|  | /\* |
| --- | --- |
|  | \* 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; |
|  |  |
|  | /\*\* |
|  | \* <code>Grid</code> provides an interface for a two-dimensional, grid-like |
|  | \* environment containing arbitrary objects. <br /> |
|  | \* This interface is testable on the AP CS A and AB exams. |
|  | \*/ |
|  | public interface Grid<E> |
|  | { |
|  | /\*\* |
|  | \* Returns the number of rows in this grid. |
|  | \* @return the number of rows, or -1 if this grid is unbounded |
|  | \*/ |
|  | int getNumRows(); |
|  |  |
|  | /\*\* |
|  | \* Returns the number of columns in this grid. |
|  | \* @return the number of columns, or -1 if this grid is unbounded |
|  | \*/ |
|  | int getNumCols(); |
|  |  |
|  | /\*\* |
|  | \* Checks whether a location is valid in this grid. <br /> |
|  | \* Precondition: <code>loc</code> is not <code>null</code> |
|  | \* @param loc the location to check |
|  | \* @return <code>true</code> if <code>loc</code> is valid in this grid, |
|  | \* <code>false</code> otherwise |
|  | \*/ |
|  | boolean isValid(Location loc); |
|  |  |
|  | /\*\* |
|  | \* Puts an object at a given location in this grid. <br /> |
|  | \* Precondition: (1) <code>loc</code> is valid in this grid (2) |
|  | \* <code>obj</code> is not <code>null</code> |
|  | \* @param loc the location at which to put the object |
|  | \* @param obj the new object to be added |
|  | \* @return the object previously at <code>loc</code> (or <code>null</code> |
|  | \* if the location was previously unoccupied) |
|  | \*/ |
|  | E put(Location loc, E obj); |
|  |  |
|  | /\*\* |
|  | \* Removes the object at a given location from this grid. <br /> |
|  | \* Precondition: <code>loc</code> is valid in this grid |
|  | \* @param loc the location of the object that is to be removed |
|  | \* @return the object that was removed (or <code>null<code> if the location |
|  | \* is unoccupied) |
|  | \*/ |
|  | E remove(Location loc); |
|  |  |
|  | /\*\* |
|  | \* Returns the object at a given location in this grid. <br /> |
|  | \* Precondition: <code>loc</code> is valid in this grid |
|  | \* @param loc a location in this grid |
|  | \* @return the object at location <code>loc</code> (or <code>null<code> |
|  | \* if the location is unoccupied) |
|  | \*/ |
|  | E get(Location loc); |
|  |  |
|  | /\*\* |
|  | \* Gets the locations in this grid that contain objects. |
|  | \* @return an array list of all occupied locations in this grid |
|  | \*/ |
|  | ArrayList<Location> getOccupiedLocations(); |
|  |  |
|  | /\*\* |
|  | \* Gets the valid locations adjacent to a given location in all eight |
|  | \* compass directions (north, northeast, east, southeast, south, southwest, |
|  | \* west, and northwest). <br /> |
|  | \* Precondition: <code>loc</code> is valid in this grid |
|  | \* @param loc a location in this grid |
|  | \* @return an array list of the valid locations adjacent to <code>loc</code> |
|  | \* in this grid |
|  | \*/ |
|  | ArrayList<Location> getValidAdjacentLocations(Location loc); |
|  |  |
|  | /\*\* |
|  | \* Gets the valid empty locations adjacent to a given location in all eight |
|  | \* compass directions (north, northeast, east, southeast, south, southwest, |
|  | \* west, and northwest). <br /> |
|  | \* Precondition: <code>loc</code> is valid in this grid |
|  | \* @param loc a location in this grid |
|  | \* @return an array list of the valid empty locations adjacent to |
|  | \* <code>loc</code> in this grid |
|  | \*/ |
|  | ArrayList<Location> getEmptyAdjacentLocations(Location loc); |
|  |  |
|  | /\*\* |
|  | \* Gets the valid occupied locations adjacent to a given location in all |
|  | \* eight compass directions (north, northeast, east, southeast, south, |
|  | \* southwest, west, and northwest). <br /> |
|  | \* Precondition: <code>loc</code> is valid in this grid |
|  | \* @param loc a location in this grid |
|  | \* @return an array list of the valid occupied locations adjacent to |
|  | \* <code>loc</code> in this grid |
|  | \*/ |
|  | ArrayList<Location> getOccupiedAdjacentLocations(Location loc); |
|  |  |
|  | /\*\* |
|  | \* Gets the neighboring occupants in all eight compass directions (north, |
|  | \* northeast, east, southeast, south, southwest, west, and northwest). |
|  | \* <br /> |
|  | \* Precondition: <code>loc</code> is valid in this grid |
|  | \* @param loc a location in this grid |
|  | \* @return returns an array list of the objects in the occupied locations |
|  | \* adjacent to <code>loc</code> in this grid |
|  | \*/ |
|  | ArrayList<E> getNeighbors(Location loc); |
|  | } |