|  |
| --- |
| package cz.cvut.fit.zum.alg;  import cz.cvut.fit.zum.api.AbstractAlgorithm;  import java.util.List;  import cz.cvut.fit.zum.api.Node;  import cz.cvut.fit.zum.api.UninformedSearch;  import java.util.ArrayList;  import java.util.Collections;  import java.util.HashSet;  import java.util.LinkedList;  import org.openide.util.lookup.ServiceProvider;  @ServiceProvider(service = AbstractAlgorithm.class, position = 1)  public class RandomSearch extends AbstractAlgorithm implements UninformedSearch {  private HashSet<Node> closed;  private List<Node> opened;  @Override  public String getName() {  return "random search";  }  @Override  public List<Node> findPath(Node startNode) {  opened = Collections.synchronizedList(new LinkedList<Node>());  closed = new HashSet<Node>();  List<Node> path = new ArrayList<Node>();  expand(startNode);  Node current = random(opened);  closed.add(current);  while (!current.isTarget()) {  expand(current);  closed.add(current);  current = random(opened);  }  return path;  }  private synchronized void expand(Node current) {  List<Node> list = current.expand();  for (Node n : list) {  if (!closed.contains(n)) {  opened.add(n);  }  }  }  private Node random(List<Node> list) {  int min = 0;  int max = list.size();    if (max == 1) {  return list.remove(0);  }  int num = min + (int) (Math.random() \* ((max - min)));    //we want to remove explored nodes  return list.remove(num);  }  } |