ES 1036 Programming Fundamentals

Control Structures

"There are no secrets to success. It is the result of preparation, hard work, and learning from failure" ~Colin Powell

"Genius is 1% talent and 99% percent hard work..." ~ Albert Einstein

Dr. Quazi M. Rahman, Ph.D, P.Eng., SMIEEE,
Office Location: TEB 361
Email: QRAHMAN@eng.uwo.ca
Phone: 519-661-2111 x81399

Outline

- Flow of Control
- Selection Statements
  - if
  - switch
- Repetition statements
  - while
  - do-while
  - for

Flow of Control

- The order in which statements are executed, is called flow of control.
- There are three basic control structures
  - Sequence – execute statements in sequence (example: all the program examples shown earlier)
  - Selection – choose which statements to execute,
  - Repetition- repeat certain statements.

Selection: the `if` Statement

```
if(expression)
      statement;
```

single statement executed if *expression* is true

```
if (expression) statement;
```

if *expression* and its following statement can be written in one line

```
if(expression)
{
      statement 1;
      statement 2;
      …
      statement n;
}
```

statements inside `{}` are executed if *expression* is true
The `{}` make all the statements appear as a single statement

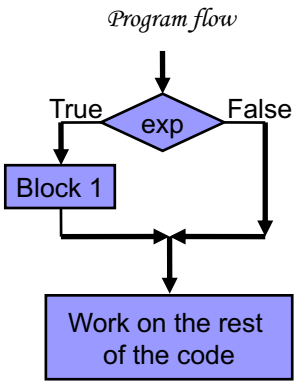
If expression is always evaluated logically (Boolean) as either true or false

Expressions in if-statements

- If-expression is always evaluated logically as either true or false (Boolean type data)
- In general, the expression in the if statement can use two types of operators :
  - Comparison
    - Relational (>, <, >=, <=)
    - Equality (==, !=)
  - Logical (&&, ||, !)
- It can also use other operators
- The expression can be a combination of more than one nested expression or it can be just a single variable without any operator (see an example later)

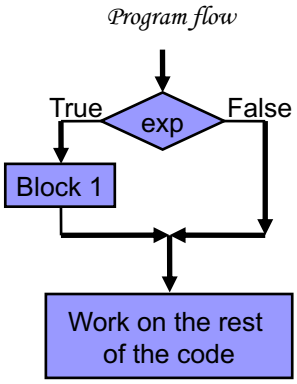
Nested code structure: a code structure where any code-block resides inside another code-block is known as nested code structure. See the example on slide #28.

The if statement – example 1




```
public static void main(String[] args) {
int x = 5,k = 1;
   if (x>0 && k>=1)
       k++;
   if (x<0 || k==1)
   {
      x=x+3;
      k++;
   }
System.out.println("x =  " + x +" and k = "+k);
}
```

Variation of example 1



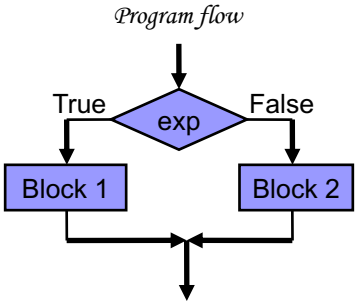

```
public static void main(String[] args) {
int x = 5,k = 1;
boolean b1 = (x>0 && k>=1);
boolean b2 = (x<0 || k==1);
   if (b1)
      k++;
   if (b2)
   {
      x=x+3;
      k++;
   }
System.out.println("x =  " + x +" and k = "+k);
}
```


The `if-else` statement

```
if(expression)
   statement 1;
else
   statement 2;
```

Considered as one C++ statement

```
if(expression)
{
   statement block 1;
}
else
{
   statement block 2;
}
```



if-else statement: Program example

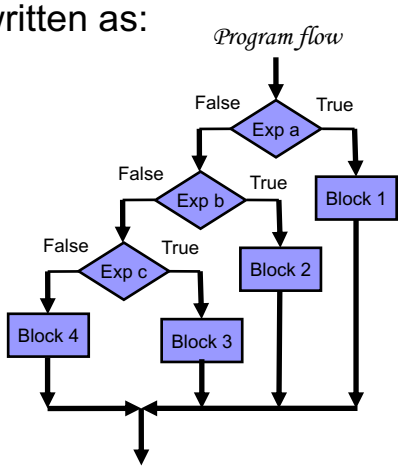
- Problem: Prompt the user to enter a number between 3 and 30 inclusive and print the entered number on the screen. If the number is outside the above range print "out of range". `Assumption: User will not enter any non-integer data.`

```
package quaziTest;
import java.util.Scanner;
 public class MyClass {
   public static void main(String[] args) {
     Scanner input = new Scanner(System.in);
     System.out.print("Enter an integer: ");
     int x =input.nextInt();
     if((x>=3)&&(x<=30))
        System.out.println("You've entered " + x);
     else
        System.out.println("Out of range");
     System.out.println("Good Bye!");
     input.close();
 }
}
```


if-else-if Statement

- The if-else-if statement is written as:

```
if (expression a)
    statement block 1;
else if (expression b)
    statement block 2;
else if (expression c)
    statement block 3;
else
    statement block 4;
```



- This is the most widely used if-structure

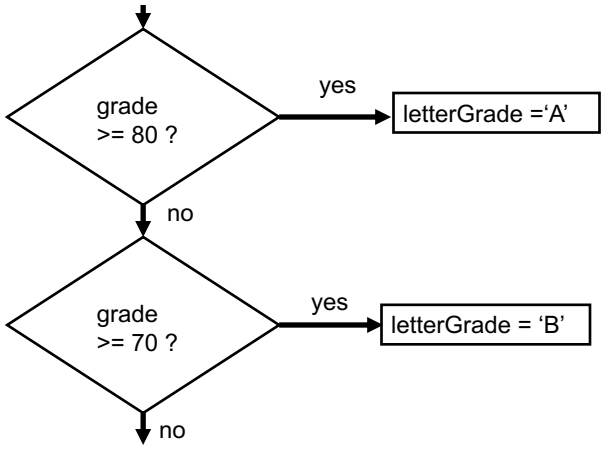
Grading Program Example

- *Consider the following table for assigning marks for a course. Assumption: user will only enter any grade between 0 and 100. Also, no non-integer value will be entered.*

Number Range	Letter Grade
80 or greater	A
70-79	B
60-69	C
50-59	D
Less than 50	F

Grading program

- Flow chart:



```java
//Solution in the main() method
public static void main(String[] args) {
  int grade; char letterGrade;
  Scanner input = new Scanner(System.in);
  System.out.print("Enter your grade: ");
  grade =input.nextInt();
  if(grade >= 80)
     letterGrade = 'A';
  else if (grade >= 70)
     letterGrade = 'B';
  else if (grade >= 60)
     letterGrade = 'C';
  else if (grade >= 50)
     letterGrade = 'D';
  else
     letterGrade = 'F';
System.out.println("Your grade is " + letterGrade);
input.close();
}
```

Homework: Modify the code so that you can display A+ when the grade is more than 89.

Review

1) What are the values of j, k and m?

```java
int x=9, y=7, z=2, k=0, m=0, j=0;
if(x > y)
      if(y>z && y>k)
            k++;
      else
            m++;
else
      j++;
```

a) j is 0, k is 1, m is 0
b) j is 1, k is 0, m is 0
c) j is 0, k is 0, m is 1

Program Examples (Slide 15 – 41)

- Program examples with if, if-else and if-else-if

The `if` statement – homework example

```
public static void main() {
  if (true) {//The if-body will always be executed
   System.out.println("Life is beautiful");
   System.out.println("Don't worry, be happy");

  }
  cout<< "have fun\n";
}
```

The `if` statement – homework examples

Draw the memory diagram and find the output:

Variable name	Content

```
int sum = 0, a = 1, count=0;
if (a < 5){
   ++count;
   sum += a;
}
System.out.println("sum = " + sum +
"count = " + count);
```



```
/*Program Example with != and if-else. The following code is
written to find out whether the entered character is a vowel
or consonant*/
//The main() method is shown below:
public static void main(String[] args) {
  char alphabet;
  Scanner input = new Scanner(System.in);
  System.out.print("Enter a character from English
  Alphabet: ");
  alphabet =input.next().charAt(0);
  if (alphabet != 'a' && alphabet != 'e' &&
      alphabet != 'i' && alphabet != 'o' &&
      alphabet != 'u')
      System.out.println("You entered a
      consonant");
  else
      System.out.println("You entered a vowel");
  input.close();
}/*Note: If you are not sure about the operator precedence, use round bracket
in a long expression*/
```

The `if-else` statement - example

```
/*Draw the flow chart for the following problem: Find out
whether the entered number is even or odd. (It's another
application of % operator). Assumption: User will not
enter any non-integer data*/

public static void main(String[] args) {
  int number;
  Scanner input = new Scanner(System.in);
  System.out.print("Enter an integer number: ");
  number = input.nextInt();
  if (number % 2 == 0)
     System.out.println( number + " is even.");
  else
     System.out.println( number + " is odd.");
 input.close();
}
```



```
/*Homework Example Problem: Write a program that prompts
the user to enter two integers and finds the greater one.
Assumption: User will not enter any non-integer data */
public static void main(String[] args) {
  //declare two integer variables
  int x, y;
  //create a scanner object
  Scanner input = new Scanner(System.in);
  //prompt the user to enter the integer values
  System.out.print("Enter x: "); x = input.nextInt();
  System.out.print("Enter y: "); y = input.nextInt();
  //process, and output the message
  if (x>y)
    System.out.println("x is greater than y");
  else
    System.out.println("y is greater than or equal to x");
input.close();
}
```

Draw the flowchart for the above problem. Modify the code so that it can also check whether the numbers are equal.

```java
/* Example Problem: How to compare two floating point
numbers? See the note at the bottom of slide #86,
lecture Unit 2 */
public static void main(String[] args){
  final double tolerance = 1E-20;//1x10^-20
  double num1 = (1 - 0.9), num2 = 0.1;
  // below, check it for yourself
  System.out.println(Math.abs(num1-num2));
  //now compare formally
  if(Math.abs(num1-num2)<=tolerance)
      System.out.println( "Equal numbers");
  else
      System.out.println("Not equal numbers");
}
```

Note:

- Floating-point literals can also be specified in scientific notation.
- Example: 1.23456e+2, same as 1.23456e2, is equivalent to 123.456, and 1.23456e-2 is equivalent to 0.0123456.
- E (or e) represents an exponent and it can be either in lowercase or uppercase.

Practice problem

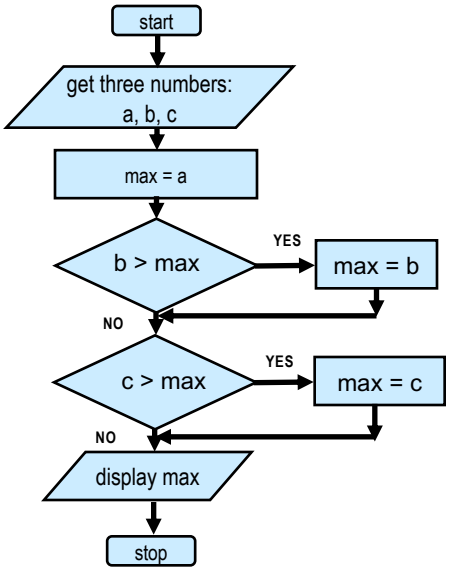
Write a program that prompts the user to enter three integers and finds the maximum of the three

Step 1: Draw the flow-chart.

Programming Tip: Whenever solving any problem that requires to find the maximum (or minimum) number from a set of numbers, declare a suitable variable (max or min) and assign the first number (from the set) to it. Then, compare all the other numbers one at a time with this variable value and refresh this value accordingly.

Note: *Any problem can be solved in many different ways. Try to solve the above problem before looking at the solution available later.*

Flow-chart for Finding max from three entered numbers




```java
//Solution without Math.max(a,b) method
public static void main(String[] args) {
// Step 1: variable declarations
  int a, b, c, max;
  Scanner input = new Scanner(System.in);
  System.out.print("Enter a-value: "); a = input.nextInt();
  System.out.print("Enter b-value: "); b = input.nextInt();
  System.out.print("Enter c-value: "); c = input.nextInt();
  //Step 2: Assign the first entered value to max
  max = a;
  /*Step 3: determine which integer has the maximum value and save it
  to max*/
  if(b>max)
      max = b;
  if(c>max)
      max = c;
  System.out.println("Maximum value: "+max);
  input.close();
}
```

```java
//Solution with Math.max(a,b) method
public static void main(String[] args) {
// Step 1: variable declarations
  int a, b, c, max;
  Scanner input = new Scanner(System.in);
  System.out.print("Enter a-value: "); a = input.nextInt();
  System.out.print("Enter b-value: "); b = input.nextInt();
  System.out.print("Enter c-value: "); c = input.nextInt();
  /*Step 3: determine which integer has the maximum value and save it
  to max*/
  max = Math.max(Math.max(a,b),c);
  System.out.println("Maximum value: "+max);
  input.close();
}
```


Example on why 'if-else if' is NOT always the right way to go

```java
//Output?
public static void main(String[] args) {
      int a = 9, b = 7, c = 4, min = a;
      if (b < min)
            min = b;
      else if (c < min)
            min = c;
      System.out.println(min);
}
```

Memory diagram

Variable name	Content

(a) 7

(b) 9

(c) 4

(d) The code will not compile.

Practice Problem

Write a program that lets the user enter a year and checks whether it is a leap year.

Note: A year is a leap-year if (http://www.mathsisfun.com/definitions/leap-year.html)

- **it is divisible by 4 but not by 100 or**
- **if it is divisible by 400.**

So you can use the following Boolean expression

```
(year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)
```

Note: *Try to solve the above problem before looking at the solution on the next slide*


```
//Solution using boolean-type data
public static void main(String[] args) {
  int year;
  boolean isLeapYear;
  Scanner input = new Scanner(System.in);
  System.out.print("Enter year: ");
  year = input.nextInt();
  isLeapYear = (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
  if(isLeapYear)
     System.out.println(year + " is a leap year");
  else
     System.out.println(year + " is not a leap year");
  input.close();
}
```

Practice Problem: A Simple Math Learning Tool

- This example creates a program to let a first grader practice additions. The program randomly generates two single-digit integers number1 and number2 and displays a question such as “What is 7 + 9?” to the student. After the student types the answer, the program displays a message to indicate whether the answer is true or false.

Note on Math.random() method:

- Math.random() method Generates a double type random number which is greater than or equal to 0.0 and less than 1
- Example 1: Generate a random integer number between 0 and 10

  int x = (int)(Math.random()*11);
- Example 2: Generate a random integer number between 5 and 15

  int x = (int)(Math.random()*11) + 5;
- General Idea: generate a random integer between max and min

  range = (max - min) + 1;

  (int)(Math.random() * range) + min;

The solution

```
public static void main(String[] args) {
    int number1 = (int)(Math.random()*10);
    int number2 = (int)(Math.random()*10);
    // Create a Scanner
    Scanner input = new Scanner(System.in);
    System.out.print(
      "What is " + number1 + " + " + number2 + "?");

    int answer = input.nextInt();

    System.out.println(
      number1 + " + " + number2 + " = " + answer + " is
      " +(number1 + number2 == answer));
    input.close();
  }
```

Practice Problem: Improved Math Learning Tool

- This example creates a program to teach a first grade child how to learn subtractions. The program randomly generates two single-digit integers **number1** and **number2** with **number1 >= number2** and displays a question such as “What is 9 - 2?” to the student. After the student types the answer, the program displays whether the answer is correct.

The solution

```
public static void main(String[] args) {
        // 1. Generate two random single-digit integers
        int number1 = (int)(Math.random() * 10);
        int number2 = (int)(Math.random() * 10);
        // 2. If number1 < number2, swap number1 with number2
        if (number1 < number2) {
          int temp = number1;
          number1 = number2;
          number2 = temp;
        }
        // 3. Prompt the student to answer "what is number1 – number2?"
        System.out.print
          ("What is " + number1 + " - " + number2 + "? ");
        Scanner input = new Scanner(System.in);
        int answer = input.nextInt();
        // 4. Grade the answer and display the result
        if (number1 - number2 == answer)
          System.out.println("You are correct!");
        else {
          System.out.println("Your answer is wrong.");
          System.out.println(number1 + " - " + number2 +
            " should be " + (number1 - number2));
        }
        input.close();
    }
```

Program Example with string-comparison

Enter two city names and print them in alphabetical orders.

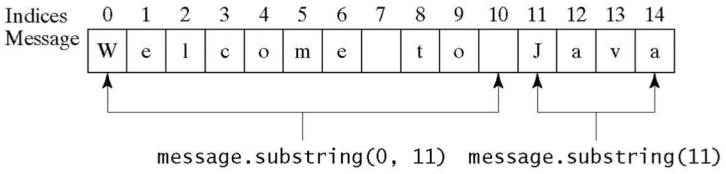
```
public static void main(String[] args) {
  Scanner input = new Scanner(System.in);
  // Prompt the user to enter two cities
  System.out.print("Enter the first city: ");
  String city1 = input.nextLine();
  System.out.print("Enter the second city: ");
  String city2 = input.nextLine();
  if (city1.compareTo(city2) < 0)
  /*see the next slide for compareTo() method description.*/
     System.out.println("The cities in alphabetical order are
     " + city1 + " " + city2);
  else
     System.out.println("The cities in alphabetical order are
     " + city2 + " " + city1);
  input.close();
}
```


For Lab only: Methods to Compare Strings

Method	Description
equals(s1)	Returns true if this string is equal to string s1.
equalsIgnoreCase(s1)	Returns true if this string is equal to string s1; it is case insensitive.
compareTo(s1)	Returns an integer greater than 0, equal to 0, or less than 0 to indicate whether this string is greater than, equal to, or less than s1.
compareToIgnoreCase(s1)	Same as compareTo except that the comparison is case insensitive.
startsWith(prefix)	Returns true if this string starts with the specified prefix.
endsWith(suffix)	Returns true if this string ends with the specified suffix.

For Lab only: Obtaining Substrings

Method	Description
substring(beginIndex)	Returns this string's substring that begins with the character at the specified beginIndex and extends to the end of the string, as shown in Figure 4.2.
substring(beginIndex, endIndex)	Returns this string's substring that begins at the specified beginIndex and extends to the character at index endIndex – 1, as shown in Figure 9.6. Note that the character at endIndex is not part of the substring.



For Lab only: Finding a Character or a Substring in a String (1 of 3)

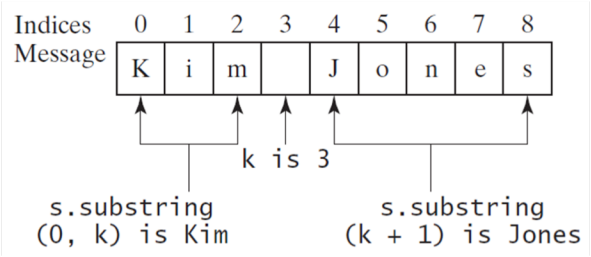
Method	Description
indexOf(ch)	Returns the index of the first occurrence of ch in the string. Returns -1 if not matched.
indexOf(ch, fromIndex)	Returns the index of the first occurrence of ch after fromIndex in the string. Returns -1 if not matched.
indexOf(s)	Returns the index of the first occurrence of string s in this string. Returns -1 if not matched.
indexOf(s, fromIndex)	Returns the index of the first occurrence of string s in this string after fromIndex. Returns -1 if not matched.

For Lab only: Finding a Character or a Substring in a String (2 of 3)

Method	Description
lastIndexOf(ch)	Returns the index of the last occurrence of ch in the string. Returns -1 if not matched.
lastindexOf(ch, fromIndex)	Returns the index of the last occurrence of ch before fromIndex in this string. Returns -1 if not matched.
lastIndexOf(s)	Returns the index of the first occurrence of string s in this string. Returns -1 if not matched.
indexOf(s, fromIndex)	Returns the index of the last occurrence of string s before fromIndex. Returns -1 if not matched.

For Lab only: Finding a Character or a Substring in a String (3 of 3)

```
int k = s.indexOf(' ');
String firstName = s.substring(0, k);
String lastName = s.substring(k + 1);
```



Exercise Problem: Calculate BMI

- Body Mass Index (B M I) is a measure of health on weight. It can be calculated by taking your weight in kilograms and dividing by the square of your height in meters. The interpretation of B M I for people 16 years or older is given below. Write a code to prompt the user to enter her/his weight and height, and calculate the BMI. Base on the BMI score, print he interpreted result based on the following table.

BMI	Interpretation
BMI < 18.5	Underweight
18.5 <= BMI < 25.0	Normal
25.0 <= BMI < 30.0	Overweight
30.0 <= BMI	Obese

Exercise Problem: Computing Taxes

- Personal income tax is calculated based on the filing status and taxable income. There are four filing statuses: single filers, married filing jointly, married filing separately, and head of household. The tax rates are shown below. Write a program to prompt the user to enter his/her taxable income, and then the choice from a menu. Calculate the tax based on the following table.

Marginal Tax Rate	*Single*	*Married Filing Jointly or Qualifying Widow(er)*	*Married Filing Separately*	*Head of Household*
10%	$0 – $8,350	$0 – $16,700	$0 – $8,350	$0 – $11,950
15%	$8,351 – $33,950	$16,701 – $67,900	$8,351 – $33,950	$11,951 – $45,500
25%	$33,951 – $82,250	$67,901 – $137,050	$33,951 – $68,525	$45,501 – $117,450
28%	$82,251 – $171,550	$137,051 – $208,850	$68,526 – $104,425	$117,451 – $190,200
33%	$171,551 – $372,950	$208,851 – $372,950	$104,426 – $186,475	$190,201 – $372,950
35%	$372,951+	$372,951+	$186,476+	$372,951+

Hint on the Exercise Problem on Computing Taxes

```
if (status == 0) {
  // Compute tax for single filers
}
else if (status == 1) {
  // Compute tax for married file jointly
  // or qualifying widow(er)
}
else if (status == 2) {
  // Compute tax for married file separately
}
else if (status == 3) {
  // Compute tax for head of household
}
else {
  // Display wrong status
}
```


Watch out!

```
if (even == true)
  System.out.println(
    "It is even.");
```

(a)

Equivalent

```
if (even)
  System.out.println(
    "It is even.");
```

(b)

Watch out!

```
if (number % 2 == 0)
  even = true;
else
  even = false;
```

(a)

Equivalent

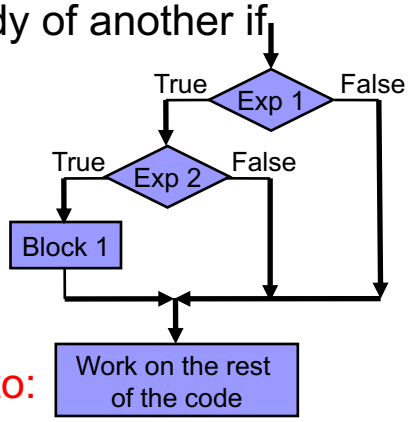
```
boolean even
  = number % 2 == 0;
```

(b)

Nested if statement

- Since the if-statement itself is a statement, it can appear in the body of another if statement

```
if(expression 1)
    if(expression 2)
      statement;
```



The above is equivalent to:

```
if(expression 1 && expression 2)
   statement;
```

Example on Nested-if statement

```
/*Problem: If the user-entered choice is Y or y and user-entered
   number is 5, print "High 5" on the screen*/
public static void main(String[] args) {
  char choice = 'y'; int anyNumber = 0;
  Scanner input = new Scanner(System.in);
  System.out.print("Enter choice: ");
  choice = input.next().charAt(0);
/*Note: toUpperCase(chracter) is a method in java.lang.Chracter
class that coverts any lower-case character to uppercase one.*/
  choice = Character.toUpperCase(choice);
  System.out.print("Enter any number: ");
  anyNumber = input.nextInt();
  //nested-if structure
  if(choice == 'Y'){
       if(anyNumber == 5){
              System.out.println("High " + anyNumber);
       }
  }
  System.out.println("goodbye");
  input.close();
}
```


Revisiting the previous Example

```
/*Draw the flow chart for the following problem: If the user-
   entered choice is Y or y and user-entered number is 5, print
   "High 5" on the screen*/
public static void main(String[] args) {
  char choice = 'y'; int anyNumber = 0;
  Scanner input = new Scanner(System.in);
  System.out.print("Enter choice: ");
  choice = input.next().charAt(0);
/*Note: toUpperCase(chracter) is a method in java.lang.Chracter
class that coverts any lower-case character to uppercase one.*/
  choice = Character.toUpperCase(choice);
  System.out.print("Enter any number: ");
  anyNumber = input.nextInt();
  if(choice == 'Y' && anyNumber == 5){
     System.out.println("High " + anyNumber);
  }
  System.out.println("goodbye");
  input.close();
}
```

FYI: Methods in the Character Class (java.lang.Character)

Method	Description
Character.isDigit(ch)	Returns true if the specified character is a digit.
Character.isLetter(ch)	Returns true if the specified character is a letter.
Character.isLetterOfDigit(ch)	Returns true if the specified character is a letter or digit.
Character.isLowerCase(ch)	Returns true if the specified character is a lowercase letter.
Character.isUpperCase(ch)	Returns true if the specified character is an uppercase letter.
Character.toLowerCase(ch)	Returns the lowercase of the specified character.
Character.toUpperCase(ch)	Returns the uppercase of the specified character.

Caution!

- **Adding a semicolon at the end of an if clause (if-header) is a common mistake.**

Logical Error

```
if (radius >=   0);
{
  area = Math.pow(radius, 2)* Math.PI;
  System.out.println("The area is: " +
                     area);
}
```

Equivalent

Empty Body

```
if (radius >=   0) {
{
  area = Math.pow(radius, 2)* Math.PI;
  System.out.println("The area is: " +
                     area);
}
```

- **This mistake is hard to find, because it is NOT a compilation error, it is a logical error.**
- **This error often occurs when you use the next-line block style (check Unit 2 for this info).**

Caution!

- **Using assignment operator (=) in place of an equality operator (==) in an if-expression is a common mistake.**
- **This mistake results in a compilation error**
  - **E.g.**
```
if (num%2 = 0){//some code}
if (num = 5){//some code}
```


Caution!

What happens if the expression,

`if(0 <= n && n <= 100)` is written as:

```
if(0 <= n <= 100)
```

The second expression is not allowed in Java. It will result in a Compilation error!

Note: Use curly braces for clarity

```
if(x > y)
     if(y < z)
          k++;
     else
          m++;
else
     j++;
```

Better with braces

```
if(x > y)
{
     if(y < z)
          k++;
     else
          m++;
}
else
     j++;
```


Review

2) True or False: The following `println()` method will be executed:

```
int s = 10;
if (s == 0);
  System.out.println("S is zero");
```

a) True
b) False

Selection: `switch` Statement

- *Switch-expression* must result an integer (byte, short, int) or a character and always be enclosed in parentheses.
- The keyword case must be followed by a *literal* (value1, ..., and valueN) and this literal (hard-coded value) must have the same data type as the value of the switch-expression.
- The resulting statements in the case statement are executed when the value in the case statement matches the value of the switch-expression.

```
switch (switch-expression) {
  case value1: statement(s)1;
       break;
  case value2: statement(s)2;
       break;
  …
  case valueN: statement(s)N;
       break;
  default: statement(s);
}
```

Note that value1, ..., and valueN are literals; they CANNOT contain variables, such as x, y etc.

`switch` Statement *cont.*

- The keyword **`break`** is optional, but it should be used at the end of each case in order to terminate the remainder of the switch statement.
- If the break statement is not present, the next case statement will be executed along with the chosen one.
- The default case, which is optional, can be used to perform actions when none of the specified cases matches the switch-expression.
- No block structure is required inside the case statement, if you don't declare any variable inside the case; otherwise block structure is required.
- The case labels can be in any order

```
switch (switch-expression) {
  case value1: statement(s)1;
       break;
  case value2: statement(s)2;
       break;
  …
  case valueN: statement(s)N;
       break;
  default: statement(s);
}
```

Review: the `switch` statement

```
switch(expression)
{
   case literal :
      statement(s);
      break;
   case literal :
      statement(s);
      break;
// default is optional
   default:
      statement(s);
}
```

```
switch(n)
{
   case 1:
      cout << "one";
      break;
   case 2:
      cout << "two";
      break;
   default:
      cout << "error";
}
```



```
//Practice problem for Output-finding

public static void main(String[] args){
     int x = 2, y = 7;
     switch(x){
     case 2:
          x = y--;
     case 3:
          x = x + y;
          break;
     default:
          x = x - 1;
     }
     System.out.println(x+", "+ y);
}
//watch out for the missing breaks!
```

Example: Enter month number to print the name of the month

```
public static void main(String[] args)
{
   System.out.print("Enter month\'s number: ");
   Scanner input = new Scanner(System.in);
/* Assumption: user will enter a valid integer number*/
   int month_number = input.nextInt();
   switch(month_number)
   {
   case 1:
        System.out.println("January");
        break;
   case 2:
        System.out.println("February");
        break;
        //other cases up to 12
   default:
        System.out.println("Invalid month number");
   }
   input.close();
}
```


Practice!

Modify the following code using **switch** statement: (solve it and then check the solution on the next slide)

```
public static void main(String[] args){
      Scanner input = new Scanner(System.in);
      System.out.print("Enter rank: ");
      int rank = input.nextInt();
      if (rank==1 || rank==2)
            System.out.println("Lower division");
      else if (rank==3 || rank==4)
            System.out.println("Upper division");
      else if (rank==5)
            System.out.println("Graduate student");
      else
            System.out.println("Invalid rank");
      input.close();
}
```

Practice Solution!

```
public static void main(String[] args){
{
       Scanner input = new Scanner(System.in);
       System.out.print("Enter rank: ");
       int rank = input.nextInt();
       switch(rank) {
          case 1: case 2:
              System.out.println("Lower division");
              break;
          case 3:
          case 4:
              System.out.println("Upper division");
              break;
          case 5:
              System.out.println("Graduate student");
              break;
         default:
              System.out.println("Invalid rank");
       input.close();
       }
}
```


Practice: Switch structure with character value

```
public static void main(String[] args){
       Scanner input = new Scanner(System.in);
       System.out.print("Enter choice: ");
       char choice = input.next().charAt(0);
       choice = Character.toUpperCase(choice);
       switch(choice)
       {
         case 'A':
              System.out.println("Choice A is entered");
              break;
         case 'B':
              System.out.println("Choice B is entered");
              break;
         case 'C':
              System.out.println("Choice C is entered");
              break;
         default:
              System.out.println("Invalid choice");
       }//end switch (choice)
       input.close();
}
```

Practice Problem: Chinese Zodiac

- Write a program that prompts the user to enter a year and displays the animal for the year.

year % 12 =
0: monkey
1: rooster
2: dog
3: pig
4: rat
5: ox
6: tiger
7: rabbit
8: dragon
9: snake
10: horse
11: sheep


```
//The solution
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    //Input the year
    System.out.print("Enter a year: ");
    int year = input.nextInt();
    //find the animal
    switch (year % 12) {
      case 0: System.out.println("monkey"); break;
      case 1: System.out.println("rooster"); break;
      case 2: System.out.println("dog"); break;
      case 3: System.out.println("pig"); break;
      case 4: System.out.println("rat"); break;
      case 5: System.out.println("ox"); break;
      case 6: System.out.println("tiger"); break;
      case 7: System.out.println("rabbit"); break;
      case 8: System.out.println("dragon"); break;
      case 9: System.out.println("snake"); break;
      case 10: System.out.println("horse"); break;
      case 11: System.out.println("sheep"); break;
    }
    input.close();
  }
```

Review

3) True or False: In Java, a relational expression is always evaluated logically as either true or false.

a) True
b) False

Review

4) The _____ statement will execute a group of statements if the test expression is true, or it'll execute a different group of statements if the expression is false.

a) if
b) if/else
c) switch
d) None of the above

Review

5) A trailing _____ placed at the end of "if/else if" statement provides a default action when none of the "if/else if"s have true expressions.

a) if
b) break
c) else
d) exit
e) None of these

Review

6) True or False: The following test expression in the *if* statement checks if the variable child is in the range between 3 and 12 inclusive.

```
if (child >= 3 || child <= 12)
        {statement(s);}
```

a) True
b) False

Review

7) If a switch statement has no _____ statements, the program "falls through" all of the statements below the one with the matching case-expression.

a) break
b) exit
c) switch
d) else
e) None of the above

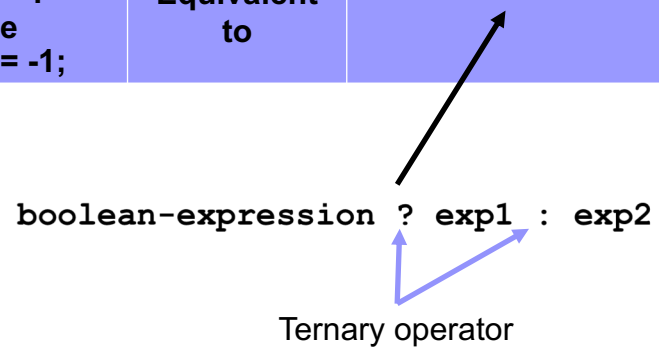
Review

8) A switch statement branches to a particular block of code depending on the value of a floating-point variable or constant.

a) True
b) False

FYI: Conditional Expressions with Ternary operator

if (x > 0)
  y = 1
else
  y = -1;
	Equivalent to	y = (x > 0) ? 1 : -1;
		(boolean-expression) ? expression1 : expression2



FYI: Conditional Expressions with Ternary operator (another example)

```
if(num%2 == 0)
    System.out.println(num + " is even.");
else
    System.out.println(num + " is odd.");
```

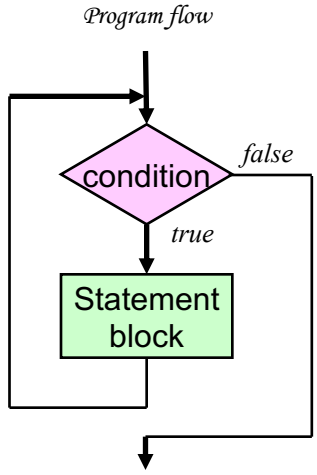
Equivalent to

```
System.out.println((num%2==0)? + num+"is even" : + num +" is odd");
```

Repetition structures

- while
- do-while
- for

Repetition: While Loop



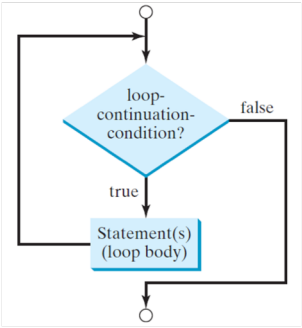

```
while (condition)
  statement;

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

Basic characteristics of *while* Loops

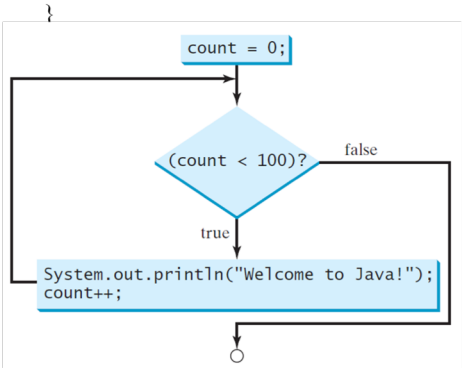
- Loop repetition condition is tested before the body of the loop.
- Body of the loop will be executed 0 (does not execute at all) or more times.
- Doesn't require block structure [braces ({ })] if the body of the loop contains only one statement, but it is a good practice to have them (always).
- Useful when the exact number of repetitions is unknown

While loop with example

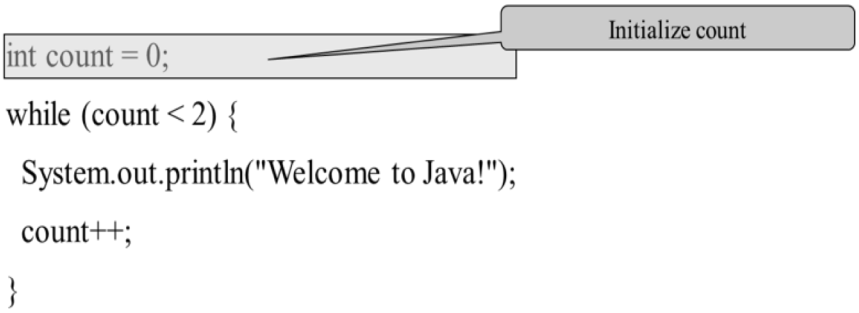
```
while (loop-continuation-condition) {
  // loop-body;
  Statement(s);
}
```



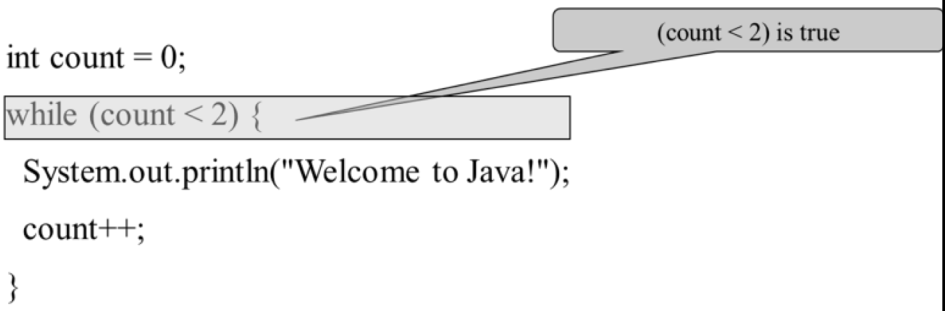

```
int count = 0;
while (count < 100) {
  System.out.println("Welcome to Java!");
  count++;
}
```



Trace While loop (1 of 9)



Trace While loop (2 of 9)



Trace While loop (3 of 9)

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

Print Welcome to Java

Trace While loop (4 of 9)

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

Increase count by 1
count is 1 now

Trace While loop (5 of 9)

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

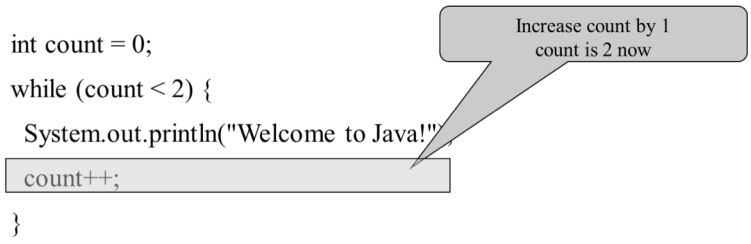
(count < 2) is still true since count is 1

Trace While loop (6 of 9)

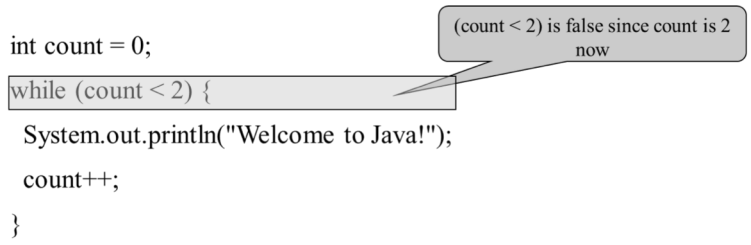
```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

Print Welcome to Java

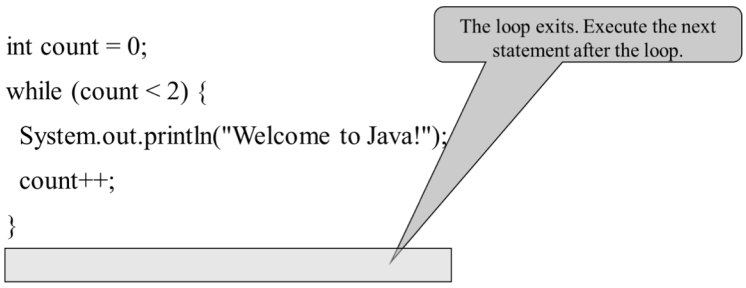
Trace While loop (7 of 9)



Trace While loop (8 of 9)



Trace While loop (9 of 9)



Practice!

```
public static void main(String[] args)
{
	int n = 4;
	while(n>0)
	{
		System.out.println(n);
		n--;
	}
	System.out.println("Outside while, n = " + n);
}
```

Memory diagram:

Variable name	Content

The Output:

Typical Use of *while* Loops

- Infinite loop: if the loop repetition condition is always true, an endless loop will result.
- sometimes Infinite loops indicate an error in a program.

```
public static void main(String[] args)
{
   int a = 5;
   while(a < 10) {
      System.out.println("a = " + a);
   }
   System.out.println("bye");
}
```


Application of *infinite* loop

- Infinite loops are vey helpful in writing a menu-driven program, where one of menu choices will be used to break the loop. (See the slides on break and continue at the end of this unit)
- To incorporate any condition in a loop, it is very convenient to use an infinite-loop, and then based on the condition, break the loop using break-statement (see the example on the next slide).

Breaking an infinite-loop with a conditional break-statement

```
/* Problem: Write a code to add numbers. Ask the user if s/he wants to
continue adding numbers or not. If the user enters 'n', print the
result and terminate the program*/

public static void main(String[] args){
       double sum = 0, num; char choice = 'y';
       Scanner input = new Scanner(System.in);
       while(true){      //infinite loop
              System.out.print("Enter a number: ");
              num = input.nextDouble();
              sum = sum + num;
              System.out.print("Do you want to continue? ");
              choice = input.next().charAt(0);
              if(Character.toUpperCase(choice) == 'N')
                     break;
              //break-statement inside a loop, breaks the loop
       }
       System.out.println("Sum = " + sum); input.close();
}
```


Typical Use of *while* Loops

- "Validation" loop: the loop repetitively checks whether the user has input a valid data (set by the loop test expression) or not
- When the user enters a valid input, the looping ends and the program continues from the statement that follows the loop structure.

Example on *Validation* loop

Problem: Prompt the user to enter an integer number between 3 and 30 inclusive, and after validating the number, print the entered number on the screen. Assumption: user will not enter any non-integer value.

Note: for validating any data input, write the expression based on the requirement and, logically invert the whole expression inside the Loop-condition

```
public static void main (String[] xyz){
   System.out.print("Enter an integer: ");
   Scanner input = new Scanner(System.in);
   int x = input.nextInt();
   while(!((x>=3)&&(x<=30))){//same as: while((x<3)||(x>30))
        System.out.println("Invalid number!");
        System.out.print("Enter an integer between 3 and 30: ");
        x = input.nextInt();
   }
   System.out.println("You've entered " + x);
   input.close();
}
```


Example on *Validation* loop

- Validating characters: the following code validates that the user enters `y` or `n` as his/her choice

```
public static void main (String[] xyz){
       System.out.print("Enter your choice: ");
       Scanner input = new Scanner(System.in);
       char choice = input.next().charAt(0);
       choice = Character.toLowerCase(choice);
       while(!(choice=='y' || choice =='n')) {
       //same as:   while(choice!='y' && choice !='n')
              System.out.print("Invalid choice! enter again: ");
              choice = input.next().charAt(0);
              choice = Character.toLowerCase(choice);
       }
       System.out.println("you entered " + choice);
       input.close();
}
```

Practice Problem: Repeat Addition Until Correct

- (Revisiting the addition problem for the first grader) Write a java program to prompt the user to enter an answer for a question on addition of two single digits. Using a loop, let the user enter a new answer until it is correct. Once the correct answer is displayed, let the user know, after how many tries the correct answer was found.


```
//The solution
public static void main(String[] args) {
 //declare and initiliaze the variables and objects
 Scanner input = new Scanner(System.in);
 int number1 = (int)(Math.random() * 10);
 int number2 = (int)(Math.random() * 10);
 int counter = 1; //count the number of tries
 System.out.print( //ask the question
 "What is " + number1 + " + " + number2 + "? ");
 int answer = input.nextInt();
 //get the answer and find out whether it is correct
 while (number1 + number2 != answer) {
  System.out.print("Wrong answer. Try again. What is "
  + number1 + " + " + number2 + "? ");
  answer = input.nextInt();
  counter++; //increment the counter
}
//print the result
System.out.println("You got it in "+counter+" tries.");
input.close();
}
```

Practice Problem: Guessing Numbers

- Write a program that randomly generates an integer between **0** and **100**, inclusive. The program prompts the user to enter a number continuously until the number matches the randomly generated number. For each user input, the program tells the user whether the input is too low or too high, so the user can choose the next input intelligently.


```
//The solution
public static void main(String[] args) {
 Scanner input = new Scanner(System.in);
 // Generate a random number to be guessed
 int number = (int)(Math.random() * 101);
 System.out.println("Guess a magic number between 0 and 100");
 int guess = -1;
 while (guess != number) {
 // Prompt the user to guess the number
  System.out.print("\nEnter your guess: ");
  guess = input.nextInt();
  if (guess == number)
    System.out.println("Yes, the number is " + number);
  else if (guess > number)
    System.out.println("Your guess is too high");
 else
    System.out.println("Your guess is too low");
 } // End of loop
input.close();
}/*Modify this code to find out how many attempts the
user made to get to the correct answer.*/
```

For lab only: How to calculate the execution time of your code

- To calculate the execution time of your code, use the *currentTimeMillis*() method from the System class. E.g.,

```
public static void main(String[] args) {
long startTime = System.currentTimeMillis();

//your code

long endTime = System.currentTimeMillis();
long testTime = endTime - startTime;

System.out.println("\nCode execution time is " +
testTime / 1000 + " seconds\n");
}
```


Exercise Problem

- Write a java program that generates five subtraction questions and reports the number of the correct answers after a student answers all five questions. Your program should report the execution time (in seconds) of your code.

Sample output of the exercise (You can add your name, and other info at the top):

```
What is 8 - 3? 5
You are correct!

What is 9 - 0? 0
Your answer is wrong.
9 - 0 should be 9

What is 6 - 6? 0
You are correct!

What is 9 - 5? 4
You are correct!

What is 9 - 6? 3
You are correct!

Correct count is 4
Test time is 17 seconds

8-3=5 correct
9-0=0 wrong
6-6=0 correct
9-5=4 correct
9-6=3 correct
```


Watch out!

- Don't use floating-point values for equality checking in a loop control. Since floating-point values are approximations for some values, using them could result in imprecise counter values and inaccurate results. Consider the following code for computing 1 + 0.9 + 0.8 + ... + 0.1:

```
double item = 1; double sum = 0;
while (item != 0) { // No guarantee item will be 0
  sum += item;
  item -= 0.1;
}
System.out.println(sum);
```

- One of the safe route: Use the idea of tolerance

Commonly Asked questions

- How to validate an integer/floating point data? (for example: if you enter a character for any numerical data entry, how can the code may ask you to correct it, instead of generating a run-time error)
  - Discussed Later

Homework on Validation loop

- Write the code for the problem given in slide #10 after removing the assumption "*the user will only enter any data between 0 and 100*" there.
- Prompt the user to enter an integer between 3 and 30 inclusive and validate this input.
- Prompt the user to enter an integer between 3 and 30 inclusive which can be divisible by 2 and 7. Validate this input and print the validated integer on to the screen
- Validate that a user-entered number is an odd number

Typical Use of *while* Loops

- Sentinel loop: A chosen value, based on the program's requirement, called the sentinel, is used to signal an end to data entry.


```
/*write a code to enter integer values from the keyboard and find the sum of
those numbers before the user enters zero. The program should let the user
input the integer numbers until the user enters zero. Once the user enters
zero, the program will display the sum of all the entered number.*/
// Here the sentinel has been chosen to be zero
public static void main (String[] xyz){
      System.out.print("Enter an int value (enter 0 to exit): ");
      Scanner input = new Scanner(System.in);
      int data = input.nextInt();
      // Keep reading data until the input is 0
      int sum = 0;
      while (data != 0)
      {
            sum += data;
            // Read the next data
            System.out.print("Enter an int value (enter 0 to exit): ");
            data = input.nextInt();
      }
      System.out.print("The sum is " + sum);
      input.close();
}
```

A letter as a Sentinel – example 2

```
public static void main(String[] xyz) {
      int number;
      char again = 'y';
      Scanner input = new Scanner(System.in);
      while (again == 'y') {
            System.out.print("Enter an integer: ");
            number = input.nextInt();
            System.out.println("The square of " +
            number + " is " + Math.pow(number, 2));
            System.out.print("Try again?(y/n) ");
            again = input.next().charAt(0);
            again = Character.toLowerCase(again);
      } /* end while loop */
      System.out.println("good-bye");
      input.close();
}
```


Example: Calculating x^y

```
public static void main(String[] xyz) {
      double result = 1, base, power, n;
      Scanner input = new Scanner(System.in);
      System.out.print("Enter x value : ");
      base = input.nextDouble();
      System.out.print("Enter y value : ");
      power = input.nextDouble();
      n = power;
      while (n > 0) { // x^y means multiply x, y-times
            result = result * base;
            n--;
      }
      System.out.print(base + "^" + power + " = " +
      result);
      input.close();
}
```

Caution!

- Adding a semicolon at the end of a while header is a common mistake.

Logical Error

```
while (radius >= 0);
{
  area = radius * radius * PI;

}
```

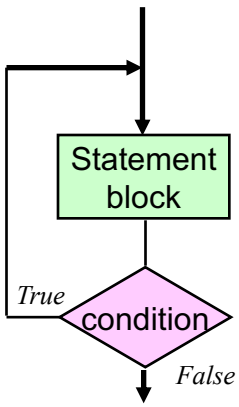
Equivalent

Empty Body

```
while (radius >= 0) { }
{
  area = radius * radius * PI;

}
```

  - This mistake is hard to find, because it is NOT a compilation error, it is a logical error.
  - This error often occurs when you use the next-line block style.

Repetition: The `do/while` statement




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

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

Basic characteristics of *do while* Loops

- Loop repetition condition is tested after the body of the loop.
- Body of the loop will be executed 1 or more times.
- Doesn't require block structure [braces ({ })] if the body of the loop contains only one statement, but it is a good practice to have them (always).
- Useful when at least one repetition of the loop is required.

Validation Example with *do-while*

Problem: Prompt the user to enter a number between 3 and 30 inclusive, and after validating the number, print the entered number on the screen.
Assumption: the user will not enter any non-integer data.

```
public static void main(String args[]){
    int x;
    do
        {
        System.out.println("Enter an integer between 3 and 30:" );
        Scanner input = new Scanner(System.in);
        x = input.nextInt();
    }while(!((x>=3)&&(x<=30)));
    System.out.println("You've entered: "+ x);
    input.close();
}
```

Validation Example(2) with *do-while*

```
//Problem: revisiting the previous problem to make it more user-friendly.
public static void main(String args[]){
    int x;
    do
        {
        System.out.println("Enter an integer between
                            3 and 30:" );
        Scanner input = new Scanner(System.in);
        x = input.nextInt();
        if(!((x>=3)&&(x<=30)))
            System.out.println("Invalid Entry!");
    }while(!((x>=3)&&(x<=30)));
    System.out.println("You've entered: "+ x);
input.close();
}
```


Home work problem on Validation with do-while

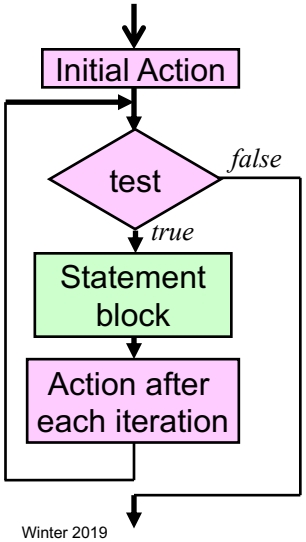
- Example: Write a program to display a menu that contains three numbered operations and prompt the user to select a correct operation-number.

```
public static void main(String args[]){
    int choice;
    System.out.println("***        Main Menu       ***" );
    System.out.println("    (1) Operation one");
    System.out.println("    (2) Operation two");
    System.out.println("    (3) Operation three");
    System.out.println("*****************************" );

    // Prompt the user to enter a number between 1 and 3
    Scanner input = new Scanner(System.in);
    do
        {
        System.out.print("Enter any number between 1 and 3: " );
        choice = input.nextInt();
    }while(!((choice>=1)&&(choice<=3)));// data validation
    System.out.flush();// clears the output-screen
    System.out.print("Performing operation " + choice);
    input.close();
}
```


Evaluating a series expression using loops

- Always initialize a variable that will hold on to the result (for summation the initialized value will be zero and for multiplication it will be 1)
- Find out the following:
  - Start point
  - step size of the series
  - End point
- Find the expression based on the step size
- In each loop update the variable, which was declared and initialized in the first step
- Print the result outside the loop
- Use any loop (while, do-while or for) of your choice
- Note: When a division-operation is present in the series expression, be very careful about data type where casting might be required for logically correct output.

Repetition: The `for` statement




```
for(init; test; inc/dec)
     statement;
for(init; test; inc/dec)
{
     statement;
     statement;
}
```

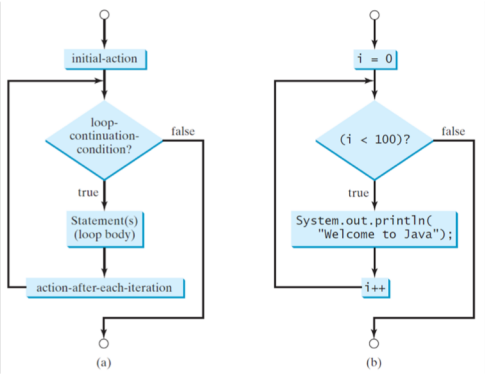

Basic characteristics of *for* Loops

- Loop repetition condition is tested before the body of the loop.
- Body of the loop will be executed 0 or more times.
- Doesn't require block structure [braces ({ })] if the body of the loop contains only one statement.
- Useful when exact number of repetition is known

for-loop with examples

```
for (initial-action; loop-
    continuation-condition; action-
    after-each-iteration) {
  // loop body;
  Statement(s);
}
```

```
int i;
for (i = 0; i < 100; i++) {
  System.out.println(
    "Welcome to Java!");
}
```



Trace for Loop (1 of 10)

Declare i

```
int i;
for (i = 0; i < 2; i++) {
  System.out.println(
    "Welcome to Java!");
}
```

Trace for Loop (2 of 10)

```
int i;
for (i = 0; i < 2; i++) {
  System.out.println(
    "Welcome to Java!");
}
```

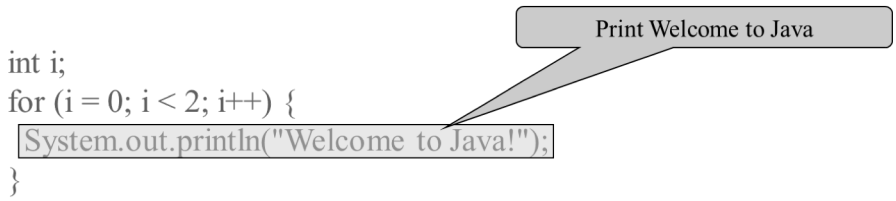
Execute initializer
i is now 0

Trace for Loop (3 of 10)

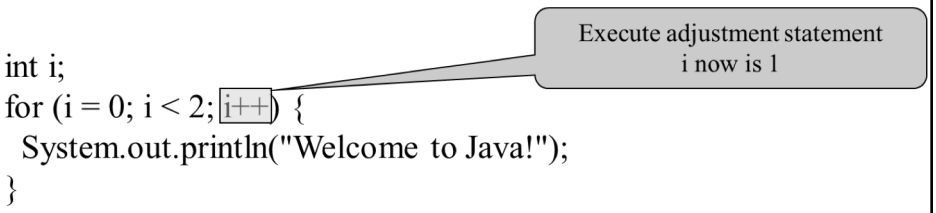
```
int i;
for (i = 0; i < 2; i++) {
  System.out.println( "Welcome to Java!");
}
```

(i < 2) is true
since i is 0

Trace for Loop (4 of 10)



Trace for Loop (5 of 10)



Trace for Loop (6 of 10)

```
int i;
for (i = 0; i < 2; i++) {
  System.out.println("Welcome to Java!");
}
```

(i < 2) is still true since i is 1

Trace for Loop (7 of 10)

```
int i;
for (i = 0; i < 2; i++) {
  System.out.println("Welcome to Java!");
}
```

Print Welcome to Java

Trace for Loop (8 of 10)

```
int i;
for (i = 0; i < 2; i++) {
  System.out.println("Welcome to Java!");
}
```

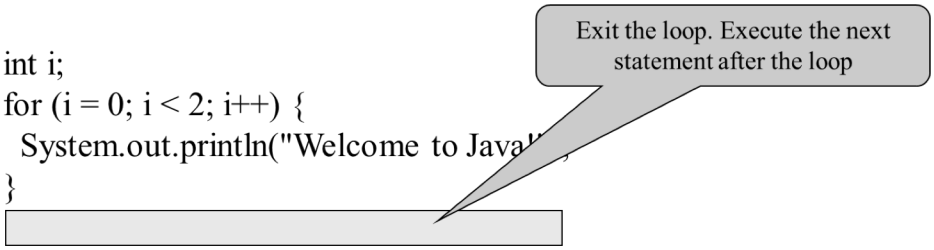
Execute adjustment statement
i now is 2

Trace for Loop (9 of 10)

```
int i;
for (i = 0; i < 2; i++) {
  System.out.println("Welcome to Java!");
}
```

(i < 2) is false
since i is 2

Trace for Loop (10 of 10)



The `for` statement: Examples

```
//sum integers from 1 to 10
int sum =0;
for(int i=1; i<=10; i++)
     sum = sum + i;

//sum odd integers from 1 to 10
int sum =0;
for(int i=1; i<=10; i+=2)
     sum = sum + i;
```

Practice!

Determine the number of times that each of the following `for` loops is executed.

```
for (int k=3; k<=10; k++)
   System.out.println("do nothing");

for (int k=3; k<=10; ++k)
{
   System.out.println("do nothing");
   k=k+2;
}

for (int count=-2; count<=10; count++)
   System.out.println("do nothing");
```


for Loop in series summation: $y = \sum_{i=1}^{4} x_i = x_1 + x_2 + x_3 + x_4$

```
/*Algorithm/Programming idea: for series summation,
declare a variable to hold on to the result and assign 0
to it. In a loop keeping adding the numbers to this
variable and store the result in this variable*/

public static void main(String args[]) {
    Scanner input = new Scanner(System.in);
    int i,y = 0,x;
    System.out.println("Add 4 integers and display the
result:\n\n");
    for (i = 1; i < 5; i++) {
        System.out.print("Enter integer number "+ i +":");
        x = input.nextInt();
        y += x;
    }
    System.out.println("\ny=" + y);
    input.close();
}
```

Typical Uses of Loops: Maximum or Minimum(v.1)

```
/*Find the max and min numbers from 5 entered
numbers. Read the programming tip from slide 22*/
public static void main(String args[]){
  Scanner input = new Scanner(System.in);
  int min, number, i, max;
  System.out.print("Enter integer 1:");
  number = input.nextInt();
  min = number;
  max = number;
  for (i = 2; i <= 5; i++){
    System.out.print("Enter integer: " + i + ":" );
    number = input.nextInt();
    min = Math.min(number, min);
    max = Math.max(number, max);
  }/* end for loop */
  System.out.println("\nThe smallest number is " +
  min  + ", and the largest number is " + max);
 input.close();
}
```


Typical Uses of Loops: Maximum or Minimum (v.2)

```
public static void main(String args[]){
  Scanner input = new Scanner(System.in);
  int min = 0, number, i, max = 0;
  for (i = 1; i <= 5; i++)
  {
    System.out.print("Enter integer " + i + ": " );
    number = input.nextInt();
    if(i == 1) {
      min = number;
      max = number;
    }
    else {
      min = Math.min(number, min);
      max = Math.max(number, max);
    }
  }/* end for loop */
  System.out.println("\nThe smallest number is " + min + ", and
  the largest number is " + max);
  input.close();
}
```

for Loop in series multiplication: $y = \prod_{i=1}^{4} x_i = x_1 \times x_2 \times x_3 \times x_4$

```
/*Algorithm/Programming idea: for series multiplication,
declare a variable to hold on to the result and assign
1 to it. In a loop keeping multiplying the numbers to
this variable and store the result in this variable*/
public static void main(String args[]){
  Scanner input = new Scanner(System.in);
  int i, number; double product = 1;
  for (i = 1; i <= 4; i++)
  {
    System.out.print("Enter number " + i + ": " );
    number = input.nextInt();
    product *= number;
  }/* end for loop */
  System.out.println("\y = " + product);
  input.close();
}
```


Practice problems with series operations (try to solve with while, do-while and for structures)

- Write a Java program to calculate the value of **e** for an integer value of **n** (n>=0) where **e** is given by the following series (factorial (!)operation has been demonstrated later):

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \ldots\ldots + \frac{1}{n!}$$

- Write a Java program to numerically calculate the value of y given by,

$y = x^2 + px + 2p^2$

where,

$-\infty < x < \infty;$

$p = \prod_{i=1}^{N} \frac{1}{i^2} = 1 \text{ x } \frac{1}{4} \text{ x } \frac{1}{9} \text{ x } \ldots\ldots \text{ x } \frac{1}{N^2}$; (N is a positive non-zero integer)

Typical Uses of Loops: Nested Loops

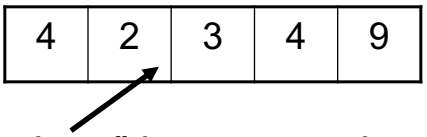
```
public static void main(String args[])
{
    int i, j;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 2; j++)
        {
          System.out.println(
          "outer loop i = " + i +
          ",inner loop j = " + j);
        }
    }
}
```


Output

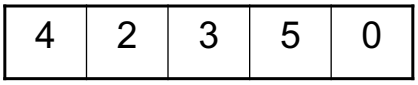
outer loop i = 0, inner loop j = 0
outer loop i = 0, inner loop j = 1
outer loop i = 1, inner loop j = 0
outer loop i = 1, inner loop j = 1

Nested Loop Analogy

- A nested loop is somewhat analogous to a an odometer or counter.



The “outer loop” increments by one each time the “inner loop” completes a full cycle.



Another example on nested loop: **This program prints the result of** $y = \sum_{n=0}^{N} x^n(2n+1)!$

```
public static void main(String args[]){
 double y = 0, x; int N;
 Scanner input = new Scanner(System.in);
 System.out.print("Enter x: " ); x = input.nextInt();
 System.out.print("Enter N: " ); N = input.nextInt();
 for (int n = 0; n <= N; n++)//outer loop as required by the summation- range
 {//inner loop 1: calculate first part p1 (xn)
   double p1 = 1;
   for (int i = 1; i <= n; i++)
     p1 = p1 * x;
   //inner loop 2: calculate second part p2 (2n+1)!
   double p2 = 1;
   for (int j = 1; j <= 2 * n + 1; j++)
     p2 *= j; //same as: p2 = p2 * j;
   //calculate y
    y = y + p1*p2;
 }
 System.out.println("y = " + y);
 input.close();
}
```

Another example on nested loop

```
//This program prints a square with
//a preselected chracter

public static void main(String [] args)
{
   char a = '&';
   int square_size = 4, row = 0, col = 0;
   for(row=0; row<square_size; row++)
   {
       for(col=0; col<square_size; col++)
       {
             System.out.print(a);//printing on the same row
       }
   System.out.println();//taking the cursor to the next row
   }
}
```

Output:
&&&&
&&&&
&&&&
&&&&

Homework problem

- Write a program to display each of the following outputs using nested for-loop:

```
X
XX
XXX
XXXX
```

```
   X
  XX
 XXX
XXXX
```

```
XXXX
XXX
XX
X
```

Another Example with Nested loop: 3 bit numbers

```
public static void main(String [] args)
{
   int d1, d2, d3;
   for (d3 = 0; d3 < 2; d3++)
    {
      for (d2 = 0; d2 < 2; d2++)
         {
         for (d1 = 0; d1 < 2; d1++)
              {
                     System.out.println("d3 = "+ d3 +
                     "d2= " + d2 + "d1= " + d1);
              }
        }
  }
}
```

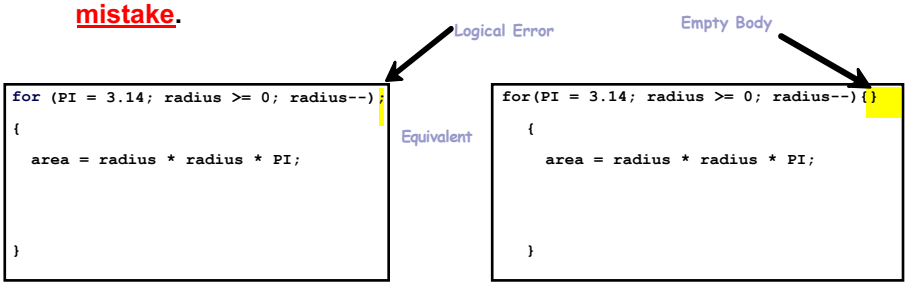

Output: 3 bit numbers

d3=0 d2=0 d1=0
d3=0 d2=0 d1=1
d3=0 d2=1 d1=0
d3=0 d2=1 d1=1
d3=1 d2=0 d1=0
d3=1 d2=0 d1=1
d3=1 d2=1 d1=0
d3=1 d2=1 d1=1

Home work Question: Modify the code in the previous slide so that it can print the above table in the reverse order (the bottom one should be printed first and so on)

Caution!

- Adding a semicolon at the end of a for header is a common mistake.



  - This mistake is hard to find, because it is NOT a compilation error, it is a logical error.
  - This error often occurs when you use the next-line block style.

Practice Problem: Calculate n! using while-loop

```
//This while loop calculates factorial n where,
//n! = 1 * 2 *..* n = n*(n-1)*(n-2)* …..*1;

public static void main(String [] args){
  Scanner input = new Scanner(System.in);
  double nfact=1; int n, m;
  System.out.print("Enter a positive integer: ");
  n = input.nextInt();
  while(n > 0){
    nfact = nfact*n;
    n--; //n is being decremented in every iteration
  }
  System.out.println("The factorial is "+nfact);
  input.close();
}/*NOTE: Always use double-type variable to store the factorial result.*/
```

Practice Problem: Calculate n! using for-loop

```
//This for-loop calculates n!
public static void main(String [] args){
 Scanner input = new Scanner(System.in);
 double nfact=1; int n;
 System.out.print("Enter a positive integer: ");
 n = input.nextInt();
 for (int i = 1; i <= n; i++){
  nfact = nfact*i;
 }
 System.out.println( "The factorial is " + nfact);
 input.close();
}
```


Practice Problem: Calculate n! using do-while loop

```
//This do-while loop calculates n!
public static void main(String [] args){
 Scanner input = new Scanner(System.in);
 double nfact=1; int n, i = 1;
 System.out.print("Enter a positive integer: ");
 n = input.nextInt();
 do
 {
    nfact = nfact*i;
 } while (++i <= n);
 System.out.println( "The factorial is " + nfact);
 input.close();
}
```

Practice Problem on nested loop

- **Problem: Write a program that prompts the user to enter an integer from 1 to 15 and displays a pyramid. For example, if the input integer is 6, the output is shown below.**

```
          1
        2 1 2
      3 2 1 2 3
    4 3 2 1 2 3 4
  5 4 3 2 1 2 3 4 5
6 5 4 3 2 1 2 3 4 5 6
```



```
//The solution
public static void main(String[] args){
 Scanner input = new Scanner(System.in);
 System.out.print("Enter the size of the pyramid: ");
 int n = input.nextInt();
 for(int i=1; i<=n; i++)//loop for ith row; i is the row no.
 {//printing spaces in each row
   for(int j=n; j>i; j--)
      System.out.print(" ");
   //printing numbers from 'row number' to 1 in each row
   for(int k=i; k>=1; k--)
      System.out.print(k);
   //printing numbers from 2 to 'row number' in each row
   for(int m=2; m<=i; m++)
      System.out.print(m);
   System.out.println();//printing a new line after each row
 }
 input.close();
}
```

Practice Problem on nested loop

Problem: Write a java program that uses nested for-loops to print a multiplication table shown below.

```
          Multiplication Table
       1   2   3   4   5   6   7   8   9
------------------------------------------
1 |    1   2   3   4   5   6   7   8   9
2 |    2   4   6   8  10  12  14  16  18
3 |    3   6   9  12  15  18  21  24  27
4 |    4   8  12  16  20  24  28  32  36
5 |    5  10  15  20  25  30  35  40  45
6 |    6  12  18  24  30  36  42  48  54
7 |    7  14  21  28  35  42  49  56  63
8 |    8  16  24  32  40  48  56  64  72
9 |    9  18  27  36  45  54  63  72  81
```



```java
//The solution
public static void main(String[] args) {
  // Display the table heading
  System.out.println("         Multiplication Table");
  // Display the number title
  System.out.print("    ");
  for (int j = 1; j <= 9; j++)
    System.out.print("   " + j);
  System.out.println("\n------------------------------------");
  // Print table body using nested-for
  for (int i = 1; i <= 9; i++) {
    System.out.print(i + " | ");
    for (int j = 1; j <= 9; j++) {
    // Display the product and align properly
      System.out.printf("%4d", i * j);//see slide 150, Unit 2
    }
   System.out.println();
  }
}
```

Practice Problem: Finding the Greatest Common Divisor

Problem: Write a program that prompts the user to enter two positive integers and finds their greatest common divisor.

Programming tip: Suppose you enter two integers 4 and 2, their greatest common divisor is 2. Suppose you enter two integers 16 and 24, their greatest common divisor is 8. So, how do you find the greatest common divisor? Let the two input integers be n1 and n2. You know number 1 is a common divisor, but it may not be the greatest commons divisor. So you can check whether k (for k = 2, 3, 4, and so on) is a common divisor for n1 and n2, until k is greater than n1 or n2.


```
//The solution
public static void main(String[] args) {
 // Create a Scanner
 Scanner input = new Scanner(System.in);
 // Prompt the user to enter two integers
 System.out.print("Enter first integer: ");
 int n1 = input.nextInt();
 System.out.print("Enter second integer: ");
 int n2 = input.nextInt();
 int gcd = 1;
 int k = 2;
 while (k <= n1 && k <= n2) {
   if (n1 % k == 0 && n2 % k == 0)
     gcd = k;
   k++;
 }
 System.out.println("The greatest common divisor for " + n1 +
 " and " + n2 + " is " + gcd);
 input.close();
}
```

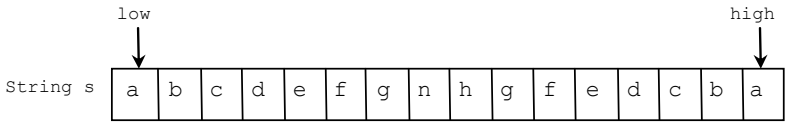
Practice Problem: Predicting the Future Tuition

- Problem: Suppose that the tuition for a university is $14,000 this year and tuition increases 7% every year. In how many years will the tuition be doubled?


```
//The solution
public static void main(String[] args) {
 double tuition = 14000;   // Year 0
 int year = 0;
 while (tuition <= 28000) {
    tuition = tuition * 1.07;
    year++;
 }
 System.out.println("Tuition will be doubled in
 " + year + " years");
 System.out.printf("Tuition will be $%.2f in %1d
 years",  tuition, year); );//see slide 151, Unit 2
}
```

Practice Problem: Checking Palindrome

- Note: A string is a palindrome if it reads the same forward and backward. The words “mom,” “dad,” and “noon,” for instance, are all palindromes.
- The problem: Write a program that prompts the user to enter a string and reports whether the string is a palindrome.
- Programming tip: One solution is to check whether the first character in the string is the same as the last character. If so, check whether the second character is the same as the second-to-last character. This process continues until a mismatch is found or all the characters in the string are checked, except for the middle character if the string has an odd number of characters.




```
//The solution
public static void main(String[] args) {
 Scanner input = new Scanner(System.in);
 System.out.print("Enter a string: ");
 String s = input.nextLine();
 // The index of the first character in the string
 int low = 0;
 // The index of the last character in the string
 int high = s.length() - 1;
 boolean isPalindrome = true;
 while (low < high) {
   if (s.charAt(low) != s.charAt(high)) {
     isPalindrome = false;
   break;
   }
 low++;
 high--;
 }
 if (isPalindrome)
   System.out.println(s + " is a palindrome");
 else
   System.out.println(s + " is not a palindrome");
 input.close();
}
```

Practice Problem: Displaying Prime Numbers

Problem: Write a program that displays whether an entered integer number is prime or not. An integer number greater than 1 is *prime* if it is divisible by 1 and by itself. For example, 2, 3, 5, and 7 are prime numbers, but 4, 6, 8, and 9 are not.


```
//The solution
public static void main(String[] args) {
 Scanner input = new Scanner(System.in);
 System.out.print("Input an integer number: ");
 int number = input.nextInt();
 boolean isPrime = true;
 // Test if number is prime
 for (int divisor = 2; divisor <= number / 2; divisor++) {
    if (number % divisor == 0) { // If true, number is not prime
    isPrime = false; // Set isPrime to false
    break; // Exit the for loop
    }
 }
 // Print the message
 if (isPrime)
    System.out.println(number + " is a prime number.");
 else
    System.out.println(number + " is NOT a prime number.");
 input.close();
}
```

Exercise Problem: Displaying Prime Numbers

Problem: Write a program that displays the first 50 prime numbers in five lines, each of which contains 10 numbers.

Programming Tip: The problem can be broken into the following tasks:

- For number = 2, 3, 4, 5, 6, ..., test whether the number is prime.
- Determine whether a given number is prime.
- Count the prime numbers.
- Print 10 prime numbers per line.

Home work problem

- Draw a flow-chart and then write the corresponding code with the following specs:
  - Prompt the user to enter an integer value n greater than 10; validate that number
  - Now ask the user to enter these n different numbers from the key board and do the following:
    - Validate that the entered numbers are in between 100 and 200 inclusive.
    - Calculate the average of all the odd numbers entered and
    - Calculate the average of all the even numbers entered.
  - Your code should print the following:

    You've entered <x> odd numbers and <n-x> even numbers; the average for the odd numbers is <odd_average> and the average for the even numbers is <even_average>

    Note: <> Represents the corresponding values.

Important Note of for-loop

- The initial-action in a for loop can be a list of zero or more comma-separated expressions.
- The action-after-each-iteration in a for loop can be a list of zero or more comma-separated statements.
- Therefore, the following two for loops are correct. They are rarely used in practice, however.

```
int i;
for (i = 1; i < 100; System.out.print(i++);
System.out.println(i);


for (int i = 0, j = 0; (i + j < 10); i++, j++)
{
  // Do something
}
```


Note on for and while loops

- If the loop-repetition-condition in a for loop is omitted, it is implicitly true. As long as the for header has two semicolons, the code will compile.
- Thus the statement given below in (a), which is an infinite loop, is correct.
- Nevertheless, it is better to use the equivalent loop shown in (b) to avoid confusion:

This is better

```
for ( ; ; )
{
   // do something
}
```

(a)

Equivalent

```
while (true)
{
   // do something
}
```

(b)

Switching between loops

- The three forms of loop statements, while, do-while, and for, are expressively equivalent; that is, you can write a loop in any of these three forms.
- For example, a while loop in (a) in the following figure can always be converted into the following for loop in (b):

```
while ( loop_repetition_condition )
{
    Statement(s);
}
```

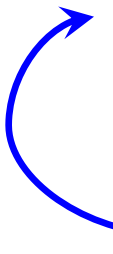
(a)

```
for ( ;loop_repetition_condition; )
{
    Statement(s);
}
```

(b)

Switching between loops *cont.*

- A for loop in (a) in the following figure can generally be converted into the following while loop in (b) except in certain special cases



```
for (initial_action; loop_repetition_condition;
    action_after_each_iteration)
    {
      //loop body
    }
```

(a)

```
initial_action;
while ( loop_repetition_condition )
{
    //loop body
    action_after_each_iteration;
}
```

(b)

Example: Converting the while-loop from slide #60 to for-loop for validation

Problem: Prompt the user to enter a number between 3 and 30 inclusive, and after validating the number, print the entered number on the screen. Assumption: user will not enter any non-integer value.

```
public static void main(String[] args){
  Scanner input = new Scanner(System.in);
  System.out.print("Enter an integer : ");
  int x = input.nextInt();
  for(;!((x>=3)&&(x<=30));){ //same as: while((x<3)||(x>30))
    System.out.println("You've entered an invalid number");
    System.out.print("Enter an integer between 3 and 30: ");
    x = input.nextInt();
  }
  System.out.println("You've entered "+x);
  input.close();
}
```


Recommendations

- **Use the loop that is most clear to you so that you feel very comfortable using it.**
- **In general, a for loop may be used if the number of repetitions is known**
  - **For example, when you need to print a message 100 times.**
- **A while loop may be used if the number of repetitions is not known,**
  - **For example, reading the numbers until the input is 0.**
- **A do-while loop can be used to replace a while loop if the loop body has to be executed before testing the continuation condition.**
  - **For example, printing a menu, and choose one of the options**

`break` and `continue`

- **`break;`**
  - terminates loop completely
  - Program execution continues with the statement that follows the loop
- **`continue;`**
  - It is used to skip remaining statements in a loop and start a new iteration

Practice! What is the output?

```
public static void main(String[] args){
     for (int i=0; i<10; i++){
          if (i%2 != 0)
               continue;
     System.out.println(i);
     }//end for
     System.out.println("done");
}//end main
```

```
0
2
4
6
8
done
```

Practice! What is the output?

```
public static void main(String[] args){
     for (int i=0; i<10; i++){
          if (i%2 != 0)
               break;
          System.out.println(i);
     }//end for
     System.out.println("done");
}//end main
```

0
done

Application of break: break the loop when the user enters 'over'

```
public static void main(String[] xyz){
 Scanner input = new Scanner(System.in);
 String name, maxName = "hhhh"; double grade = 1000, max = 10;
 int counter=0;
 do{
  System.out.print("Name: "); name = input.next();
  if(name.equals("over"))//any object needs to be compared using .equals() method
    break;
  System.out.print("Grade: "); grade = input.nextDouble(); counter++;
  if(counter == 1){
    max = grade; maxName = name;
  }
  else if(grade>max){
    max = grade; maxName = name;
  }
 }while(true);
 System.out.println(maxName+" got the highest marks "+max);
 input.close();
}
/*modify the code to find the student scored lowest mark, the class average, total students etc.*/
```

Practice: Counting the positive numbers divisible by 5 but not by 15

```
public static void main(String[] xyz){
 Scanner input = new Scanner(System.in);
 int num = 0, counter=0;
 do{
  System.out.print("Enter number: ");
  num = input.nextInt();
  if(num<=0)
    break;
  if(num%5==0 && num%15!=0)
    counter++;
  }while(true);
 System.out.println("You have entered
 "+counter+" numbers which are divisible by 5
 but not by 15");
 input.close();
}
```


Practice: validation using for loop: validate that an entered character is in between ‘a’ and ‘d’ inclusive

```
public static void main(String[] xyz){
  Scanner input = new Scanner(System.in);
  char choice = 'i';
  System.out.print("Enter a character between a and d
                    inclusive: ");
  choice = input.next().charAt(0);
  for( ;!(choice>='a' && choice<='d'); ){
    System.out.println("Invalid character! ");
    System.out.print("Enter a character between a and d
                      inclusive: ");
    choice = input.next().charAt(0);
   }
   System.out.println("You entered " + choice + ".");
   input.close();
}/*Note: The alphabets (a to z) are stored as ASCII ( http://www.asciitable.com/)
equivalent integer numbers in the memory. These integer values are such that A<B<C
…..<Z<a<b<c…..<z; in other words, uppercase A gets the minimum value and lowercase z
gets the maximum value.*/
```

Review

9) What is the output?

```
int x(5), y(6);
if (x > y)
 System.out.println("x is greater");
else
 System.out.println("y is greater");
```

a) `x is greater`

b) `y is greater`

c) `x is greater`
`y is greater`

d) `No output`

Review

10) Assume x is 0. What is the output of the following statement?

```
if (x > 0)
 System.out.println("x is greater than 0");
else if (x < 0)
 System.out.println("x is less than 0");
else
 System.out.println("x equals 0");
```

a) x is greater than 0

b) x is less than 0

c) x equals 0

d) No output

Review

11) Which of the following statements will display “True!" given the following declaration

```
int i(100), j(0);
```

```
a) if (!(j<1)) System.out.println(“True”);
b) if (i>0 || j>50) System.out.println(“True”);
c) if (i<3) System.out.println(“True”);
d) if((j<i)&&(i<=10))System.out.println(“True”);
```

Review

12) How many times the following code prints “Welcome to Java”?

```
int count = 0;
while (count < 10)
{
  System.out.println(“Welcome to Java”);
  count++;
}
```

a) 8
b) 9
c) 10
d) 11
e) 0

Review

13) How many times the following code prints "Welcome to Java"?

```
int count = 0;
while (count++ < 10)
{
 System.out.println("Welcome to Java");
}
```

a) 8
b) 9
c) 10
d) 11
e) 0

Review

14) How many times the following code prints "Welcome to Java"?

```
int count = 0;
do
{
 System.out.println("Welcome to Java");
} while (count++ < 10);
```

a) 8
b) 9
c) 10
d) 11
e) 0

Review

15) How many times the following code prints “Welcome to Java”?

```
int count = 0;
do
{
  System.out.println(“Welcome to Java”);
} while (++count < 10);
```

a) 8
b) 9
c) 10
d) 11
e) 0

Review

16) Analyze the following code.

```
int x = 1;
while (0 < x) && (x < 100)
 System.out.print(x++);
```

a) The loop runs for ever.
b) The code does not compile because the loop body is not in the braces.
c) The code does not compile because (0 < x) && (x < 100) is not enclosed in a pair of parenthesis
d) The numbers 1 to 99 are displayed.
e) The numbers 2 to 100 are displayed.

Review

17) The following code will result in an infinite loop?

```
int balance = 10;
while (balance >= 1)
{
   if (balance < 9)
        continue;
   balance = balance - 9;
}
```

a) True
b) False

Review

18) What will be the value of the variable 'balance' after the following code is executed?

```
int balance = 10;
while (balance >= 1) {
   if (balance < 9)
       break;
   balance = balance - 9;
}
```

a) -1
b) 0
c) 1
d) 2

Review

19) What is the output of the following code segment?

```
for (int i = 0; i < 15; i++)
{
  if (i % 4 == 1)
    System.out.print(i+" ");
}
```

a) 1 3 5 7 9 11 13 15
b) 1 5 9 13
c) 1 5 9 13 16
d) 1 3 5 7 9 11 13
e) 1 4 8 12

Review

20) What is the output of the following code segment?

```
for (int i = 0; i < 15; i++)
{
 if (i % 4 = 1)
   System.out.print(i+" ");
}
```

a) 1 3 5 7 9 11 13 15
b) 1 5 9 13
c) 1 5 9 13 16
d) 1 3 5 7 9 11 13
e) The code will not compile

Answer Key

1. A
2. A
3. A
4. B
5. C
6. B
7. A
8. B
9. B
10. C
11. B
12. C
13. C
14. D
15. C
16. C
17. A
18. C
19. B
20. E

Common mistakes in code writing with control statements

- Using conditional statements (if or switch) in place of loop statements.
- Missing ‘do’ or ‘while’ in the do-while statements
- Mixing ‘while’ with ‘do-while’ in the same block
- Using assignment (=) operator in place of equality (==) operator.
- Using ‘break’ to break an if-statement.
- Missing semicolon at the end of the while in do-while statement.
- Using semicolon at the end of while, for or if headers.