NodeStore (H1)

Based on the codes from the book:
Artificial Intelligence : A Modern Approach
The copyrights of the codes belong to
Ravi Mohan, Peter Norvig, Stuart Russell, Ciaran O'Reilly

```
package aima.search.framework;

import java.util.ArrayList;
import java.util.List;

/**
 * @author Ravi Mohan
 *
 */

public class NodeExpander {
	protected Metrics metrics;

	protected static String NODES_EXPANDED = "nodesExpanded";

	public NodeExpander() {
		metrics = new Metrics();
	}

	public void clearInstrumentation() {
		metrics.set(NODES_EXPANDED, 0);
	}

	public List<Node> expandNode(Node node, Problem problem) {

		List<Node> nodes = new ArrayList<Node>();
		List successors = problem.getSuccessorFunction().getSuccessors(
				node.getState());
		for (int i = 0; i < successors.size(); i++) {
			Successor successor = (Successor) successors.get(i);
			Node aNode = new Node(node, successor.getState());
			aNode.setAction(successor.getAction());
			Double stepCost = problem.getStepCostFunction().calculateStepCost(
					node.getState(), successor.getState(),
					successor.getAction());
			aNode.setStepCost(stepCost);
			aNode.addToPathCost(stepCost);
			nodes.add(aNode);

		}
		metrics.set(NODES_EXPANDED, metrics.getInt(NODES_EXPANDED) + 1);
		// System.out.println("Nodes expanded = " +
		// metrics.getInt(NODES_EXPANDED));
		return nodes;
	}

	public int getNodesExpanded() {
		return metrics.getInt(NODES_EXPANDED);
	}

	public void setNodesExpanded(int nodesExpanded) {
		metrics.set(NODES_EXPANDED, nodesExpanded);
	}

	public Object getSearchMetric(String name) {
		return metrics.get(name);
	}

	public Metrics getMetrics() {
		return metrics;
	}
}
```

expandNode();
getNodesExpanded();
setNodesExpanded();
getSearchMetric();
getMetrics()