COMP 202

Programming With Iterations

CONTENT:
• The WHILE, DO and FOR Statements

Repetition Statements

- *Repetition statements* or *iteration* allow us to execute a statement multiple times repetitively
- They are often simply referred to as *loops*
- Like conditional statements, they are controlled by boolean expressions
- Java has three kinds of repetition statements: the *while loop*, the *do loop*, and the *for loop*
- The programmer must choose the right kind of loop for the situation

Part 1

The WHILE Statement

The while Statement

- The *while statement* has the following syntax:

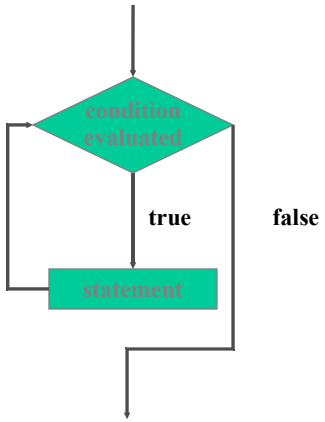
`while` is a reserved word

```
while ( condition )
   statement;
```

If the condition is true, the statement is executed.
Then the condition is evaluated again.

The statement is executed repetitively until
the condition becomes false.

Logic of a while loop



The while Statement

- Note that if the condition of a while statement is false initially, the statement is never executed
- Therefore, the body of a while loop will execute zero or more times

Counter.java

```
public class Counter
{
   public static void main (String[] args)
   {
      final int LIMIT = 5;
      int count = 1;

      while (count <= LIMIT)
      {
         System.out.println (count);
         count = count + 1;
      }

      System.out.println ("Done.");
   }
}
```

What does this print out?

BusPercentage.java

```
final int NUM_SEATS = 56;
int passengers;  double ratio;
Scanner scan = new Scanner(System.in);

System.out.print ("Enter the number of passengers (0 to " + NUM_SEATS +
   "): ");
passengers = scan.nextInt();

while (passengers < 0 || passengers > NUM_SEATS){
   if (passengers > NUM_SEATS)
      System.out.print ("Too many...Please reenter: ");
   else
      System.out.print ("That can't be...Please reenter: ");
   passengers = scan.nextInt();
}

ratio = (double)passengers / NUM_SEATS;

DecimalFormat fmt = new DecimalFormat("0.##");
System.out.println ("The bus is " + fmt.format(ratio*100) + "% full.");
```

How does this do what it does?

Infinite Loops

- The body of a while loop must eventually make the condition false
- If not, it is an *infinite loop*, which will execute until the user interrupts the program
- This is a common type of logical error
- You should always double check to ensure that your loops will terminate normally

Abyss.java

```
public class Abyss
{
public static void main (String[] args)
   {
      int count = 1;

      System.out.println ("I'm going in...");

      while (count <= Integer.MAX_VALUE)
      {
         System.out.println (count);
         count = count - 1;
      }

      System.out.println ("Found the bottom of the abyss!");
   }
}
```

What is wrong here?
Which statement is never reached?

Nested Loops

- Similar to nested if statements, loops can be nested as well
- That is, the body of a loop could contain another loop
- Each time through the outer loop, the inner loop will go through its entire set of iterations

PalindromeTester.java

```
String str, another = "y";   int left, right;
Scanner scan = new Scanner (System.in);

while (another.equalsIgnoreCase("y")) {  // allows y or Y
  System.out.println ("Enter a potential palindrome:");
  str   = scan.nextLine();
  left  = 0;
  right = str.length() - 1;

  while (str.charAt(left) == str.charAt(right) && left < right){
     left++;
     right--;
  }

  if (left < right)
     System.out.println ("That string is NOT a palindrome.");
  else
     System.out.println ("That string IS a palindrome.");

  System.out.print ("Test another palindrome (y/n)? ");
  another = scan.nextLine();
}
```

Explain how this program works.

Part 2

The DO Statement

The do Statement

- The *do statement* has the following syntax:

Uses both the `do` and `while` reserved words

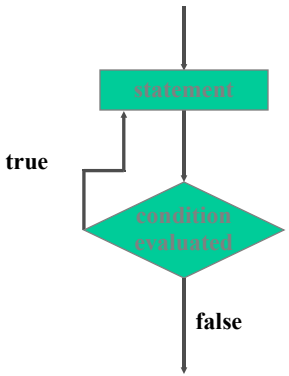
```
do
{
    statement;
}
while ( condition );
```

The statement is executed once initially, then the condition is evaluated

The statement is repetitively executed until the condition becomes false

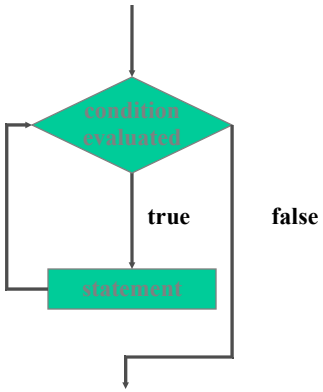
Do loops execute STATEMENT, always, at least once!

Logic of a do loop

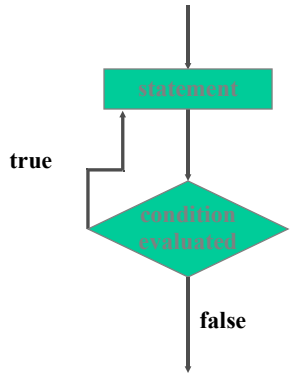


Comparing the while and do loops

while loop



do loop



The do Statement

- A do loop is similar to a while loop, except that the condition is evaluated after the body of the loop is executed
- Therefore the body of a do loop will execute at least one time

Counter2.java

```
public class Counter2
{
   public static void main (String[] args)
   {
      final int LIMIT = 5;
      int count = 0;

      do
      {
         count = count + 1;
         System.out.println (count);
      }
      while (count < LIMIT);

      System.out.println ("Done.");
   }
}
```

What does this print out?

ReverseNumber.java

```
import java.util.Scanner;
public class ReverseNumber
{
   public static void main (String[] args)
   {
      int number, lastDigit, reverse = 0;
      Scanner scan = new Scanner (System.in);

      System.out.print ("Enter a positive integer: ");
      number = scan.nextInt();

      do {
         lastDigit = number % 10;
         reverse = (reverse * 10) + lastDigit;
         number = number / 10;
      }while (number > 0);

      System.out.println ("That number reversed is " + reverse);
   }
}
```

How does this program work?

Part 3

The FOR Statement

The for Statement

- The *for statement* has the following syntax:

Reserved word

The *initialization* portion is executed once before the loop begins

The statement is executed until the *condition* becomes false

```
for ( initialization ; condition ; increment )
   statement;
```

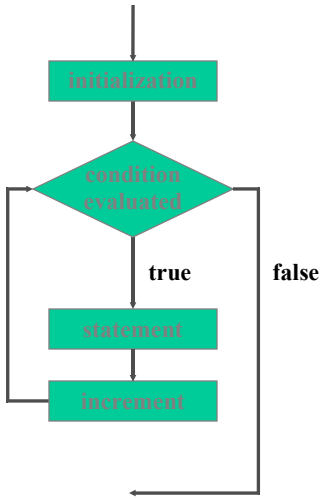
The *increment* portion is executed at the end of each iteration

The for Statement

- A for loop is equivalent to the following while loop structure:

```
initialization;
while ( condition )
{
   statement;
   increment;
}
```


Logic of a for loop



The for Statement

- Like a while loop, the condition of a for statement is tested prior to executing the loop body
- Therefore, the body of a for loop will execute zero or more times
- It is well suited for executing a specific number of times that can be determined in advance

Counter3.java

```
public class Counter3
{
   public static void main (String[] args)
   {
      final int LIMIT = 5;

      for (int count=1; count <= LIMIT; count++)
         System.out.println (count);

      System.out.println ("Done");
   }
}
```

What does this print out?

Multiples.java

```
final int PER_LINE = 5;
int value, limit, mult, count = 0;
Scanner scan = new Scanner(System.in);

System.out.print ("Enter a positive value: ");
value = scan.nextInt();
System.out.print ("Enter an upper limit: ");
limit = scan.nextInt();

System.out.println ("The multiples of " + value + " between " +
        value + " and " + limit + " (inclusive) are:");

for (mult = value; mult <= limit; mult += value) {
   System.out.print (mult + "\t");
   // Print a specific number of values per line of output
   count++;
   if (count % PER_LINE == 0)
      System.out.println();
}
```

What is this program doing? How?

Stars.java

```
public class Stars
{
   public static void main (String[] args)
   {
      final int MAX_ROWS = 10;

      for (int row = 1; row <= MAX_ROWS; row++)
      {
         for (int star = 1; star <= row; star++)
            System.out.print ("*");

         System.out.println();
      }
   }
}
```

What shape does this print out?

X.java

```
final int MAX_ROWS = 10;

for (int row = 1; row <= MAX_ROWS; row++) {
   for (int space = 1; space <= MAX_ROWS-row; space++)
      System.out.print (" ");
   for (int star = 1; star <= row*2; star++)
      System.out.print ("*");
   System.out.println ();
}
for (int trunc=3; trunc>0; trunc--){
   for (int space = 1; space <= MAX_ROWS-1; space++)
      System.out.print (" ");
   System.out.println("**");
}
```

What shape does this print?

The for Statement

- Each expression in the header of a for loop is optional
  - If the initialization is left out, no initialization is performed
  - If the condition is left out, it is always considered to be true, and therefore creates an infinite loop
  - If the increment is left out, no increment operation is performed
- Both semi-colons are always required in the for loop header

Part 4

Thinking Like A Programmer

Top-Down Program Development

- This refers to a way of thinking when you try to solve a programming problem:
  1. First, read and understand the problem
  2. Then, subdivide the problem into chunks. Each chunk is one task, like: initializing variables, inputs, outputs, if-statements and loops.
  3. Then order all these elements in the correct order.
  4. Lastly, only now start writing your code
- Steps 1-3 are either done in your head or on scrap paper. They are not done using the language editor or compiler. They do not need to be in Java even, they could be in simple English/French/etc.

For Example

- Let me show you how to use top-down to solve:
  - Make a program that asks the user for a positive integer number (at least 2). The program then displays that numbers and its greatest prime factor. The program repetitively does this until the user inputs a number less than 2.

Try This Problem

- Write a program that asks the user for a positive integer number (including zero). The program then displays a solid square followed by a hollow square. The program does nothing if the user inputs a negative value.

N = 4

```
****    ****
****    *  *
****    *  *
****    ****
```


Split the Problem

- Get the input and check for correctness
- A general solution for n >= 3 is clearly possible:
  - How are the first and last line printed?
    - How do you space solid from hollow text?
  - How is a solid part of a square printed?
  - How is a hollow part of a square printed?
  - Print top, solid and hollow parts, and bottom
- What about n=0, 1, 2 ?
  - Think of how your program will react