**public** **class** Sorting {

**public** **static** **void** mergsSort(**int**[] a){

**int** temp[] = **new** **int**[a.length];

*internalMergeSort*(a, temp, 0, a.length-1);

}

**private** **static** **void** internalMergeSort(**int**[] a, **int**[] temp, **int** m, **int** n){

**if**(m < n){

**int** middle = (m+n) / 2;

*internalMergeSort*(a, temp, m , middle);

*internalMergeSort*(a, temp, middle+1, n);

*merge*(a, temp, m, middle, middle+1, n);

}

}

**private** **static** **void** merge(**int**[] a, **int**[] temp, **int** m, **int** p, **int** q, **int** n){

**int** t = m;

**int** numElements = n - m + 1;

**while**(m <= p && q <= n){

**if**( a[m] < a[q])

temp[t++] = a[m++];

**else**

temp[t++] = a[q++];

}

**while**( m <= p)

temp[t++] = a[m++];

**while**( q <= n)

temp[t++] = a[q++];

**for**(**int** i = 0; i < numElements; i++,n--)

a[n] = temp[n];

}

**public** **static** **void** quickSort(**int**[] a){

*internalQuickSort*(a, 0, a.length-1);

}

**public** **static** **void** internalQuickSort(**int** [] a, **int** m, **int** n)

{

**int** p;

**if**(m>n)

**return**;

p = *partition*(a, m, n);

*internalQuickSort*(a, m, p-1);

*internalQuickSort*(a, p+1, n);

}

**public** **static** **int** partition(**int** [] A, **int** i, **int** j)

{

**int** middle, pivot, temp, p;

middle = (i+j) / 2;

pivot = A[middle];

A[middle] = A[i];

A[i] = pivot;

p = i;

**for**(**int** k =i+1; k<=j; k++)

{

**if**(A[k] < pivot)

{

p++;

temp = A[p];

A[p] = A[k];

A[k] = temp;

}

}

temp = A[i];

A[i] = A[p];

A[p] = temp;

**return** p;

}

**public** **static** **void** heapSort(**int**[] a){

**int** n = a.length-1;

**for**(**int** i = n/2; i >= 1;i--){

*heapify*(a, i, n);

}

**for**(**int** i = n-1 ;i >= 1; i--){

**int** temp;

temp = a[1];

a[1] = a[i+1];

a[i+1] = temp;

*heapify*(a, 1, i);

}

}

**private** **static** **void** heapify(**int**[] a, **int** h, **int** m){

**int** ah = a[h];

**int** j;

**for**(j = 2\*h; j <= m; j = 2\*j){

**if**(j < m)

**if**(a[j] < a[j + 1])

j=j+1;

**if**(ah >= a[j])

**break**;

**else**{

a[j/2] = a[j];

}

}

a[j/2] = ah;

}

}

**public** **class** SortMain {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int**[] a = { 8, 1, 23, 15, 31, 2, 26, 12};

**int**[] b = {3, 1, 4, 5, 9, 8, 7};

**int**[] c = {0, 20, 40, 50, 70, 30, 100, 80, 10, 90, 60};

**int** i;

System.***out***.println("정렬 전 배열 원소: ");

**for**(i = 0; i < a.length; i++)

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

System.***out***.println();

Sorting.*mergsSort*(a);

System.***out***.println("합병 정렬된 배열 원소: ");

**for**(i = 0; i < a.length; i++)

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

System.***out***.println();

System.***out***.println();

System.***out***.println("정렬 전 배열 원소 : ");

**for**(i=0; i<b.length; i++)

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

System.***out***.println();

Sorting.*quickSort*(b);

System.***out***.println("퀵 정렬된 배열 원소 : ");

**for**(i=0; i<b.length; i++)

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

System.***out***.println();

System.***out***.println();

System.***out***.println("정렬 전 배열 원소");

**for**(i = 1; i < c.length; i++)

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

System.***out***.println();

Sorting.*heapSort*(c);

System.***out***.println("하프 정렬된 배열 원소: ");

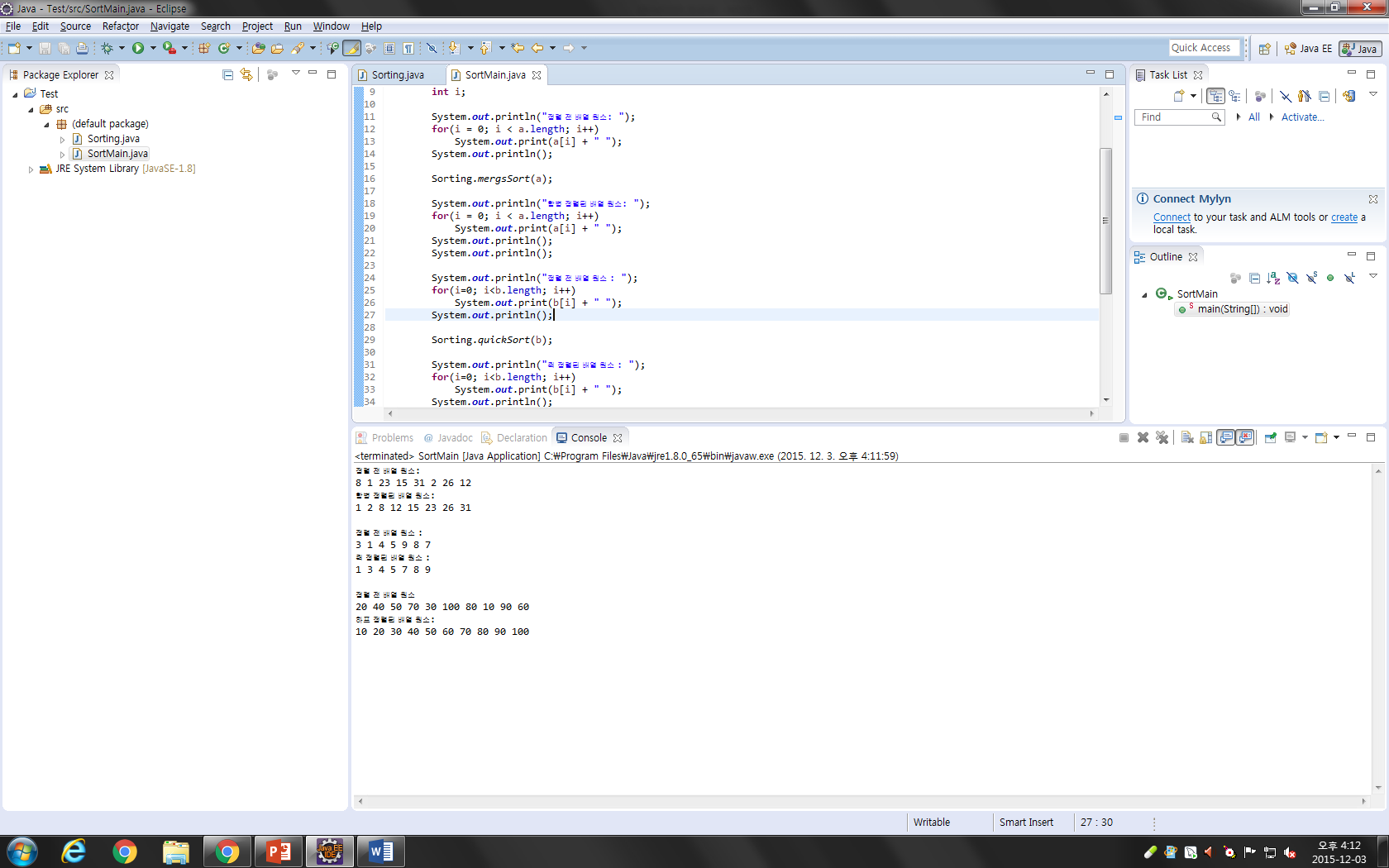
**for**(i = 1; i < c.length; i++)

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

System.***out***.println();

}

}



**public** **class** Node {

**public** **int** data;

**public** Node left;

**public** Node right;

**public** Node(**int** thedata){

**this**(thedata, **null**, **null**);

}

**public** Node(**int** thedata, Node lLink, Node rLink){

data = thedata;

**this**.left = lLink;

**this**.right = rLink;

}

}

**public** **class** BST {

**public** Node root;

**public** BST(**int** x){

root = **new** Node(x);

}

**public** Node insert(Node node, Integer x){

**if**(node == **null**){

**return** node = **new** Node(x);

}

**if**(x<(Integer) node.data){

node.left = insert(node.left, x);

}**else**{

node.right = insert(node.right, x);

}

**return** node;

}

**public** **void** inOrder(Node node){

**if**(node != **null**){

inOrder(node.left);

System.***out***.print(node.data + ",");

inOrder(node.right);

}

}

}

**public** **class** TreeSort {

**public** **static** **void** main(String[] args) {

**int**[] a = { 9, 1, 7, 11, 1, 5, 4, 3, 8 };

BST bst = **new** BST(**new** Integer(a[0]));

System.***out***.print("정렬 전 배열 원소 : " + a[0] + ",");

**for** (**int** i = 1; i < a.length; i++) {

System.***out***.print(a[i] + ",");

bst.insert(bst.root, **new** Integer(a[i]));

}

System.***out***.print("\n트리 정렬된 배열 원소 : ");

bst.inOrder(bst.root);

System.***out***.println();

}

}

