Random class (must import java.lang.Random)

Random()
  Constructor - creates a new pseudorandom number generator. Default seed value is the system time in milliseconds.
  Random generator = new Random();

float nextFloat()
  Returns a random number between 0.0 and 1.0 (exclusive).
  y = generator.nextFloat();

int nextInt()
  Returns a random number that ranges over all possible int values (positive and negative).
  x = generator.nextInt( ) [or x = generator.nextInt(10) \rightarrow returns a value in the range 0..9

Math class (in the package java.lang)

abs (int num)
  Returns the absolute value of num.
  y = Math.abs( y );

pow (double num, double power)
  Returns the value num raised to the specified power.
  Double area = Math.PI * Math.pow( radius, 2 );

random()
  Returns a positive random number as a double value between 0.0 (inclusive) and 1.0 (exclusive)
  Int value = (int)(Math.random() * 6) + 1;

sqrt (double num)
  Returns the square root of num as a double value. The value of num must be positive.
  System.out.println( "The square root of 1234 is: " + Math.sqrt( 1234 ) );

Java String Methods

Creating a String: String s = new String ("Hello World"); or String s = "Hello World";

charAt
  Requires an integer argument which indicates the position of the character the method returns.
  Indexes range from 0 to string length – 1.
  s1.charAt(3)

compareTo
  Returns:
  0 if the strings are the same
  Negative number if the calling object is less than the argument.
  Positive number if the calling object is greater than the argument.
  if ( s1.compareTo(s2) > 0 )

equals
  Evaluates the contents of two strings to determine if they are equivalent.
  if ( s1.equals("Hello") )
equalsIgnoreCase
  Ignores case when determining if two String are equivalent.
  if ( s1.equalsIgnoreCase("Hello") )

indexOf
  Determines whether a specific character occurs within a String. Returns the element number if found,
  else –1. Element numbers are in the range 0 to string length – 1.
  s1.indexOf('e')

length
  Returns the number of characters in this string
  s1.length()

endsWith
  Takes a String argument and returns true or false if a String object does or does not end with the
  specified argument.
  if ( s1.endsWith("ing") )

equals
  Evaluates the contents of two strings to determine if they are equivalent.
  if ( s1.equals("Hello") )
equalsIgnoreCase
  Ignores case when determining if two String are equivalent.
  if ( s1.equalsIgnoreCase("Hello") )

startsWith
  Takes a String argument and returns true or false if the String object does or does not start with the
  specified argument.
  if ( s1.startsWith("Cerritos") )

substring
  Extracts a substring of string begining at element n and ending at element m-1
  s1.substring(n, m)

toLowerCase
  Converts a string to its lowercase equivalent.
  s1 = s1.toLowerCase();

toUpperCase
  Converts a string to its uppercase equivalent.
  s1 = s1.toUpperCase();

toString
  Converts any primitive type to a String.
  int x=123; String s; s = String.toString(x); // equivalent to "" + x

trim
  Trims leading and trailing spaces from a String
  s1.trim();
Java Arrays

Create a single dimension array:

<datatype> obj-ref-var [ ] = new <datatype>[size];

//create an array of 20 elements. Valid subscripts are 0 .. 19
int myArray[ ] = new int[20];

//create an array of 5 elements set to the values in an initializer list. Valid subscripts are 0 .. 4
int myArray[ ] = { 1, 2, 3, 4, 5 };

Create a 2 dimensional array:

<datatype> obj-ref-var[ ][ ] = new <datatype>[rows][columns];

//create a 2D array with 10 rows and 50 columns.
double myArray[ ][ ] = new double[ 10 ][ 50 ];

//valid subscripts for 1st dimension are 0 .. 9
//valid subscripts for 2nd dimension are 0 .. 49
//valid range of elements: myArray[ 0 ][ 0 ] .. myArray[ 9 ][ 49 ]

Arrays are created using the operator new and specifying the number of elements in the array or created using an initializer list.

int studentScore[ ] = new int[ 5 ];      or      int studentScore[ ] = { 85, 92, 78, 83, 95 };
studentScore[ 0 ] = 85;
studentScore[ 1 ] = 92;
studentScore[ 2 ] = 78;
studentScore[ 3 ] = 83;
studentScore[ 4 ] = 95;

Arrays can be used to store a collection of values instead of declaring a number of independent variables. Anywhere a variable may be used in a Java program, an array element of the same data type could also be used.

int a, b, c, d, e;
int array[ ] = new int[ 5 ];

Examples of usage:

a = 1;
array[ 0 ] = 1;
e = a + b;

b = 2;
array[ 1 ] = 2;
array[ 4 ] = array[ 0 ] + array[ 1 ];

Arrays can be used to count and accumulate

c = 3;
array[ 2 ] = 3;
c = 2 * a + 5;
d = 4;
array[ 3 ] = 4;
c = 2 * array[ 0 ] + 5;
e = 5;
array[ 4 ] = 5;

Arrays can be used to count and accumulate

classCount[ 1 ]++;     //count
classTotal[ 1 ] += studentTotal; //accumulate

All arrays have a public field named length which holds the number of elements in the array. If x is a reference to a single-dimension array, x.length is the number of elements in the array.

Arrays are often used with loops:

for ( int j = 0; j < array.length; j++ )
    System.out.println( array[ j ] );