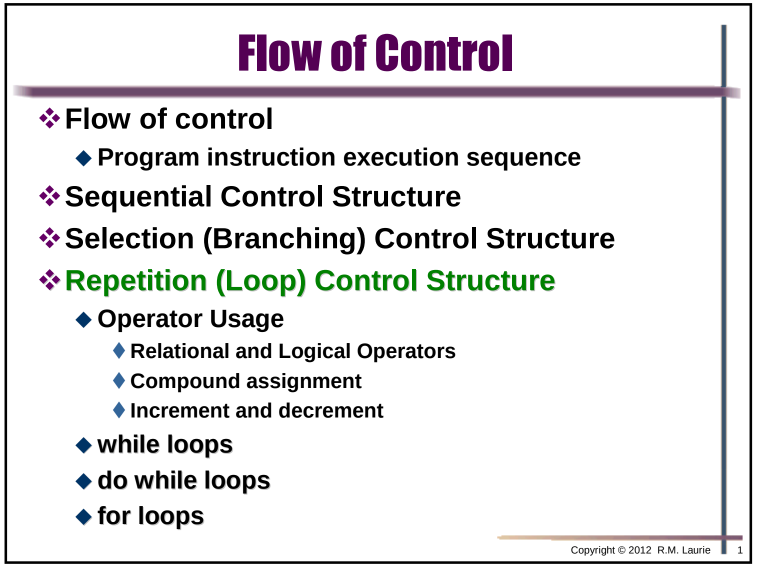
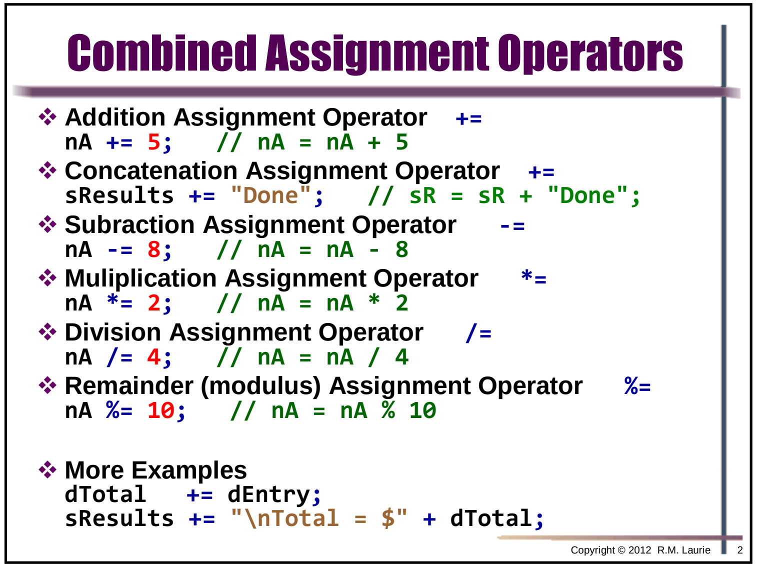


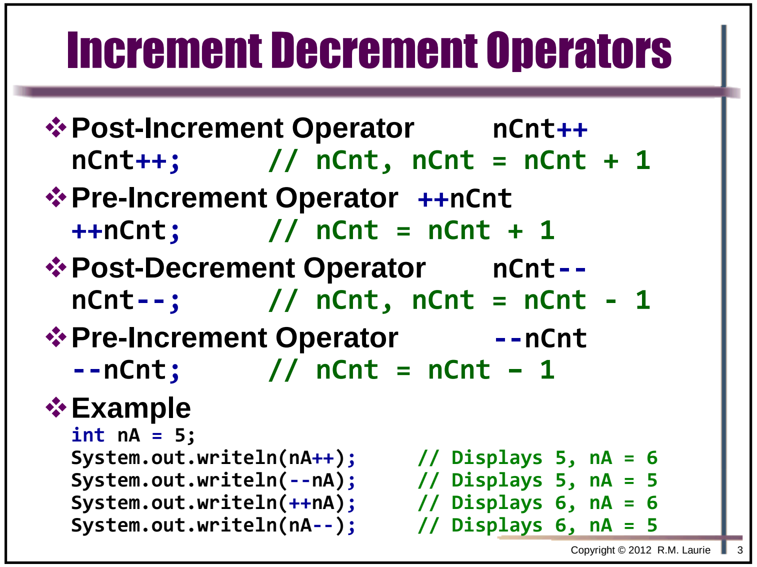
Flow of Control

- Flow of control
  - Program instruction execution sequence
- Sequential Control Structure
- Selection (Branching) Control Structure
- Repetition (Loop) Control Structure
  - Operator Usage
    - Relational and Logical Operators
    - Compound assignment
    - Increment and decrement
  - while loops
  - do while loops
  - for loops



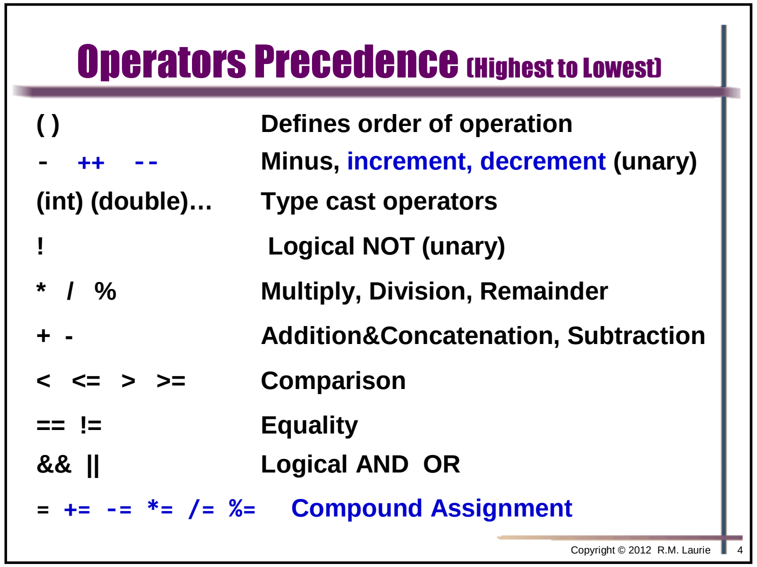
Combined Assignment Operators

- Addition Assignment Operator `+=`
```
nA += 5;    // nA = nA + 5
```
- Concatenation Assignment Operator `+=`
```
sResults += "Done";    // sR = sR + "Done";
```
- Subraction Assignment Operator `-=`
```
nA -= 8;    // nA = nA - 8
```
- Muliplication Assignment Operator `*=`
```
nA *= 2;    // nA = nA * 2
```
- Division Assignment Operator `/=`
```
nA /= 4;    // nA = nA / 4
```
- Remainder (modulus) Assignment Operator `%=`
```
nA %= 10;    // nA = nA % 10
```
- More Examples
```
dTotal    += dEntry;
sResults += "\nTotal = $" + dTotal;
```



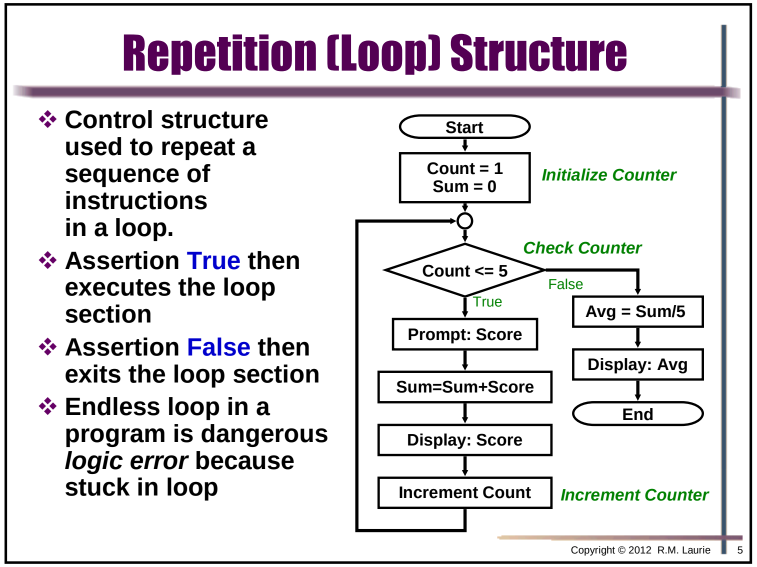
Increment Decrement Operators

- Post-Increment Operator `nCnt++`
```
nCnt++;      // nCnt, nCnt = nCnt + 1
```
- Pre-Increment Operator `++nCnt`
```
++nCnt;      // nCnt = nCnt + 1
```
- Post-Decrement Operator `nCnt--`
```
nCnt--;      // nCnt, nCnt = nCnt - 1
```
- Pre-Increment Operator `--nCnt`
```
--nCnt;      // nCnt = nCnt – 1
```
- Example
```
int nA = 5;
System.out.writeln(nA++);      // Displays 5, nA = 6
System.out.writeln(--nA);      // Displays 5, nA = 5
System.out.writeln(++nA);      // Displays 6, nA = 6
System.out.writeln(nA--);      // Displays 6, nA = 5
```



Operators Precedence (Highest to Lowest)

( )	Defines order of operation
- ++ --	Minus, increment, decrement (unary)
(int) (double)…	Type cast operators
!	Logical NOT (unary)
* / %	Multiply, Division, Remainder
+ -	Addition&Concatenation, Subtraction
< <= > >=	Comparison
== !=	Equality
&& ||	Logical AND OR
= += -= *= /= %=	Compound Assignment



Repetition (Loop) Structure

- Control structure used to repeat a sequence of instructions in a loop.
- Assertion True then executes the loop section
- Assertion False then exits the loop section
- Endless loop in a program is dangerous *logic error* because stuck in loop

Start

Count = 1
Sum = 0

Initialize Counter

Check Counter

Count <= 5

False

True

Avg = Sum/5

Prompt: Score

Display: Avg

Sum=Sum+Score

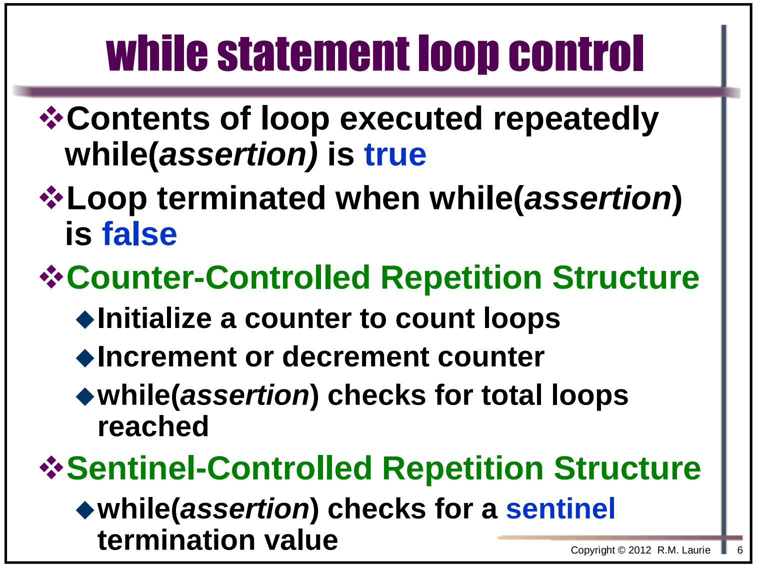
End

Display: Score

Increment Count

Increment Counter

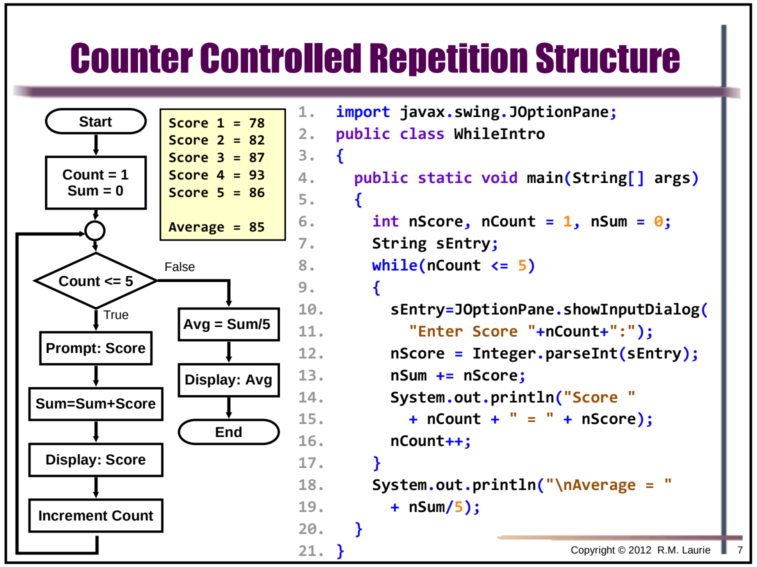
5



while statement loop control

- Contents of loop executed repeatedly while(*assertion*) is true
- Loop terminated when while(*assertion*) is false
- Counter-Controlled Repetition Structure
  - Initialize a counter to count loops
  - Increment or decrement counter
  - while(*assertion*) checks for total loops reached
- Sentinel-Controlled Repetition Structure
  - while(*assertion*) checks for a sentinel termination value

6



Counter Controlled Repetition Structure

Start

Count = 1
Sum = 0

Count <= 5

False

True

Avg = Sum/5

Prompt: Score

Display: Avg

Sum=Sum+Score

End

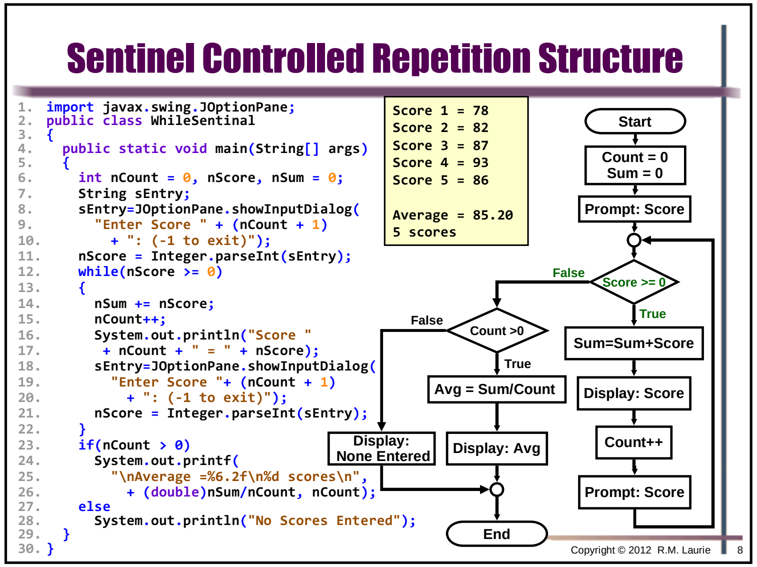
Display: Score

Increment Count

```
Score 1 = 78
Score 2 = 82
Score 3 = 87
Score 4 = 93
Score 5 = 86

Average = 85
```

```java
import javax.swing.JOptionPane;
public class WhileIntro
{
  public static void main(String[] args)
  {
    int nScore, nCount = 1, nSum = 0;
    String sEntry;
    while(nCount <= 5)
    {
      sEntry=JOptionPane.showInputDialog(
        "Enter Score "+nCount+":");
      nScore = Integer.parseInt(sEntry);
      nSum += nScore;
      System.out.println("Score "
        + nCount + " = " + nScore);
      nCount++;
    }
    System.out.println("\nAverage = "
      + nSum/5);
  }
}
```

7



Sentinel Controlled Repetition Structure

```java
import javax.swing.JOptionPane;
public class WhileSentinal
{
  public static void main(String[] args)
  {
    int nCount = 0, nScore, nSum = 0;
    String sEntry;
    sEntry=JOptionPane.showInputDialog(
      "Enter Score " + (nCount + 1)
        + ": (-1 to exit)");
    nScore = Integer.parseInt(sEntry);
    while(nScore >= 0)
    {
      nSum += nScore;
      nCount++;
      System.out.println("Score "
       + nCount + " = " + nScore);
      sEntry=JOptionPane.showInputDialog(
        "Enter Score "+ (nCount + 1)
          + ": (-1 to exit)");
      nScore = Integer.parseInt(sEntry);
    }
    if(nCount > 0)
      System.out.printf(
        "\nAverage =%6.2f\n%d scores\n",
          + (double)nSum/nCount, nCount);
    else
      System.out.println("No Scores Entered");
  }
}
```

```
Score 1 = 78
Score 2 = 82
Score 3 = 87
Score 4 = 93
Score 5 = 86

Average = 85.20
5 scores
```

Start

Count = 0
Sum = 0

Prompt: Score

False

Score >= 0

True

False

Count >0

True

Sum=Sum+Score

Avg = Sum/Count

Display: Score

Display: None Entered

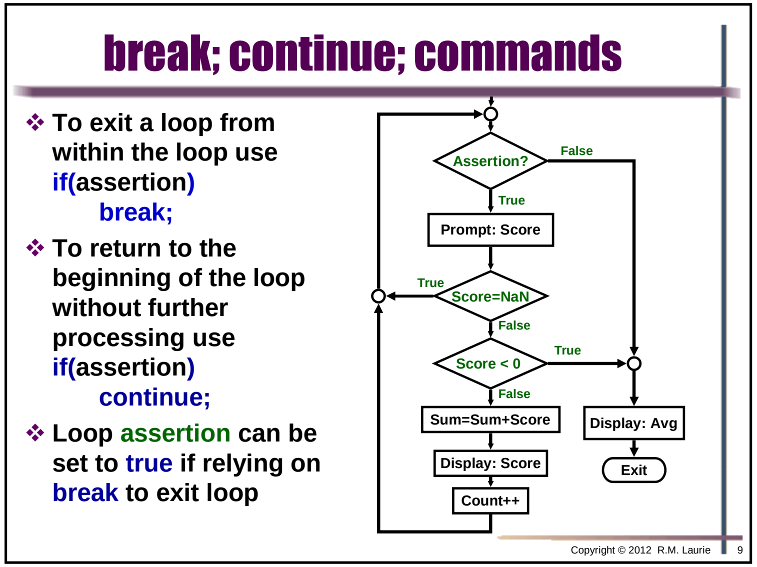
Display: Avg

Count++

Prompt: Score

End

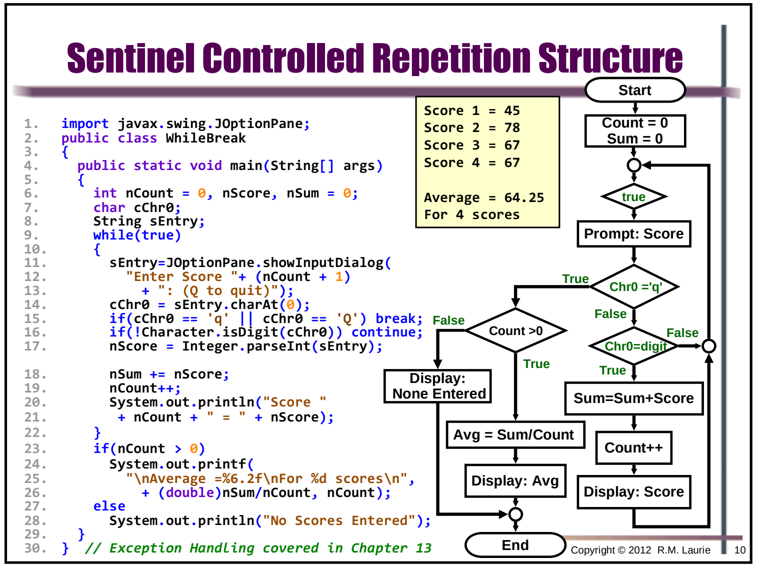
8



break; continue; commands

- To exit a loop from within the loop use **if(assertion)**
  break;
- To return to the beginning of the loop without further processing use **if(assertion)**
  continue;
- Loop **assertion** can be set to **true** if relying on **break** to exit loop

9

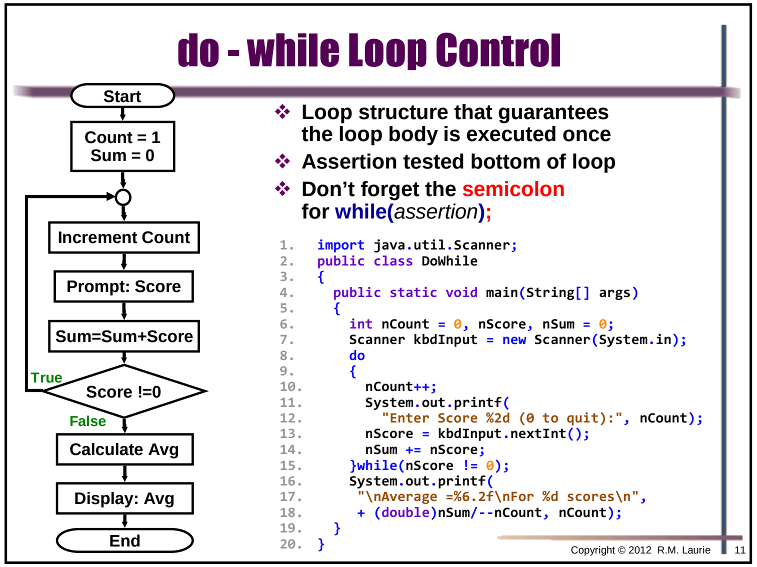


Sentinel Controlled Repetition Structure

```java
import javax.swing.JOptionPane;
public class WhileBreak
{
  public static void main(String[] args)
  {
    int nCount = 0, nScore, nSum = 0;
    char cChr0;
    String sEntry;
    while(true)
    {
      sEntry=JOptionPane.showInputDialog(
        "Enter Score "+ (nCount + 1)
          + ": (Q to quit)");
      cChr0 = sEntry.charAt(0);
      if(cChr0 == 'q' || cChr0 == 'Q') break;
      if(!Character.isDigit(cChr0)) continue;
      nScore = Integer.parseInt(sEntry);

      nSum += nScore;
      nCount++;
      System.out.println("Score "
       + nCount + " = " + nScore);
    }
    if(nCount > 0)
      System.out.printf(
        "\nAverage =%6.2f\nFor %d scores\n",
          + (double)nSum/nCount, nCount);
    else
      System.out.println("No Scores Entered");
  }
}  // Exception Handling covered in Chapter 13
```

```
Score 1 = 45
Score 2 = 78
Score 3 = 67
Score 4 = 67

Average = 64.25
For 4 scores
```

10

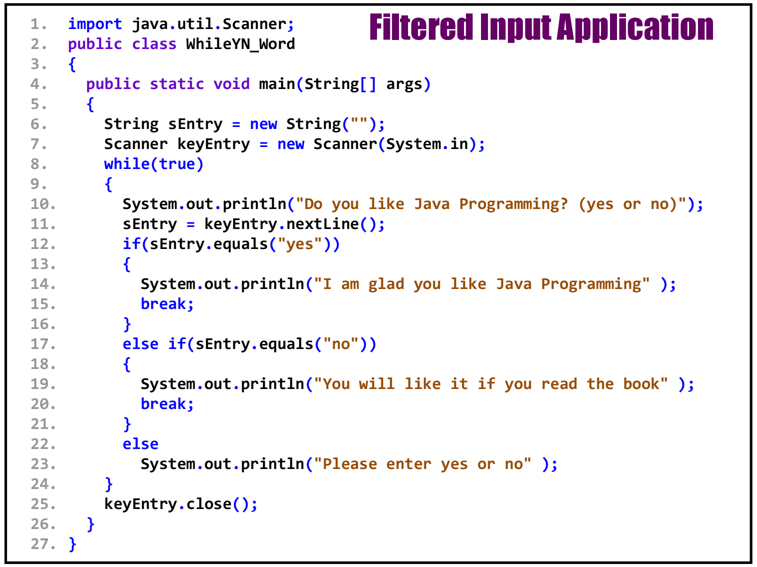


do - while Loop Control

- Loop structure that guarantees the loop body is executed once
- Assertion tested bottom of loop
- Don't forget the semicolon for **while(*assertion*);**

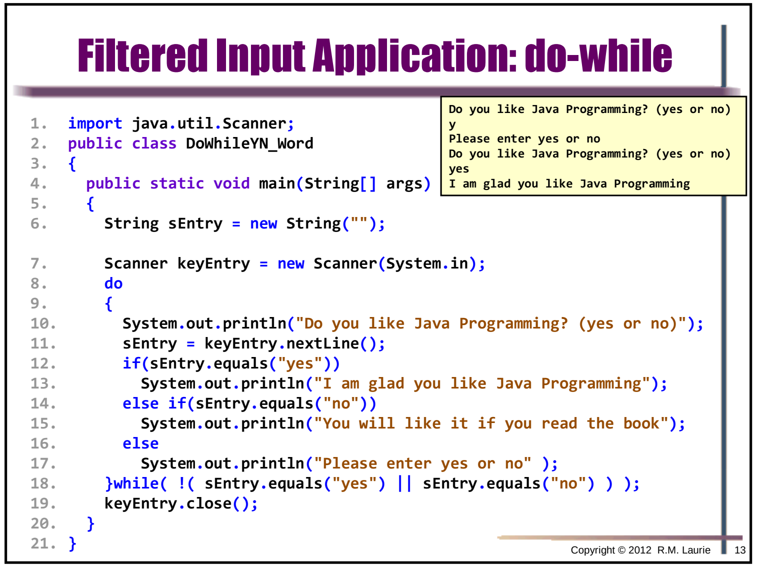
```java
import java.util.Scanner;
public class DoWhile
{
  public static void main(String[] args)
  {
    int nCount = 0, nScore, nSum = 0;
    Scanner kbdInput = new Scanner(System.in);
    do
    {
      nCount++;
      System.out.printf(
        "Enter Score %2d (0 to quit):", nCount);
      nScore = kbdInput.nextInt();
      nSum += nScore;
    }while(nScore != 0);
    System.out.printf(
     "\nAverage =%6.2f\nFor %d scores\n",
     + (double)nSum/--nCount, nCount);
  }
}
```

11



Filtered Input Application

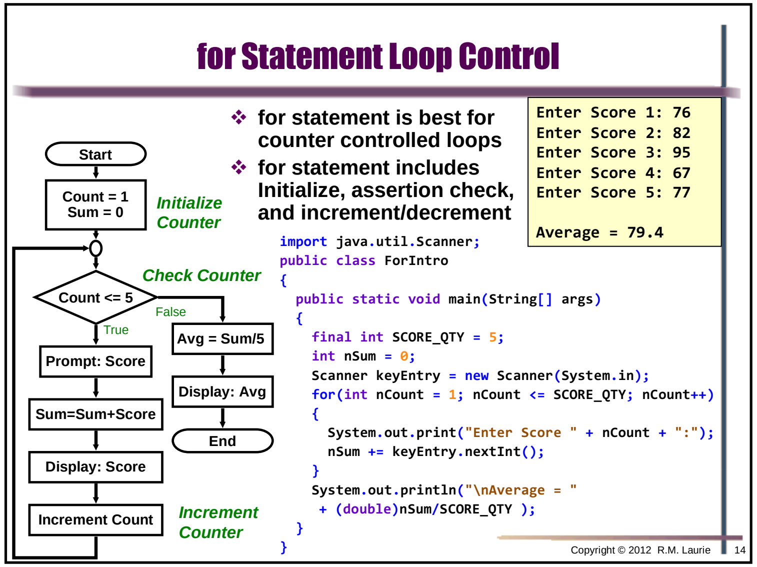
```java
import java.util.Scanner;
public class WhileYN_Word
{
  public static void main(String[] args)
  {
    String sEntry = new String("");
    Scanner keyEntry = new Scanner(System.in);
    while(true)
    {
      System.out.println("Do you like Java Programming? (yes or no)");
      sEntry = keyEntry.nextLine();
      if(sEntry.equals("yes"))
      {
        System.out.println("I am glad you like Java Programming" );
        break;
      }
      else if(sEntry.equals("no"))
      {
        System.out.println("You will like it if you read the book" );
        break;
      }
      else
        System.out.println("Please enter yes or no" );
    }
    keyEntry.close();
  }
}
```



Filtered Input Application: do-while

```java
import java.util.Scanner;
public class DoWhileYN_Word
{
  public static void main(String[] args)
  {
    String sEntry = new String("");

    Scanner keyEntry = new Scanner(System.in);
    do
    {
      System.out.println("Do you like Java Programming? (yes or no)");
      sEntry = keyEntry.nextLine();
      if(sEntry.equals("yes"))
        System.out.println("I am glad you like Java Programming");
      else if(sEntry.equals("no"))
        System.out.println("You will like it if you read the book");
      else
        System.out.println("Please enter yes or no" );
    }while( !( sEntry.equals("yes") || sEntry.equals("no") ) );
    keyEntry.close();
  }
}
```

```
Do you like Java Programming? (yes or no)
y
Please enter yes or no
Do you like Java Programming? (yes or no)
yes
I am glad you like Java Programming
```



for Statement Loop Control

- ❖ for statement is best for counter controlled loops
- ❖ for statement includes Initialize, assertion check, and increment/decrement

Start

Count = 1
Sum = 0

Initialize Counter

Check Counter

Count <= 5

True

False

Prompt: Score

Sum=Sum+Score

Display: Score

Increment Count

Increment Counter

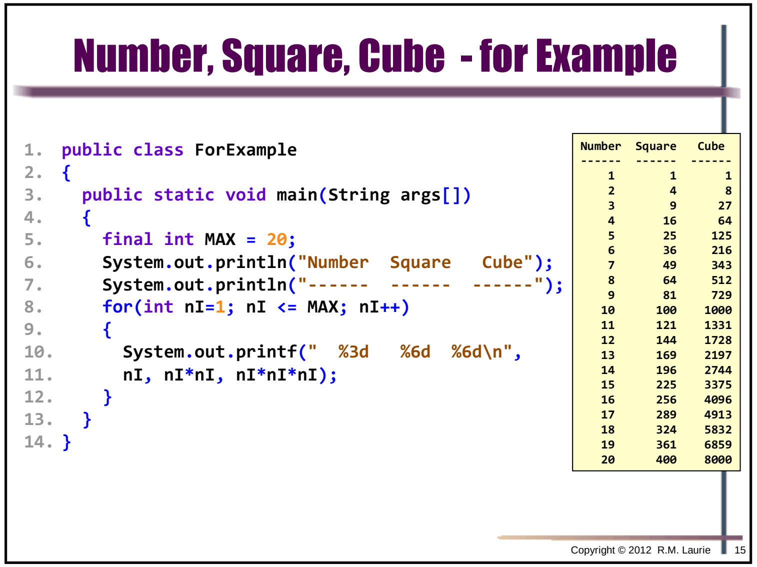
Avg = Sum/5

Display: Avg

End

```java
import java.util.Scanner;
public class ForIntro
{
  public static void main(String[] args)
  {
    final int SCORE_QTY = 5;
    int nSum = 0;
    Scanner keyEntry = new Scanner(System.in);
    for(int nCount = 1; nCount <= SCORE_QTY; nCount++)
    {
      System.out.print("Enter Score " + nCount + ":");
      nSum += keyEntry.nextInt();
    }
    System.out.println("\nAverage = "
     + (double)nSum/SCORE_QTY );
  }
}
```

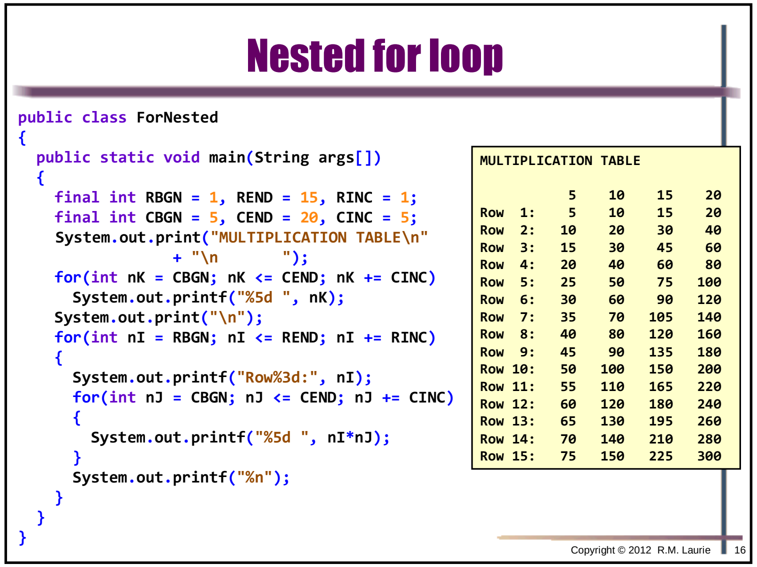
```
Enter Score 1: 76
Enter Score 2: 82
Enter Score 3: 95
Enter Score 4: 67
Enter Score 5: 77

Average = 79.4
```



Number, Square, Cube - for Example

```java
public class ForExample
{
  public static void main(String args[])
  {
    final int MAX = 20;
    System.out.println("Number  Square   Cube");
    System.out.println("------  ------  ------");
    for(int nI=1; nI <= MAX; nI++)
    {
      System.out.printf("  %3d   %6d  %6d\n",
      nI, nI*nI, nI*nI*nI);
    }
  }
}
```

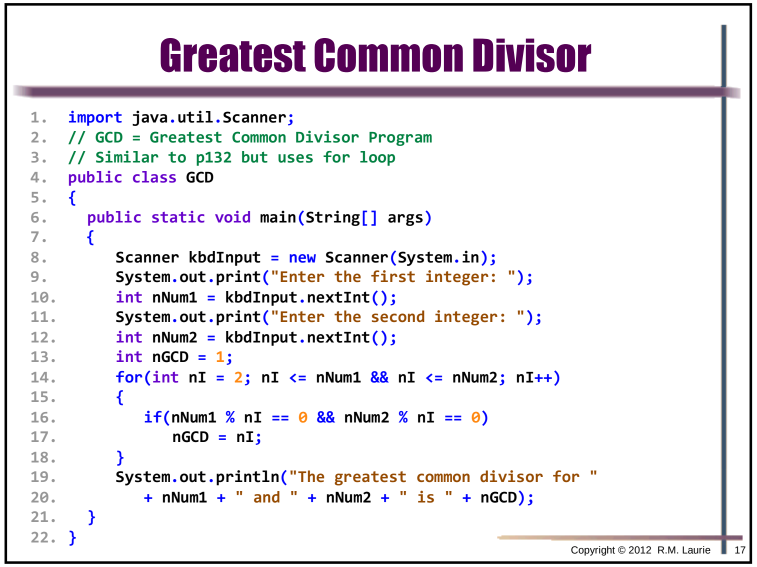
```
Number  Square   Cube
------  ------  ------
     1       1       1
     2       4       8
     3       9      27
     4      16      64
     5      25     125
     6      36     216
     7      49     343
     8      64     512
     9      81     729
    10     100    1000
    11     121    1331
    12     144    1728
    13     169    2197
    14     196    2744
    15     225    3375
    16     256    4096
    17     289    4913
    18     324    5832
    19     361    6859
    20     400    8000
```



Nested for loop

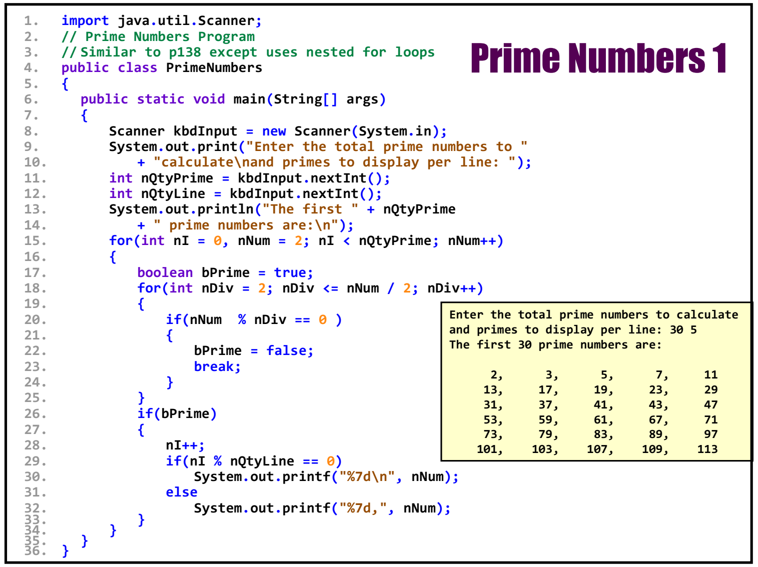
```java
public class ForNested
{
  public static void main(String args[])
  {
    final int RBGN = 1, REND = 15, RINC = 1;
    final int CBGN = 5, CEND = 20, CINC = 5;
    System.out.print("MULTIPLICATION TABLE\n"
                   + "\n         ");
    for(int nK = CBGN; nK <= CEND; nK += CINC)
      System.out.printf("%5d ", nK);
    System.out.print("\n");
    for(int nI = RBGN; nI <= REND; nI += RINC)
    {
      System.out.printf("Row%3d:", nI);
      for(int nJ = CBGN; nJ <= CEND; nJ += CINC)
      {
        System.out.printf("%5d ", nI*nJ);
      }
      System.out.printf("%n");
    }
  }
}
```

```
MULTIPLICATION TABLE

             5     10     15     20
Row  1:      5     10     15     20
Row  2:     10     20     30     40
Row  3:     15     30     45     60
Row  4:     20     40     60     80
Row  5:     25     50     75    100
Row  6:     30     60     90    120
Row  7:     35     70    105    140
Row  8:     40     80    120    160
Row  9:     45     90    135    180
Row 10:     50    100    150    200
Row 11:     55    110    165    220
Row 12:     60    120    180    240
Row 13:     65    130    195    260
Row 14:     70    140    210    280
Row 15:     75    150    225    300
```



Greatest Common Divisor

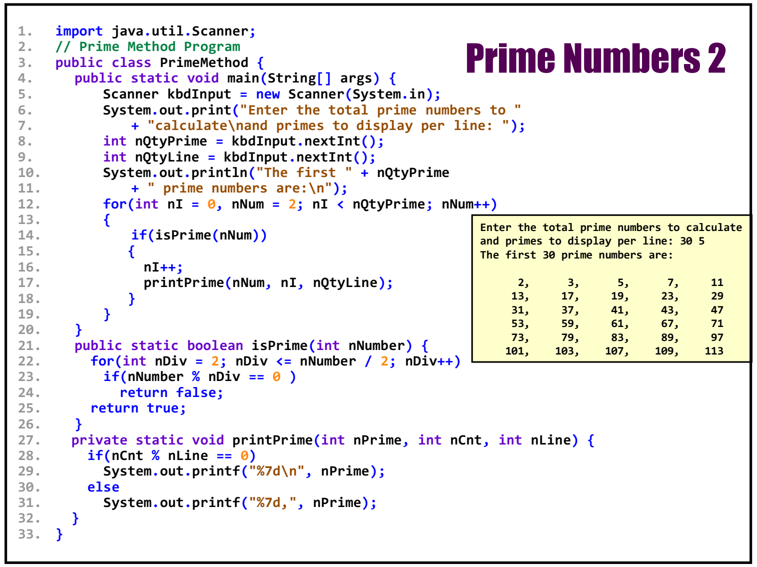
```java
import java.util.Scanner;
// GCD = Greatest Common Divisor Program
// Similar to p132 but uses for loop
public class GCD
{
   public static void main(String[] args)
   {
      Scanner kbdInput = new Scanner(System.in);
      System.out.print("Enter the first integer: ");
      int nNum1 = kbdInput.nextInt();
      System.out.print("Enter the second integer: ");
      int nNum2 = kbdInput.nextInt();
      int nGCD = 1;
      for(int nI = 2; nI <= nNum1 && nI <= nNum2; nI++)
      {
         if(nNum1 % nI == 0 && nNum2 % nI == 0)
            nGCD = nI;
      }
      System.out.println("The greatest common divisor for "
         + nNum1 + " and " + nNum2 + " is " + nGCD);
   }
}
```



Prime Numbers 1

```java
import java.util.Scanner;
// Prime Numbers Program
// Similar to p138 except uses nested for loops
public class PrimeNumbers
{
   public static void main(String[] args)
   {
      Scanner kbdInput = new Scanner(System.in);
      System.out.print("Enter the total prime numbers to "
         + "calculate\nand primes to display per line: ");
      int nQtyPrime = kbdInput.nextInt();
      int nQtyLine = kbdInput.nextInt();
      System.out.println("The first " + nQtyPrime
         + " prime numbers are:\n");
      for(int nI = 0, nNum = 2; nI < nQtyPrime; nNum++)
      {
         boolean bPrime = true;
         for(int nDiv = 2; nDiv <= nNum / 2; nDiv++)
         {
            if(nNum  % nDiv == 0 )
            {
               bPrime = false;
               break;
            }
         }
         if(bPrime)
         {
            nI++;
            if(nI % nQtyLine == 0)
               System.out.printf("%7d\n", nNum);
            else
               System.out.printf("%7d,", nNum);
         }
      }
   }
}
```

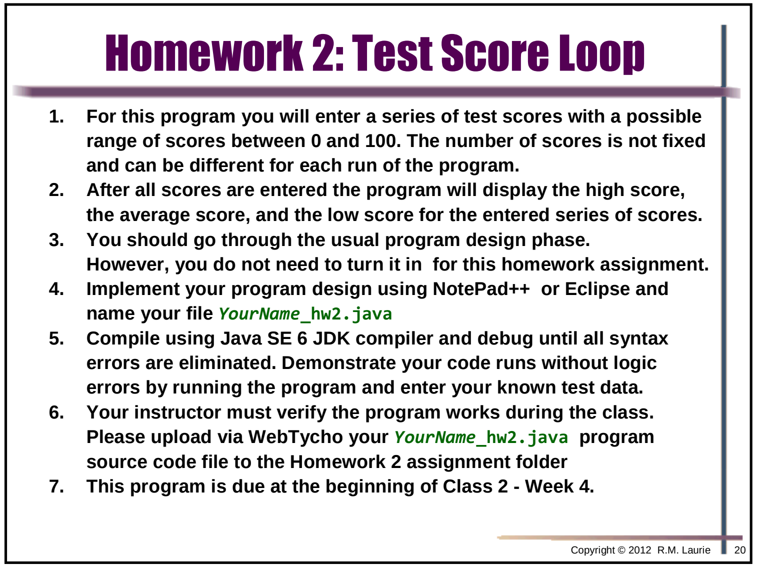
```
Enter the total prime numbers to calculate
and primes to display per line: 30 5
The first 30 prime numbers are:

      2,      3,      5,      7,     11
     13,     17,     19,     23,     29
     31,     37,     41,     43,     47
     53,     59,     61,     67,     71
     73,     79,     83,     89,     97
    101,    103,    107,    109,    113
```



Prime Numbers 2

```java
import java.util.Scanner;
// Prime Method Program
public class PrimeMethod {
   public static void main(String[] args) {
      Scanner kbdInput = new Scanner(System.in);
      System.out.print("Enter the total prime numbers to "
         + "calculate\nand primes to display per line: ");
      int nQtyPrime = kbdInput.nextInt();
      int nQtyLine = kbdInput.nextInt();
      System.out.println("The first " + nQtyPrime
         + " prime numbers are:\n");
      for(int nI = 0, nNum = 2; nI < nQtyPrime; nNum++)
      {
         if(isPrime(nNum))
         {
           nI++;
           printPrime(nNum, nI, nQtyLine);
         }
      }
   }
   public static boolean isPrime(int nNumber) {
     for(int nDiv = 2; nDiv <= nNumber / 2; nDiv++)
       if(nNumber % nDiv == 0 )
         return false;
     return true;
   }
  private static void printPrime(int nPrime, int nCnt, int nLine) {
    if(nCnt % nLine == 0)
      System.out.printf("%7d\n", nPrime);
    else
      System.out.printf("%7d,", nPrime);
  }
}
```

```
Enter the total prime numbers to calculate
and primes to display per line: 30 5
The first 30 prime numbers are:

      2,      3,      5,      7,     11
     13,     17,     19,     23,     29
     31,     37,     41,     43,     47
     53,     59,     61,     67,     71
     73,     79,     83,     89,     97
    101,    103,    107,    109,    113
```



Homework 2: Test Score Loop

1. For this program you will enter a series of test scores with a possible range of scores between 0 and 100. The number of scores is not fixed and can be different for each run of the program.
2. After all scores are entered the program will display the high score, the average score, and the low score for the entered series of scores.
3. You should go through the usual program design phase. However, you do not need to turn it in for this homework assignment.
4. Implement your program design using NotePad++ or Eclipse and name your file *YourName*`_hw2.java`
5. Compile using Java SE 6 JDK compiler and debug until all syntax errors are eliminated. Demonstrate your code runs without logic errors by running the program and enter your known test data.
6. Your instructor must verify the program works during the class. Please upload via WebTycho your *YourName*`_hw2.java` program source code file to the Homework 2 assignment folder
7. This program is due at the beginning of Class 2 - Week 4.