6.Three sort

**package** gr.duth.ee.euclid.datastructures.three\_sort;

**public** **class** ThreeSort {

**public** **static** **void** threeSort(IntegerArray a) {

**int** lo = 0;

**int** hi = a.length()- 1;

**int** mid = 0;

**while** (mid <= hi) {

**int** x=(a.compare(mid,1));

**if**(x<0) {

a.swap(mid,lo);

lo=lo+1;

mid=mid+1;

}

**else** **if**(x>0){

a.swap(mid,hi);

hi=hi-1;

}

**else** mid=mid+1;

}

}

}

5.SimpleIntegerSorting

**package** gr.duth.ee.euclid.datastructures.simple\_integer\_sorting;

**public** **class** SimpleIntegerSorting {

**public** **static** **int**[] simpleIntegerSorting(**int**[] values) {

**int** x=values.length;

**int** temp=0;

**for**(**int** i=0; i < x; i++){

**for**(**int** j=1; j < (x-i); j++){

**if**(values[j-1] < values[j]){

//swap elements

temp = values[j-1];

values[j-1] = values[j];

values[j] = temp;

}

}

}

**return** values;

}

}

4.Harmonic mean

**package** gr.duth.ee.euclid.datastructures.harmonic\_mean;

**public** **class** FindHarmonicMean {

**public** **static** **double** findHarmonicMean(**int**[] values) {

**double** x=0.0;

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

x=x+(1.0/values[i]);

}

**double** hm=values.length/x;

**return** hm;

}

}

3.Find Max Value

**package** gr.duth.ee.euclid.datastructures.find\_max\_value\_mdim;

**public** **class** FindMaxValue {

**public** **static** **int** findMaxValue(**int**[][] values) {

**int** max=values[0][0];

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

**for** (**int** j=0;j <values[i].length ;j++) {

**if** (values[i][j]> max){

max=values[i][j];

}

}

}

**return** max;

}

}

2.Factorial

**package** gr.duth.ee.euclid.datastructures.factorial;

**public** **class** Factorial {

**public** **static** **long** compute(**long** x) {

**long** y=1;

**for**(**int** i=1;i <= x;i++) {

y=y\*i;

}

**return** y;

}

}

1.Binary Search

**package** gr.duth.ee.euclid.datastructures.binary\_search;

**public** **class** BinarySearch {

**public** **static** **int** binarySearch(IntegerArray a, **int** valueToFind) {

**int** begin = 0;

**int** end = a.length() -1;

**int** y=-1;

**while** (begin <= end && y==-1 )

{

**int** middle = (begin + end)/2;

**int** x=a.get(middle);

**if** (x == valueToFind) {

y=middle;

}

**else** **if** (x > valueToFind)

{

begin = middle +1;

}

**else** end = middle-1 ;

}

**return** y;

}}