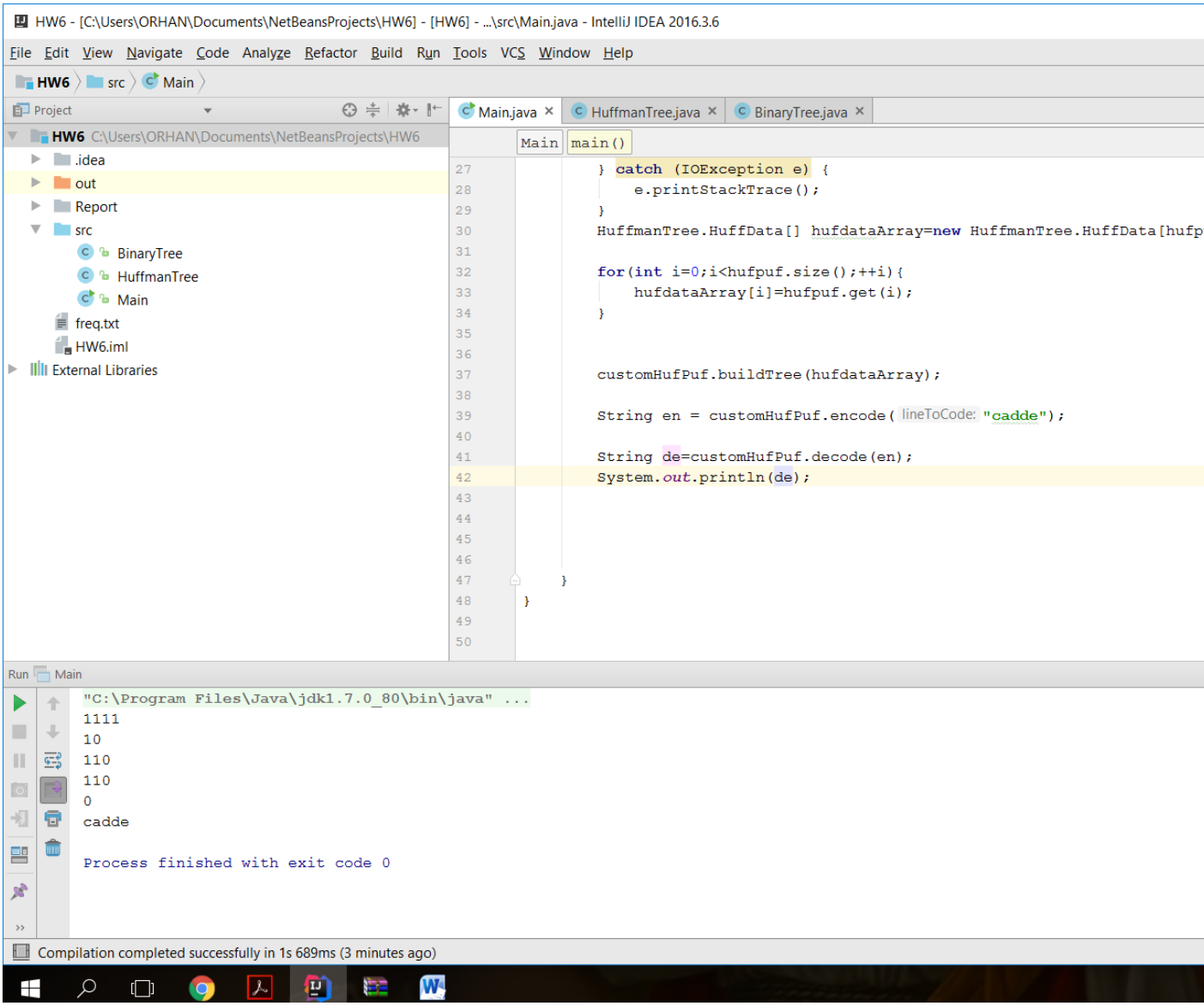
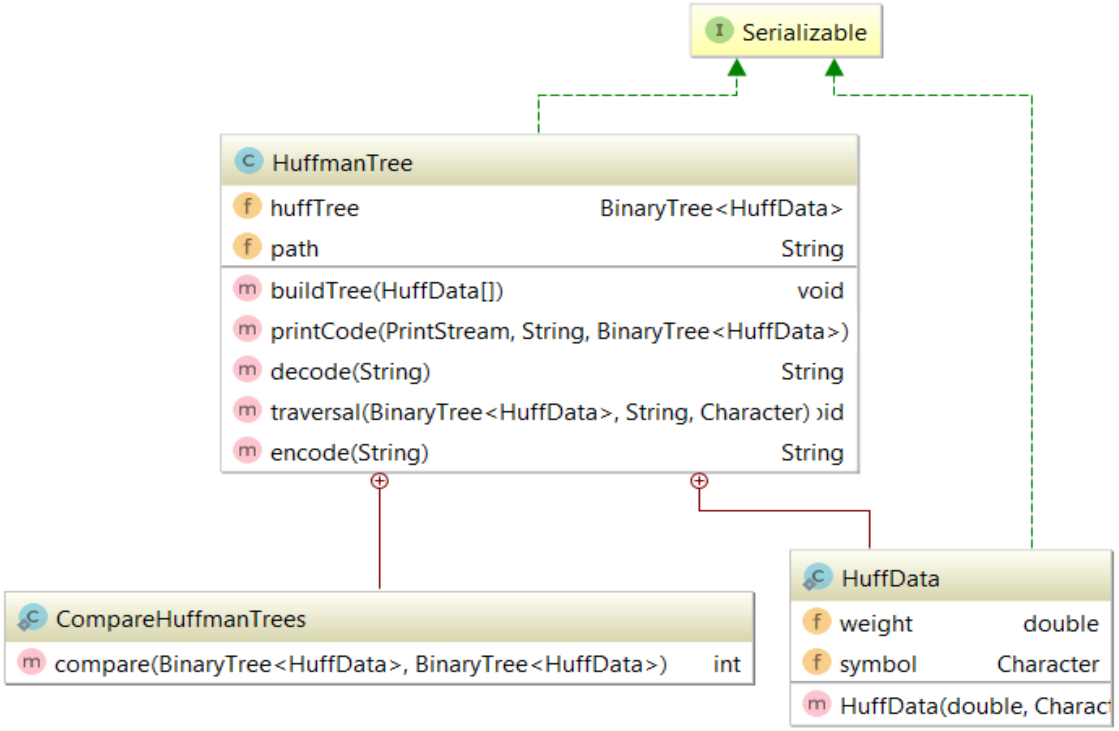
Q2-)Test Case.Frequencies are taken from chapter 6, figure 6.37. b 13 ,c 22,d 32, a 64, e 103.



Class Diagram of HuffmanTree



Problem Solution Approach

I have used global string to assign the path when the char has found.I have modified recursive traversal method.

```
private String path;

protected void traversal(BinaryTree<HuffData> rootNode ,String yol,Character key  ){



  if(rootNode!=null){
     if(rootNode.getData().symbol==key){
        path=yol;
        return;
     }

    String temp=yol;
    traversal(rootNode.getLeftSubtree(),temp+="0",key);
    temp=yol;
    traversal(rootNode.getRightSubtree(),temp+="1",key);
  }

}
```

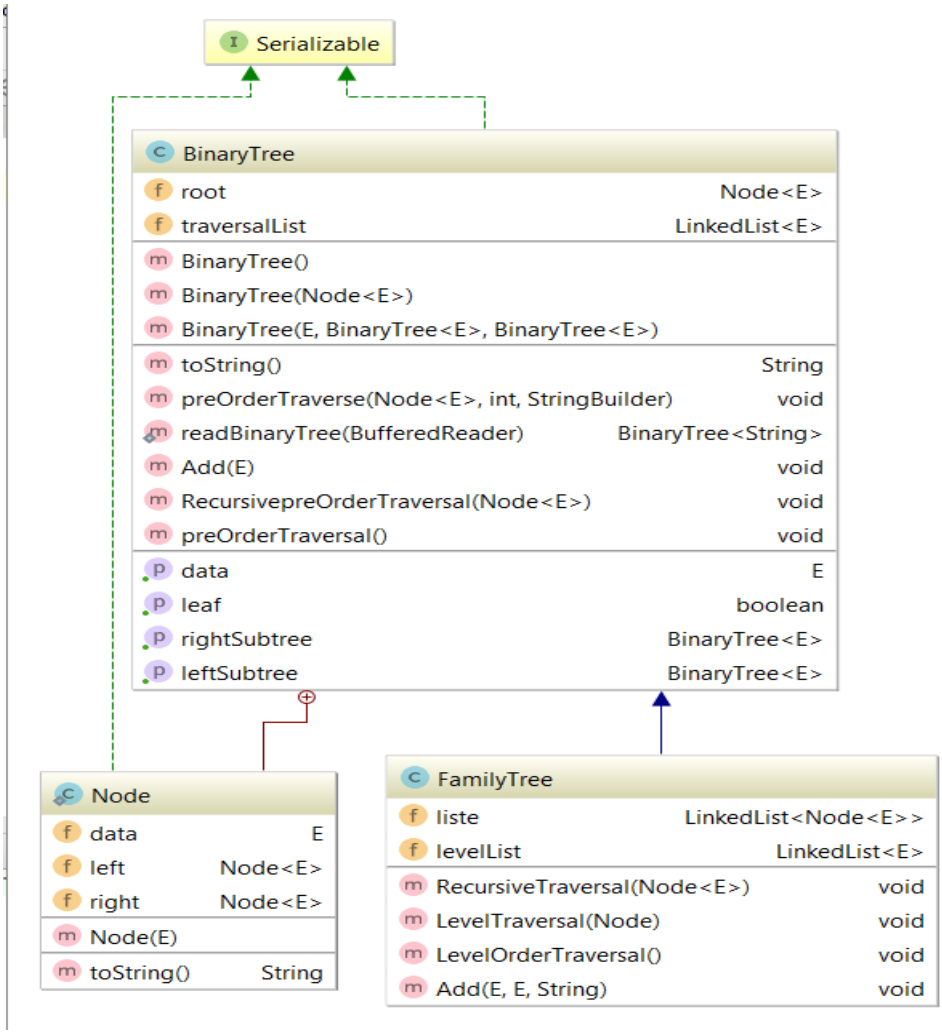
Then i used this method in my encode method.

```
public String encode(String lineToCode){
  StringBuilder result = new StringBuilder();
  BinaryTree < HuffData > currentTree = huffTree;
  for (int i=0;i<lineToCode.length();i++){
    String yol="";
    traversal(currentTree,yol,lineToCode.charAt(i));
    result.append(path);
    System.out.println(path+" ");////Print line.It's okay to remove.
  }
  return result.toString();


}
```

Q3-)Test Case.

```
public class Main {

    public static void main(String[] args) {

        FamilyTree<String> family= new FamilyTree<>();
        try {
            family.Add( person: "Hasan",  Parent: null,  nick: null);
            family.Add( person: "Ayşe",  Parent: "Hasan",  nick: "ebu-Ayşe");
            family.Add( person: "Ali",  Parent: "Ayşe",  nick: "ibn-Hasan");
            family.Add( person: "Sema",  Parent: "Hasan",  nick: "ebu-Ayşe");
        } catch (Exception e) {
            e.printStackTrace();
        }

        family.LevelOrderTraversal();

    }
}
```

Class Diagram.



Problem Solution Approach.

LevelList is a global variable of the class.I have used FIFO approach to put subtrees in a linked list.Idea is getting the leafs of the first root that has entered the linked list.First left then right.

```
protected LinkedList <E> levelList  = new LinkedList<>();


  private void LevelTraversal(Node root){
      LinkedList<Node> level  = new LinkedList<>();
      level.add(root);
      while(!level.isEmpty()){
          Node node = level.poll();

          levelList.add((E) node.data);

          if(node.left!= null)
              level.add(node.left);
          if(node.right!= null)
              level.add(node.right);
      }
  }
```

Then use these methods for our main method.Thats why other method is private.

```
public void LevelOrderTraversal(){//////////////////HW 6 Question 3

    LevelTraversal(super.root);

    Iterator<E> it=levelList.iterator();

    while (it.hasNext())
        System.out.println(it.next());

    levelList.clear();
}
```

It wasn't clear in the whether we were just print all of it or return them as string or return the method as an iterator.So i have used an iterator in the.If necessary i can easily change the method into an iterator.