

```
1  import java.io.*;
2  import java.util.*;
3
4
5  class BTNode{
6
7      private int nodeId;
8      private int data;
9      private int levelNum;
10     private BTNode leftChildPtr;
11     private BTNode rightChildPtr;
12
13     public BTNode () {}
14
15     public void setNodeId(int id){
16         nodeId = id;
17     }
18
19     public int getNodeId(){
20         return nodeId;
21     }
22
23     public void setData(int d){
24         data = d;
25     }
26
27     public int getData(){
28         return data;
29     }
30
31     public void setLevelNum(int level){
32         levelNum = level;
33     }
34
35     public int getLevelNum(){
36         return levelNum;
37     }
38
39     public void setLeftChildPtr(BTNode ptr){
40         leftChildPtr = ptr;
41     }
42
43     public void setRightChildPtr(BTNode ptr){
44         rightChildPtr = ptr;
45     }
46
47     public BTNode getLeftChildPtr(){
48         return leftChildPtr;
49     }
50
51     public BTNode getRightChildPtr(){
52         return rightChildPtr;
53     }
54
55     public int getLeftChildID(){
56         if (leftChildPtr == null)
57             return -1;
58
59         return leftChildPtr.getNodeId();
60     }
61
62     public int getRightChildID(){
63         if (rightChildPtr == null)
64             return -1;
```

```

65
66     return rightChildPtr.getNodeId();
67 }
68
69 }
70
71
72
73 class BinarySearchTree{
74
75     private int numNodes;
76     private BTNode[] arrayOfBTNodes;
77     private int rootNodeID;
78
79     public BinarySearchTree(int n){
80         numNodes = n;
81         arrayOfBTNodes = new BTNode[numNodes];
82
83         for (int index = 0; index < numNodes; index++){
84             arrayOfBTNodes[index] = new BTNode();
85             arrayOfBTNodes[index].setNodeId(index);
86             arrayOfBTNodes[index].setLeftChildPtr(null);
87             arrayOfBTNodes[index].setRightChildPtr(null);
88             arrayOfBTNodes[index].setLevelNum(-1);
89
90         }
91     }
92
93
94     public void setLeftLink(int upstreamNodeID, int downstreamNodeID){
95         arrayOfBTNodes[upstreamNodeID].setLeftChildPtr(arrayOfBTNodes[downstreamNodeID]);
96     }
97
98     public void setRightLink(int upstreamNodeID, int downstreamNodeID){
99         arrayOfBTNodes[upstreamNodeID].setRightChildPtr(arrayOfBTNodes[downstreamNodeID
100     ]);
101 }
102
103     public void selectionSort(int array[], int arraySize){
104
105         for (int iterationNum = 0; iterationNum < arraySize-1; iterationNum++){
106
107             int minIndex = iterationNum;
108
109             for (int j = iterationNum+1; j < arraySize; j++){
110
111                 if (array[j] < array[minIndex])
112                     minIndex = j;
113
114             }
115
116             // swap array[minIndex] with array[iterationNum]
117             int temp = array[minIndex];
118             array[minIndex] = array[iterationNum];
119             array[iterationNum] = temp;
120
121         }
122
123     }
124
125
126
127     public void constructBSTree(int[] array){

```

```

128
129     int leftIndex = 0;
130     int rightIndex = numNodes-1;
131     int middleIndex = (leftIndex + rightIndex)/2;
132
133     rootNodeID = middleIndex;
134     arrayOfBTNodes[middleIndex].setData(array[middleIndex]);
135
136     ChainNodes(array, middleIndex, leftIndex, rightIndex);
137
138 }
139
140
141 public void ChainNodes(int[] array, int middleIndex, int leftIndex, int rightIndex){
142
143     if (leftIndex < middleIndex){
144         int rootIDLeftSubtree = (leftIndex + middleIndex-1)/2;
145         setLeftLink(middleIndex, rootIDLeftSubtree);
146         arrayOfBTNodes[rootIDLeftSubtree].setData(array[rootIDLeftSubtree]);
147         ChainNodes(array, rootIDLeftSubtree, leftIndex, middleIndex-1);
148     }
149
150
151     if (rightIndex > middleIndex){
152         int rootIDRightSubtree = (rightIndex + middleIndex + 1)/2;
153         setRightLink(middleIndex, rootIDRightSubtree);
154         arrayOfBTNodes[rootIDRightSubtree].setData(array[rootIDRightSubtree]);
155         ChainNodes(array, rootIDRightSubtree, middleIndex+1, rightIndex);
156     }
157
158 }
159
160
161
162 public void printLeafNodes(){
163
164     for (int id = 0; id < numNodes; id++){
165
166         if (arrayOfBTNodes[id].getLeftChildPtr() == null && arrayOfBTNodes[id].
167             getRightChildPtr() == null)
168             System.out.print(arrayOfBTNodes[id].getData() + " ");
169     }
170
171     System.out.println();
172 }
173
174
175
176 public void InOrderTraversal(int nodeid){
177
178     if (nodeid == -1)
179         return;
180
181
182
183     InOrderTraversal(arrayOfBTNodes[nodeid].getLeftChildID());
184     System.out.print(arrayOfBTNodes[nodeid].getData() + " ");
185     InOrderTraversal(arrayOfBTNodes[nodeid].getRightChildID());
186
187 }
188
189
190 public void PrintInOrderTraversal(){

```

```

191         InOrderTraversal(rootNodeID);
192         System.out.println();
193     }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 class BSTRandomArraySortedInputs{
204 }
205     public static void selectionSort(int array[], int arraySize){
206 }
207         for (int iterationNum = 0; iterationNum < arraySize-1; iterationNum++){
208 }
209             int minIndex = iterationNum;
210 }
211             for (int j = iterationNum+1; j < arraySize; j++){
212 }
213                 if (array[j] < array[minIndex])
214                     minIndex = j;
215 }
216             }
217 }
218             // swap array[minIndex] with array[iterationNum]
219             int temp = array[minIndex];
220             array[minIndex] = array[iterationNum];
221             array[iterationNum] = temp;
222 }
223         }
224 }
225 }
226 }
227 }
228     public static void main(String[] args){
229 }
230         Scanner input = new Scanner(System.in);
231 }
232         int numElements;
233         System.out.print("Enter the number of elements: ");
234         numElements = input.nextInt();
235 }
236         int array[] = new int[numElements];
237 }
238         int maxValue;
239         System.out.print("Enter the maximum value for an element: ");
240         maxValue = input.nextInt();
241 }
242         Random randGen = new Random(System.currentTimeMillis());
243 }
244         System.out.print("Array Generated: ");
245         for (int index = 0; index < numElements; index++){
246             array[index] = randGen.nextInt(maxValue);
247             System.out.print(array[index] + " ");
248         }
249         System.out.println();
250 }
251         selectionSort(array, numElements);
252 }
253         BinarySearchTree bsTree = new BinarySearchTree(numElements);
254         bsTree.constructBSTree(array);

```

```
255
256
257     System.out.println("Inorder Traversal: ");
258     bsTree.PrintInOrderTraversal();
259
260     System.out.print("Leaf nodes: ");
261     bsTree.printLeafNodes();
262     System.out.println();
263
264 }
265
266 }
```

```
Enter the number of elements: 10
Enter the maximum value for an element: 25
array generated: 1 19 13 1 19 5 12 11 3 10
Inorder traversal: 1 1 3 5 10 11 12 13 19 19
Leaf nodes: 1 5 12 19
```
