**Sorts and Searches for Project 1**

**Sequential Search**

public static int search (int[] x, int target)

{ int i;

for (i = 0; i < x.length; i++)

if (x[i] == target)

return i;

return -1; // not found

}

**Binary Search – when array key is sorted**

public static int binarysearch (int[] x, int target)

{ int first = 0, middle, last = x.length-1;

while (first <= last)

{

middle = (first + last)/2;

if (x[middle] == target)

{

return middle;

}

else if (x[middle] < target)

first = middle + 1;

else

last = middle – 1;

} // end of while

return -1;

} // end of method

**Selection Sort in text, Chapter 8, pages 513-514, uses two methods to accomplish the sort. The one given below uses only 1 method.**

public static void SelectionSort (int [] list) //ascending order

{

int min, temp;

for (int index = 0; index < list.length-1; index++)

{

min = index;

for (int scan = index + 1; scan < list.length; scan++)

if (list[scan] < list[min])

min = scan;

// Swap the values

temp = list[min];

list[min] = list[index];

list[index] = temp;

}

}

**Maximum-Minimum Sort //ascending order**

public static void max\_min (int[] x)

{ int i, j, temp;

for ( i = 0; i<x.length-1; i++)

for ( j = i+1; j<x.length; j++)

if (x[i] > x[j])

{ temp = x[i];

x[i] = x[j];

x[j] = temp;

}

}

**Enhanced Bubble Sort (which exits early if array becomes sorted)**

public static void bubble (int[] x) // ascending order

{ boolean sorted;

int temp;

int numpairs = x.length-1;

do

{ sorted = true;

for (int i = 0; i < numpairs; ++i)

if (x[i] > x[i+1])

{ temp = x[i];

x[i] = x[i+1];

x[i+1] = temp;

sorted = false;

}

numpairs--;

}

while (sorted == false);

}

**Insertion Sort //ascending order**

public static void insertion (int [] x)

{

for ( int i = 0; i < x.length; ++i)

{

int key = x[i];

int position = i;

while (position > 0 && x[position-1] > key)

{

x[position] = x[position-1];

position--;

}

x[position] = key;

}

}