**package** com.lession4.OrderStatics;

**public** **class** OrderStaticsDemo {

**int** OrdeStaticsDemo(**int** arr[], **int** l, **int** r, **int** k)

{

**if** (k > 0 && k <= r - l + 1)

{

**int** pos = randomPartition(arr, l, r);

**if** (pos-l == k-1)

**return** arr[pos];

**if** (pos-l > k-1)

**return** OrdeStaticsDemo(arr, l, pos-1, k);

**return** OrdeStaticsDemo(arr, pos+1, r, k-pos+l-1);

}

**return** Integer.***MAX\_VALUE***;

}

**void** swap(**int** arr[], **int** i, **int** j)

{

**int** temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

**int** partition(**int** arr[], **int** l, **int** r)

{

**int** x = arr[r], i = l;

**for** (**int** j = l; j <= r - 1; j++)

{

**if** (arr[j] <= x)

{

swap(arr, i, j);

i++;

}

}

swap(arr, i, r);

**return** i;

}

**int** randomPartition(**int** arr[], **int** l, **int** r)

{

**int** n = r-l+1;

**int** pivot = (**int**)(Math.*random*()) \* (n-1);

swap(arr, l + pivot, r);

**return** partition(arr, l, r);

}

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

{

OrderStaticsDemo ob = **new** OrderStaticsDemo();

**int** arr[] = {16, 3, 4, 7, 9, 12, 36};

**int** n = arr.length,k = 5;

System.***out***.println("K'th smallest element is "+ ob.OrdeStaticsDemo(arr, 0, n-1, k));

}

}

}