
- **for** loop body may contain simple or compound statements

## *Execution of for loop:*

- `initial expr` executes
- `logical expr` evaluated
  - If loop condition evaluates to `true`, execute `statement`
  - Execute `update statement`
- Repeat until loop condition evaluates to `false`

- Primarily used to implement counter-controlled loops Called **counted** or **indexed**
- update expr should change value of loop control variable
- If logical expr omitted, it is assumed true

## *Example :-*

```
public class MyLoop {

public static void main(String[] args) {
int i;
  for (  i = 0 ; i<=3 ; i++)  {
     System.out.println("i->>"+i);        }
  System.out.println("Yes:"+i);

}        }
```



# ***Examples :-***

```
public class MyLoop {

public static void main(String[] args) {
int i;
  for (  i = 5 ; i>=3 ; --i){
        System.out.println("i->>"+i);      }
  System.out.println("Yes:"+i);
}         }
```

```
public class MyLoop {

public static void main(String[] args) {
int i,j;
  for ( i=0 ,j=10; (  i<=10 && j > 5  ) ; i++ , j--)     {
    System.out.println("i->>  "+i+"  J-->  "+j);         }
    System.out.println("Yes:"+i);
}         }
```

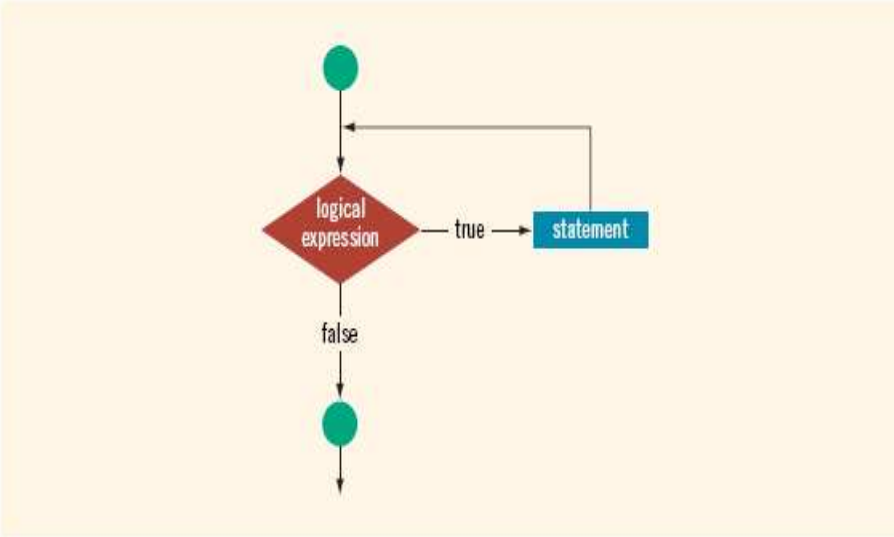


# *Using the while Loop*

- **while** Loop- To execute a body of statements continually as long as the Boolean expression continues to be true
  - Consists of the keyword **while** followed by a Boolean expression within parentheses followed by the body of the loop
  - Use when you need to perform a task a predetermined number of times
- logical expression is called a **loop condition**
- statement may be simple or compound
- If logical expression evaluates to true the statement executes
- **Infinite loop**: executes indefinitely
- Loop control variable must be initialized before executing the loop
- If semicolon is at the end of the loop, the action of the loop is empty or null

## *Syntax:*

```
while (logical expression)
    statement
```



# *Using a while Loop*

- **Incrementing** – Altering a loop by adding one to loop control variable
- **Decrementing** – Altering a loop by subtracting one from a loop control variable
- **Sentinel value**- A value that a user must supply to stop a loop
- **Accumulating**- Adding a value to a variable to get a new value in the same variable

# *Counter-Controlled while Loops*

- Number of loop iterations known in advance
  - Assume *n* iterations
- Counter initialized to 0 before while statement
- ***Before executing loop body, counter compared to n***
- ***If counter less than n, loop body executes***
- ***Inside loop body, counter incremented***

# *Example :-*

```
public class MyLoop {

public static void main(String[] args) {
int myval=1;
     while (myval < 5 ){
        System.out.println(myval);
        myval++;
        myval+=2; // myval=myval+2;
        myval++; // myval=myval+1;
        myval--; // myval=myval-1;
        myval-=2; // myval=myval-2;      }
}       }
```



## ***Sentinel-Controlled while Loops***

- **Sentinel**: special value that signals end of processing
- Read first item before entering while statement
- If item is not the sentinel, loop body executes
- Loop continues as long as sentinel value not encountered
    - `while (variable != sentinel)`

## ***Example :-***

```
import java.util.*;
public class MyLoop {
       static Scanner console = new Scanner(System.in);
public static void main(String[] args) {
   System.out.println("Program Written By Husain Gholoom");
    System.out.println("The function of this program");
    System.out.println("is to illustrate the while loop with");
    System.out.println("Sentinel Value");
    System.out.println("Enter a Number in order to terminate the loop ");

     int sentinelValue, myval=1;
     sentinelValue = console.nextByte();
     while (myval < sentinelValue ){
          System.out.println(myval);
          myval++;
          myval+=2; // myval=myval+2;
          myval++; // myval=myval+1;
          myval--; // myval=myval-1;
          myval-=2; }// myval=myval-2;
}        }
```



# Flag-Controlled while Loops

- Uses a `boolean` variable to control loop
- The `boolean` variable is known as a flag
    - Must eventually be set to `true` in loop body
        - `boolean found = false;`
        - `while (!found)`

## Example :-

```
import java.util.*;
public class MyLoop {
       static Scanner console = new Scanner(System.in);
public static void main(String[] args) {
    System.out.println("Program Written By Husain Gholoom");
    System.out.println("This program reads a sequence of positive integers input");
     System.out.println("by the user, and it will print out the average of those");
     System.out.println("integers.  The user is prompted to enter one integer at a");
     System.out.println("time.  The user must enter a 0 to mark the end of the");
     System.out.println("data.  (The zero is not counted as part of the data to");
     System.out.println("be averaged.)  The program does not check whether the");
     System.out.println("user's input is positive, so it will actually work for");
     System.out.println("both positive and negative input values.");

        int inputNumber;   // One of the integers input by the user.
        int sum;           // The sum of the positive integers.
        int count;         // The number of positive integers.
        double average;    // The average of the positive integers.

        /* Initialize the summation and counting variables. */

        sum = 0;
        count = 0;
```



# ***Continue Example :-***

```
/* Read and process the user's input. */

        System.out.println("Enter your first positive integer: ");
        inputNumber = console.nextInt();

        while (inputNumber != 0) {
          sum += inputNumber;   // Add inputNumber to running sum.
          count++;              // Count the input by adding 1 to count.
          System.out.println("Enter your next positive integer, or 0 to end: ");
          inputNumber = console.nextInt();
        }

        /* Display the result. */

        if (count == 0) {
          System.out.println("It seems that You didn't enter any data!");
        }
        else {
          average = ((double)sum) / count;
          System.out.println();
          System.out.println("You entered " + count + " positive integers.");
          System.out.printf("Their average is %.4f %n", average);
        }
}       }
```

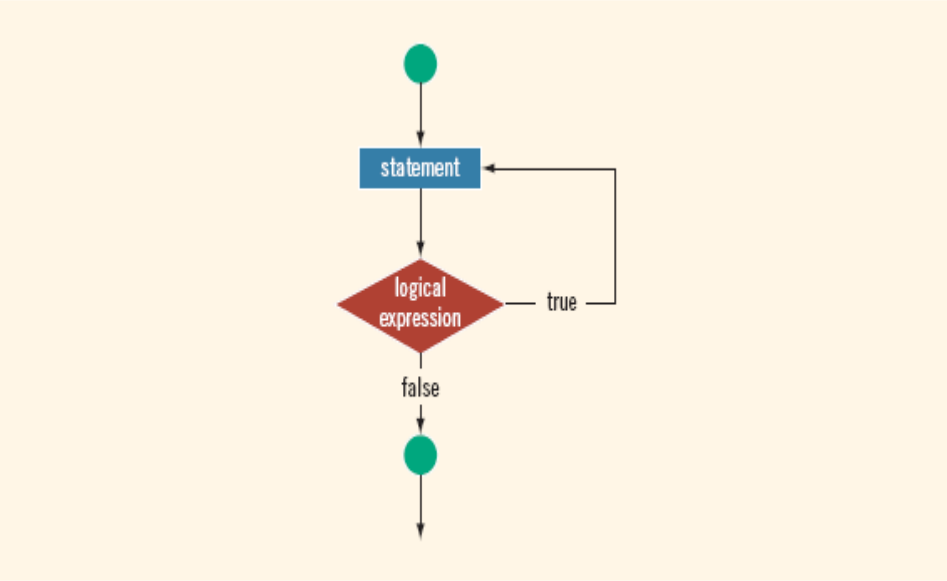


## *Learning How and When to Use a do…while loop*

- Test at the bottom of the loop after one repetition has occurred
- Loop body executes at least **one** time
- The loop starts with the keyword **do**
- The body of the loop is contained within curly braces

## *Syntax:*

```
do
  statement
while (logical expr);
```




- while and for loops have entry conditions
  - May never execute
- do…while loop has exit condition
  - Always executes at least once



## *Choosing the Right Looping Structure*

- for loop: number of repetitions known in advance
- while loop: number of repetitions cannot be determined in advance
- do…while loop: number of repetitions not known, but must be at least one

## *Example :-*

```
public class MyLoop {

public static void main(String[] args) {
//  The do statement that is the repeat statement in pascal
  int i = 3;
  do{
     System.out.println("i-->"+i);
      i++;
  }  while (i <5);          }         }
```



## ***Loops and the break Statement***

- break statement alters flow of program control
- In switch structure, provides immediate exit from structure
- Used in while, for, do…while loops to exit from structures before loop ends
- Two purposes:
  - Exit early from a loop
  - Skip remainder of `switch` structure
- After break statement executes, loop terminates
  - Remaining loop statements skipped
  - Program continues at first statement after loop

## ***Example :-***

```
public class  breakStatement{

public static void main(String[] args) {
while (true) {  // looks like it will run forever!
            System.out.println("Enter a positive number: ");
           int N = console.nextInt();
            if (N > 0)   // input is OK; jump out of loop
              break;
            System.out.println("Your answer must be > 0.");  }

      System.out.println("I Am Out of the loop ");
       }        }
```



# Nested for Loop

## Example :-

```
public class MyLoop {

public static void main(String[] args) {
int i,j;
int i,j;
  for ( i=0 ;   i<=2 ; i++)     {
       for ( j=10; j > 8   ; j--)     {

    System.out.println("i->>  "+i+"  J-->  "+j);        }
    System.out.println("Yes:"+i);  }
}        }
```

## Infinite Loop

- Infinite loop- A loop that never ends

## Example :-

```
While ( 4 > 2 )   System.out.println(“Hello”);
```

**an infinite loop** ***{ Do Not Try this Example }***

Can omit all three statements:

```
for (;;)
```



# *More on Expressions in while Statements*

- `while` loop may be controlled by single variable
- Logical expression in `while` statement may be complex

```
while ((numGuess < 7) && (!guessRight))
```

## Key Terms

- **control expressions** – Expressions in a `for` loop that control the body of the `for` statement.
- **counted `for` loop** – Another name for a `for` loop, so called because a `for` loop implements a counter-controlled loop.
- **counter-controlled `while` loop** – A form of `while` loop in which the number of iterations is stored in a counter, which is incremented or decremented each time the loop iterates.
- **EOF (End-of-File)-controlled `while` loop** – A `while` loop that executes until an end-of-file marker is detected.
- **Fibonacci number** – A number determined by the Fibonacci sequence.
- **Fibonacci sequence** – The *n*th number in the sequence is the sum of the previous two, for example 1, 1, 2, 3, 5, 8, 13, 21, 34, …
- **flag-controlled `while` loop** – A form of `while` loop that uses a `boolean` variable to control the loop.
- **indexed `for` loop** – Another name for a `for` loop, so called because a `for` loop implements a counter-controlled loop.
- **infinite loop** – A loop that executes endlessly.
- **loop condition –** Logical expression that determines whether the loop body executes.
- **loop control variable** – Variables the loop condition determines satisfy certain conditions.
- **origin** – The point (0,0) in a coordinate system.
- **posttest loop** – Loop whose condition is evaluated after executing the loop body.
- **pretest loop** – Loop whose condition is evaluated before executing the loop body.
- **sentinel** – Special value that determines whether a loop should terminate.
- **sentinel-controlled `while` loop** – A loop that continues to execute until a sentinel value is detected.