```java

package edu.caltech.cs2.lab03;

import org.w3c.dom.Attr;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.concurrent.atomic.AtomicReference;

public class DecisionTree {
    private final DecisionTreeNode root;

    public DecisionTree(DecisionTreeNode root) {
        this.root = root;
    }

    public String predict(Dataset.Datapoint point) {
        return this.predictHelper(point, this.root);
    }

    private String predictHelper(Dataset.Datapoint point, DecisionTreeNode curr) {
        // Node is an outcome node
        if (curr.isLeaf()) {
           return ((OutcomeNode) curr).outcome;
        }

        // Node is an attribute node
        String treeAttribute = ((AttributeNode) curr).attribute;
        String pointFeature = point.attributes.get(treeAttribute);
        curr = ((AttributeNode) curr).children.get(pointFeature);

        return predictHelper(point, curr);
    }

    public static DecisionTree id3(Dataset dataset, List<String> attributes) {
        return new DecisionTree(id3helper(dataset, attributes));
    }

    private static DecisionTreeNode id3helper(Dataset data, List<String> attributes) {
        // If all remaining data points have same outcome, return OutcomeNode with this outcome
        String outcome = data.pointsHaveSameOutcome();
        if (outcome.length() > 0) {
            return new OutcomeNode(outcome);
        }

        // If no attributes left, return OutcomeNode with most common outcome
        if (attributes.size() == 0) {
            String mostCommonOutcome = data.getMostCommonOutcome();
            return new OutcomeNode(mostCommonOutcome);
        }


```

```
        // Find the lowest entropy attribute, a.
        String a = data.getAttributeWithMinEntropy(attributes);

        // For each feature, f, of a:
        List<String> lowestEntropyFeatures = data.getFeaturesForAttribute(a);
        Map<String, DecisionTreeNode> attributeToNode = new HashMap<>();
        for (String f : lowestEntropyFeatures) {

            // Find all the data points that have the feature f
            Dataset newDSet = data.getPointsWithFeature(f);

            // If there are none
            if (newDSet.isEmpty()) {

                // Guess! Make an outcome child with the most common outcome.
                String mostCommonOutcome = newDSet.getMostCommonOutcome();
                OutcomeNode newNode = new OutcomeNode(mostCommonOutcome);
                attributeToNode.put(f, newNode);

            // Otherwise:
            } else {

                // Use the data! Recursively make a child out of the remaining data points, using all at
                List<String> newAttributes = new ArrayList<>();
                for (String attribute : attributes) {
                    if (!attribute.equals(a)) {
                        newAttributes.add(attribute);
                    }
                }
                attributeToNode.put(f, id3helper(newDSet, newAttributes));
            }
        }

        // Return the attribute node with the children generated above.
        return new AttributeNode(a, attributeToNode);
    }
}
```