//Mike Scott

//examples of array manipulations

public class ArrayExamples

{ public static void main(String[] args)

{ int[] list = {1, 2, 3, 4, 1, 2, 3};

findAndPrintPairs(list, 5);

bubblesort(list);

showList(list);

list = new int[]{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11};

bubblesort(list);

showList(list);

list = new int[]{11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, -1, -2};

bubblesort(list);

showList(list);

list = new int[]{1};

bubblesort(list);

showList(list);

}

// pre: list != null, list.length > 0

// post: return index of minimum element of array

public static int findMin(int[] list)

{ assert list != null && list.length > 0 : "failed precondition";

int indexOfMin = 0;

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

{ if(list[i] < list[indexOfMin])

{ indexOfMin = i;

}

}

return indexOfMin;

}

/\*

\*pre: list != null, newSize >= 0

\*post: nothing. the method does not succeed it resizing the

\* argument

\*/

public static void badResize(int[] list, int newSize)

{ assert list != null && newSize >= 0 : "failed precondition";

int[] temp = new int[newSize];

int limit = Math.min(list.length, newSize);

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

{ temp[i] = list[i];

}

// uh oh!! Changing pointer, not pointee. This breaks the

// relationship between the parameter and argument

list = temp;

}

/\*

\*pre: list != null, newSize >= 0

\*post: returns an array of size newSize. Elements from 0 to newSize - 1

\* will be copied into the new array

\*/

public static int[] goodResize(int[] list, int newSize)

{ assert list != null && newSize >= 0 : "failed precondition";

int[] result = new int[newSize];

int limit = Math.min(list.length, newSize);

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

{ result[i] = list[i];

}

return result;

}

/\*

\*pre: list != null

\*post: prints out the indices and values of all pairs of numbers

\*in list such that list[a] + list[b] = target

\*/

public static void findAndPrintPairs(int[] list, int target)

{ assert list != null : "failed precondition";

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

{ for(int j = i + 1; j < list.length; j++)

{ if(list[i] + list[j] == target)

{ System.out.println("The two elements at indices " + i + " and " + j

+ " are " + list[i] + " and " + list[j] + " add up to " + target);

}

}

}

}

/\*

\*pre: list != null;

\*post: sort the elements of list so that they are in ascending order

\*/

public static void bubblesort(int[] list)

{

assert list != null : "failed precondition";

int temp;

boolean changed = true;

for(int i = 0; i < list.length && changed; i++)

{ changed = false;

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

{ assert (j > 0) && (j + 1 < list.length) : "loop counter j " + j +

"is out of bounds.";

if(list[j] > list[j+1])

{ changed = true;

temp = list[j + 1];

list[j + 1] = list[j];

list[j] = temp;

}

}

}

assert isAscending( list );

}

public static void showList(int[] list)

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

System.out.print( list[i] + " " );

System.out.println();

}

/\* pre: list != null

post: return true if list is sorted in ascedning order, false otherwise

\*/

public static boolean isAscending( int[] list )

{ boolean ascending = true;

int index = 1;

while( ascending && index < list.length )

{ assert index >= 0 && index < list.length;

ascending = (list[index - 1] <= list[index]);

index++;

}

return ascending;

}

}