|  | /\* |
| --- | --- |
|  | \* AP(r) Computer Science GridWorld Case Study: |
|  | \* Copyright(c) 2002-2006 College Entrance Examination Board |
|  | \* (http://www.collegeboard.com). |
|  | \* |
|  | \* This code is free software; you can redistribute it and/or modify |
|  | \* it under the terms of the GNU General Public License as published by |
|  | \* the Free Software Foundation. |
|  | \* |
|  | \* This code is distributed in the hope that it will be useful, |
|  | \* but WITHOUT ANY WARRANTY; without even the implied warranty of |
|  | \* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the |
|  | \* GNU General Public License for more details. |
|  | \* |
|  | \* @author Alyce Brady |
|  | \* @author APCS Development Committee |
|  | \* @author Cay Horstmann |
|  | \*/ |
|  |  |
|  | package info.gridworld.grid; |
|  |  |
|  | import java.util.ArrayList; |
|  |  |
|  | import java.util.\*; |
|  |  |
|  | /\*\* |
|  | \* An <code>UnboundedGrid</code> is a rectangular grid with an unbounded number of rows and |
|  | \* columns. <br /> |
|  | \* The implementation of this class is testable on the AP CS AB exam. |
|  | \*/ |
|  | public class UnboundedGrid<E> extends AbstractGrid<E> |
|  | { |
|  | private Map<Location, E> occupantMap; |
|  |  |
|  | /\*\* |
|  | \* Constructs an empty unbounded grid. |
|  | \*/ |
|  | public UnboundedGrid() |
|  | { |
|  | occupantMap = new HashMap<Location, E>(); |
|  | } |
|  |  |
|  | public int getNumRows() |
|  | { |
|  | return -1; |
|  | } |
|  |  |
|  | public int getNumCols() |
|  | { |
|  | return -1; |
|  | } |
|  |  |
|  | public boolean isValid(Location loc) |
|  | { |
|  | return true; |
|  | } |
|  |  |
|  | public ArrayList<Location> getOccupiedLocations() |
|  | { |
|  | ArrayList<Location> a = new ArrayList<Location>(); |
|  | for (Location loc : occupantMap.keySet()) |
|  | a.add(loc); |
|  | return a; |
|  | } |
|  |  |
|  | public E get(Location loc) |
|  | { |
|  | if (loc == null) |
|  | throw new NullPointerException("loc == null"); |
|  | return occupantMap.get(loc); |
|  | } |
|  |  |
|  | public E put(Location loc, E obj) |
|  | { |
|  | if (loc == null) |
|  | throw new NullPointerException("loc == null"); |
|  | if (obj == null) |
|  | throw new NullPointerException("obj == null"); |
|  | return occupantMap.put(loc, obj); |
|  | } |
|  |  |
|  | public E remove(Location loc) |
|  | { |
|  | if (loc == null) |
|  | throw new NullPointerException("loc == null"); |
|  | return occupantMap.remove(loc); |
|  | } |
|  | } |