

### PART 1

✓ **HuffmanTree** Classında

➔ Encode method implement edildi.

\***Private** recursive method;

/\*\*

\* Search for character and when found return coded string

\* @param ch is the search character on huffman tree.

\* @param huffmanTree is the builed huffman tree.

\* @param code is the code when found ch will return.

\* @return is the for ch code on huffmantree.

\*/

- **private String encode(char ch, BinaryTree<HuffData> huffmanTree, String code)**

\* **Public** method;

/\*\*

\* Method to encode string message into Huffman encodes.

\* @param message The input message as a String

\* which is composed on the specified alphabet in the book

\* @param huffmanTree It's created huffman code for the alphabet

\* @return The encoded message as a String zero and ones.

\*/

**public String encode(String message, BinaryTree<HuffData> huffmanTree);**

### PART 2

➔ **AscendingOrderTraversal** Classında;

➔ private final LinkedList<BinaryTree<E>> parents;

➔ public AscendinOrderTraversal(BinaryTree<E> root);

-> **BinaryTree<E> current;**

**for (current= root; current != null; current = current.getLeftSubtree()) {**

**this.parents.push(current);**

**}**

**ile ilk önce ağacın sol kısmı linkedliste eklenir root'dan başlayarak.**

➔ **public boolean hasNext();**

/\*\*

\* Look if with iterator we have next element.

\* @return true if have next element, otherwise false.

\*/

➔ **public E next();**

/\*\*

\* This method return for next element. Rule of inorder tree

\* @return next element

\*/

**BinaryTree current = parents.pop();**

**for(BinaryTree<E> child = current.getRightSubtree(); child !=null; child = child.getLeftSubtree())**

**{**

**this.parents.push(child);**



```
*/  
@Override  
➤ public void enqueue(E element)  
Sona eleman eklenir add(size-1, item )ile.
```

---

```
/**  
 * This method, look my array queue is empty.  
 * @return if empty true, otherwise false.  
 */  
@Override  
➤ public boolean isEmpty()  
Arraylistimde eleman var mı ona bakar.
```

---

```
/**  
 * Return number of my array queue  
 * @return the size of my array queue.  
 */  
@Override  
➤ public int size()  
Arraylistimde kaç eleman var onu verir.
```

---

#### ➔ **PriorityQueue LinkedList Classında**

```
/**  
 * This Constructor take priority queue and comparator  
 * @param myQueue my priority queue  
 * @param comp my current comparator  
 */  
➤ Constuctor with 2 parameters  
-> Priortiy queue elemanları linkedlisteme eklenir.
```

---

```
/**  
 * Remove first element of Array queue  
 * @return and return removed element.  
 */  
@Override  
➤ public E dequeue()  
Ilk eleman silinir- removefirst() ile
```

---

```
/**  
 * Add element at end  
 * @param element will be add.  
 */  
@Override  
➤ public void enqueue(E element)  
Sona eleman eklenir addLast() ile.
```

---

```
/**  
 * This method, look my link queue is empty.  
 * @return if empty true, otherwise false.  
 */  
@Override
```

- **public boolean isEmpty()**  
**LinkedListimde eleman var mı ona bakar.**

```
/**
 * Return number of my link queue
 * @return the size of my link queue.
 */
@Override
```

- **public int size()**  
**LinkedListimde kaç eleman var onu verir.**

## ➔ PriorityQueue BinarySearchTree Classında

- ```
/**
 * This Constructor take priority queue and comparator
 * @param myQueue my priority queue
 * @param comp my current comparator
 */
```

- **Constuctor with 2 parameters**  
**-> Priortiy queue elemanları binaryTreeme eklenir.**

```
/**
 * Remove first element of Tree queue
 * @return and return removed element.
 */
@Override
```

- **public E dequeue()**  
**Ilk eleman silinir- delete(root.getData()) ile**

```
/**
 * Add element at end
 * @param element will be add.
 */
@Override
```

- **public void enqueue(E element)**  
**Sona eleman eklenir add(element) ile.**

```
/**
 * This method, look my tree queue is empty.
 * @return if empty true, otherwise false.
 */
@Override
```

- **public boolean isEmpty()**  
**BinaryTreem de eleman var mı ona bakar.**

```
/**
 * Return number of my tree queue
 * @return the size of my tree queue.
 */
@Override
```

- **public int size()**

## BinaryTree de kaç eleman var onu verir.

### ➔ PriorityQueue\_Vector Classında

➤  
➤ `/**`  
    `* This Constructor take priority queue and comparator`  
    `* @param myQueue my priority queue`  
    `* @param comp my current comparator`  
    `*/`

➤ **Constructor with 2 parameters**  
➔ **Priortiy queue elemanları vectore eklenir.**

`/**`  
    `* Remove first element of vector queue`  
    `* @return and return removed element.`  
    `*/`

➤ `@Override`  
    **`public E dequeue()`**  
➔ **Ilk eleman silinir- remove(0) ile**

`/**`  
    `* Add element at end`  
    `* @param element will be add.`  
    `*/`

➤ `@Override`  
    **`public void enqueue(E element)`**  
➔ **Sona eleman eklenir add(size, element) ile.**

`/**`  
    `* This method, look my vector queue is empty.`  
    `* @return if empty true, otherwise false.`  
    `*/`

➤ `@Override`  
    **`public boolean isEmpty()`**  
➔ **Vectorumde eleman var mı ona bakar.**

`/**`  
    `* Return number of my tree queue`  
    `* @return the size of my tree queue.`  
    `*/`

➤ `@Override`  
    **`public int size()`**  
➔ **Vectorumde kaç eleman var onu verir.**

```

classDiagram
    class HuffmanTree_HuffmanData {
        <<static>>
        + symbol: Character
        + weight: double
        + getSymbol(): Character
        + get HuffmanData(double, Character)
    }
    class AscendOrderTraversal {
        + parents: LinkedList<BinaryTree-E>
        + AscendOrderTraversal(BinaryTree-E)
        + hasNext(): boolean
        + iterator(): Iterator-E
        + main(String[]): void
        + next(): E
        + remove(): void
    }
    class HuffmanTree_CompareHuffmanTrees {
        <<static>>
        + compare(BinaryTree-HuffmanData, BinaryTree-HuffmanData)
    }
    class HuffmanTree {
        <<static>>
        + huffmanTree: BinaryTree-HuffmanData
        + buildTree(HuffmanData[]): void
        + decode(String): String
        + encode(String, BinaryTree-HuffmanData): String
        + encode(char, BinaryTree-HuffmanData): String
        + main(String[]): void
        + printCode(PrintStream, String, BinaryTree-HuffmanData): void
        + printCode(PrintStream): void
        + toString(): String
    }
    class BinarySearchTree {
        <<static>>
        + addReturn: boolean
        + deleteReturn: E
        + add(E): boolean
        + add(Node-E, E): Node-E
        + delete(E): E
        + delete(Node-E, E): Node-E
        + find(E): E
        + find(Node-E, E): E
        + findLargestChild(Node-E): E
        + main(String[]): void
    }
    class MyQueue {
        <<interface>>
        + dequeue(): E
        + enqueue(E): void
        + isEmpty(): boolean
        + size(): int
    }
    class PriorityQueue_UnsortedVector {
        + comparator: Comparator-E
        + myUnsortedVectorQueue: Vector-E
        + dequeue(): E
        + enqueue(E): void
        + isEmpty(): boolean
        + PriorityQueue_UnsortedVector(PriorityQueue-E, Comparator-E)
        + size(): int
        + toString(): String
    }
    class PriorityQueue_LinkedList {
        + comparator: Comparator-E
        + myLinkedListQueue: LinkedList-E
        + dequeue(): E
        + enqueue(E): void
        + isEmpty(): boolean
        + PriorityQueue_LinkedList(PriorityQueue-E, Comparator-E)
        + size(): int
        + toString(): String
    }
    class PriorityQueue_ArrayList {
        + comparator: Comparator-E
        + myArrayListQueue: ArrayList-E
        + dequeue(): E
        + enqueue(E): void
        + isEmpty(): void
        + PriorityQueue_ArrayList(PriorityQueue-E, Comparator-E)
        + size(): int
        + toString(): String
    }
    class Part3_MainClassTest {
        + main(String[]): void
        + test_ArrayList(PriorityQueue-Integer, Comparator-Integer): void
        + test_LinkedList(PriorityQueue-Integer, Comparator-Integer): void
        + test_UnsortedVector(PriorityQueue-Integer, Comparator-Integer): void
    }

    HuffmanTree_HuffmanData "1" -- "1" HuffmanTree : huffmanTree
    HuffmanTree_HuffmanData "1" -- "1" AscendOrderTraversal : parents
    HuffmanTree_CompareHuffmanTrees "1" -- "1" HuffmanTree : compare
    HuffmanTree "1" -- "1" BinarySearchTree : myBinarySearchTreeQueue
    MyQueue <|-- PriorityQueue_UnsortedVector
    MyQueue <|-- PriorityQueue_LinkedList
    MyQueue <|-- PriorityQueue_ArrayList
    MyQueue <|-- Part3_MainClassTest
    
```

```
graph TD
    User((User))
    Message((Message))
    Tree((Tree))
    ShowTest_times((ShowTest_times))
    PART1_encode_huffTree((PART1_encode_huffTree))
    PART2_IteratorBinaryTree((PART2_IteratorBinaryTree))
    PART3_TestDiffrentType_PriorityQueue((PART3_TestDiffrentType_PriorityQueue))

    User -- «flow» --> Message
    User -- «flow» --> Tree
    User -- «flow» --> ShowTest_times
    PART1_encode_huffTree --> Message
    PART2_IteratorBinaryTree --> Tree
    PART3_TestDiffrentType_PriorityQueue --> ShowTest_times
```

## Tests

The screenshot displays an IDE window with three main panels:

- Source:** Shows the `HuffmanTreeTest.java` file with the following code:

```
133     String message = "e";
134     BinaryTree<HuffmanTree.HuffData> huffmanTree = new BinaryTree<>();
135     HuffmanTree instance = new HuffmanTree();
136     instance.buildTree(symbols);
137     String expResult = "010";
138     String result = instance.encode(message, huffmanTree);
139     assertEquals(expResult, result);
140
```
- Output:** Shows the build output for the `Test (HuffmanTreeTest)` task. It indicates a **BUILD SUCCESS** with a total time of 3.298s, finished at Thu Apr 14 03:33:25 EEST 2016, and a final memory usage of 4M/15M.
- Test Results:** Shows the test results for the `com.mycompany:CSE222_HW06_091044011_ercanuca_2016_huffmanCode:jar:1.0-SNAPSHOT` project. A green progress bar indicates **100.00 %** completion. The results show that all 5 tests passed in 0.213s:
  - `com.mycompany.cse222_hw06_091044011_ercanuca_2016_huffmanCode.HuffmanTreeTest` passed
  - `testMain` passed (0.038 s)
  - `testBuildTree` passed (0.0 s)
  - `testToString` passed (0.0 s)
  - `testDecode` passed (0.001 s)
  - `testEncode` passed (0.0 s)

AscendinOrderTraversalTest.java

```

61 AscendinOrderTraversal instance = new AscendinOrderTraversal(tree);
62 Integer expResult = new Integer(23);
63 Integer result = (Integer) instance.next();
64 assertEquals(expResult, result);
65 }
66
67 /**

```

Output

ASK FTP Site Deployer x Test (AscendinOrderTraversalTest) x

```

L
-----
BUILD SUCCESS
-----
Total time: 1.352s
Finished at: Thu Apr 14 03:35:52 EEST 2016

```

Test Results

com.mycompany:CSE222\_HW06\_091044011\_ercanuca\_2016\_HuffmanCode:jar:1.0-SNAPSHOT x

100.00 %

All 5 tests passed.(0.086 s)

- com.mycompany.cse222\_hw06\_091044011\_ercanuca\_2016\_huffmancode.AscendinOrderTraversalTest passed
  - testNext passed (0.02 s)
  - testHasNext passed (0.0 s)
  - testIterator passed (0.0 s)
  - testRemove passed (0.0 s)
  - testMain passed (0.0 s)

## Main tests

### -----PART\_1-----

Encoded Codes :

c: 00000  
 u: 00001  
 h: 0001  
 r: 0010  
 s: 0011  
 e: 010  
 i: 0110  
 n: 0111  
 b: 100000  
 g: 100001  
 p: 100010  
 y: 100011  
 o: 1001  
 a: 1010  
 l: 10110  
 d: 10111  
 v: 1100000  
 j: 1100001000



q: 1100001001  
 x: 1100001010  
 z: 1100001011  
 k: 11000011  
 w: 110001  
 m: 110010  
 f: 110011  
 t: 1101  
 space: 111

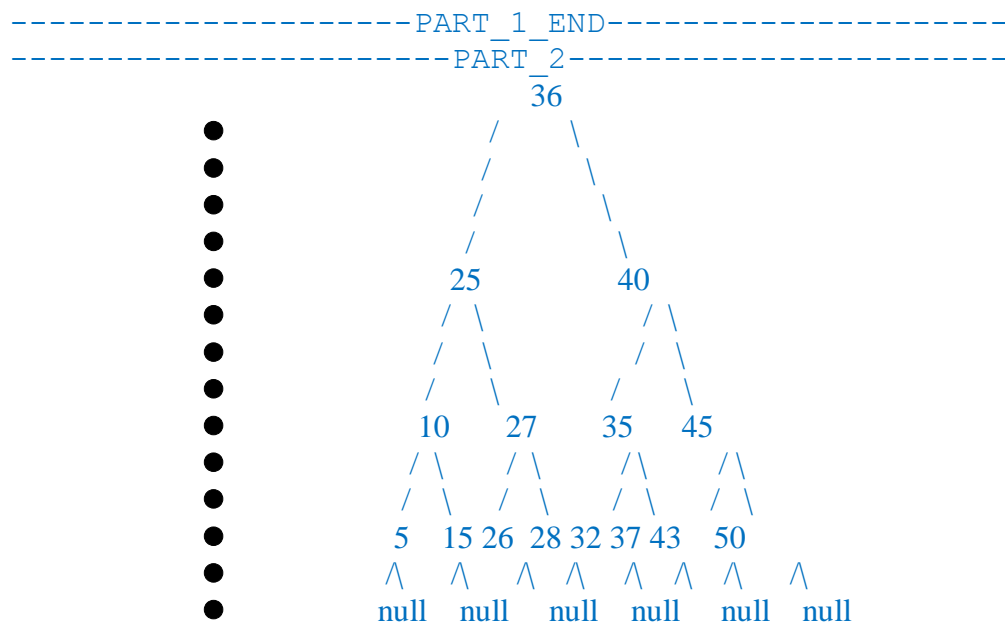
Code to Message :

1100001001111111100101000011 : q rg

Message to code :

merhaba ben ercan uca :

11001001000100001101010000010101111000000100111111010001000000101001111110000100001010



36  
 25  
 10  
 5  
 null  
 null  
 15  
 null  
 null  
 27  
 26  
 null  
 null  
 28  
 null  
 null  
 40

35  
32  
null  
null  
37  
null  
null  
45  
43  
null  
null  
50  
null  
null

5  
10  
15  
25  
26  
27  
28  
36  
32  
35  
37  
40  
43  
45  
50

-----PART\_2\_END-----  
-----PART\_3-----  
-----ARRAYLIST\_QUEUE\_TEST-----  
\*\*\*\*\*For 1 integer\*\*\*\*\*  
\*\*\*\*\*Enqueue 1 element\*\*\*\*\*  
Took approximately 18566 ns  
\*\*\*\*\*Dequeue 1 element\*\*\*\*\*  
Took approximately 18112 ns  
\*\*\*\*\*End for 1 number\*\*\*\*\*  
  
\*\*\*\*\*For 10 integer\*\*\*\*\*  
\*\*\*\*\*Enqueue 10 elements\*\*\*\*\*  
Took approximately 27169 ns  
\*\*\*\*\*Dequeue 10 elements\*\*\*\*\*  
Took approximately 11320 ns  
\*\*\*\*\*End for 10 numbers\*\*\*\*\*  
  
\*\*\*\*\*For 100 integer\*\*\*\*\*  
\*\*\*\*\*Enqueue 100 elements\*\*\*\*\*

Took approximately 153505 ns  
\*\*\*\*\*Dequeue 100 elements\*\*\*\*\*  
Took approximately 68828 ns  
\*\*\*\*\*End for 100 number\*\*\*\*\*

\*\*\*\*\*For 1000 integer\*\*\*\*\*  
\*\*\*\*\*Enqueue 1000 elements\*\*\*\*\*  
Took approximately 1336715 ns  
\*\*\*\*\*Dequeue 1000 elements\*\*\*\*\*  
Took approximately 950463 ns  
\*\*\*\*\*End for 1000 number\*\*\*\*\*

\*\*\*\*\*For 10000 integer\*\*\*\*\*  
\*\*\*\*\*Enqueue 10000 elements\*\*\*\*\*  
Took approximately 4 ms  
\*\*\*\*\*Dequeue 10000 elements\*\*\*\*\*  
Took approximately 14 ms  
\*\*\*\*\*End for 10000 number\*\*\*\*\*

\*\*\*\*\*For 100000 integer\*\*\*\*\*  
\*\*\*\*\*Enqueue 100000 elements\*\*\*\*\*  
Took approximately 21 ms  
\*\*\*\*\*Dequeue 100000 elements\*\*\*\*\*  
Took approximately 1923 ms  
\*\*\*\*\*End for 100000 number\*\*\*\*\*

\*\*\*\*\*For 1000000 integer\*\*\*\*\*  
\*\*\*\*\*Enqueue 1000000 elements\*\*\*\*\*  
Took approximately 265 ms  
\*\*\*\*\*Dequeue 1000000 elements\*\*\*\*\*  
Took approximately 274361 ms  
\*\*\*\*\*End for 1000000 number\*\*\*\*\*

-----END\_ARRAYLIST\_QUEUE\_TEST-----

-----LINKEDLIST\_QUEUE\_TEST-----

\*\*\*\*\*For 1 integer\*\*\*\*\*  
\*\*\*\*\*Enqueue 1 element\*\*\*\*\*  
Took approximately 20377 ns  
\*\*\*\*\*Dequeue 1 element\*\*\*\*\*  
Took approximately 26716 ns  
\*\*\*\*\*End for 1 number\*\*\*\*\*

\*\*\*\*\*For 10 integer\*\*\*\*\*  
\*\*\*\*\*Enqueue 10 elements\*\*\*\*\*  
Took approximately 6793 ns  
\*\*\*\*\*Dequeue 10 elements\*\*\*\*\*

```

Took approximately 14943 ns
*****End for 10 numbers*****

*****For 100 integer*****
*****Enqueue 100 elements*****
Took approximately 62941 ns
*****Dequeue 100 elements*****
Took approximately 64753 ns
*****End for 100 number*****

*****For 1000 integer*****
*****Enqueue 1000 elements*****
Took approximately 573266 ns
*****Dequeue 1000 elements*****
Took approximately 499910 ns
*****End for 1000 number*****

*****For 10000 integer*****
*****Enqueue 10000 elements*****
Took approximately 2 ms
*****Dequeue 10000 elements*****
Took approximately 1 ms
*****End for 10000 number*****

*****For 100000 integer*****
*****Enqueue 100000 elements*****
Took approximately 15 ms
*****Dequeue 100000 elements*****
Took approximately 5 ms
*****End for 100000 number*****

*****For 1000000 integer*****
*****Enqueue 1000000 elements*****
Took approximately 267 ms
*****Dequeue 1000000 elements*****
Took approximately 22 ms
*****End for 1000000 number*****

-----END_LINKEDLIST_QUEUE_TEST-----
-----UNSORTED_VECTOR_QUEUE_TEST-----
*****For 1 integer*****
*****Enqueue 1 element*****
Took approximately 13131 ns
*****Dequeue 1 element*****
Took approximately 12226 ns
*****End for 1 number*****

```

```
*****For 10 integer*****
*****Enqueue 10 elements*****
Took approximately 8151 ns
*****Dequeue 10 elements*****
Took approximately 9510 ns
*****End for 10 numbers*****
```

```
*****For 100 integer*****
*****Enqueue 100 elements*****
Took approximately 81960 ns
*****Dequeue 100 elements*****
Took approximately 57960 ns
*****End for 100 number*****
```

```
*****For 1000 integer*****
*****Enqueue 1000 elements*****
Took approximately 662471 ns
*****Dequeue 1000 elements*****
Took approximately 935067 ns
*****End for 1000 number*****
```

```
*****For 10000 integer*****
*****Enqueue 10000 elements*****
Took approximately 3 ms
*****Dequeue 10000 elements*****
Took approximately 12 ms
*****End for 10000 number*****
```

```
*****For 100000 integer*****
*****Enqueue 100000 elements*****
Took approximately 15 ms
*****Dequeue 100000 elements*****
Took approximately 1286 ms
*****End for 100000 number*****
```

```
*****For 1000000 integer*****
*****Enqueue 1000000 elements*****
Took approximately 171 ms
*****Dequeue 1000000 elements*****
Took approximately 261435 ms
*****End for 1000000 number*****
```

```
-----END_UNSORTED_VECTOR_QUEUE_TEST-----
-----BINARY_SEARCH_TREE_QUEUE_TEST-----
```

```
*****For 1 integer*****
*****Enqueue 1 element*****
Took approximately 6792 ns
*****Dequeue 1 element*****
Took approximately 1358 ns
*****End for 1 number*****
```

```
*****For 10 integer*****
*****Enqueue 10 elements*****
Took approximately 3169 ns
*****Dequeue 10 elements*****
Took approximately 2265 ns
*****End for 10 numbers*****
```

```
*****For 100 integer*****
*****Enqueue 100 elements*****
Took approximately 31697 ns
*****Dequeue 100 elements*****
Took approximately 9056 ns
*****End for 100 number*****
```

```
*****For 1000 integer*****
*****Enqueue 1000 elements*****
Took approximately 156675 ns
*****Dequeue 1000 elements*****
Took approximately 155316 ns
*****End for 1000 number*****
```

```
*****For 10000 integer*****
*****Enqueue 10000 elements*****
Took approximately 1 ms
*****Dequeue 10000 elements*****
Took approximately 10 ms
*****End for 10000 number*****
```

```
*****For 100000 integer*****
*****Enqueue 100000 elements*****
Took approximately 23 ms
*****Dequeue 100000 elements*****
Took approximately 1348 ms
*****End for 100000 number*****
```

```
*****For 1000000 integer*****
*****Enqueue 1000000 elements*****
Took approximately 172 ms
*****Dequeue 1000000 elements*****
```

Took approximately 287336 ms

\*\*\*\*\*End for 1000000 number\*\*\*\*\*

-----END\_BINARY\_SEARCH\_TREE\_QUEUE\_TEST-----

-----PART\_3\_END-----

Ödev githup linki

[https://github.com/erccanuca/cse222\\_hw06\\_BinarySeachTree\\_HuffmanTree\\_encoding.git](https://github.com/erccanuca/cse222_hw06_BinarySeachTree_HuffmanTree_encoding.git)

(Ödev teslim süresi geçince public yapılacak.)