| /\* |  |
| --- | --- |
|  | \* AP(r) Computer Science GridWorld Case Study: |
|  | \* Copyright(c) 2005-2006 Cay S. Horstmann (http://horstmann.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 Cay Horstmann |
|  | \*/ |
|  |  |
|  | package info.gridworld.actor; |
|  |  |
|  | import info.gridworld.grid.Grid; |
|  | import info.gridworld.grid.Location; |
|  |  |
|  | import java.awt.Color; |
|  |  |
|  | /\*\* |
|  | \* An <code>Actor</code> is an entity with a color and direction that can act. |
|  | \* <br /> |
|  | \* The API of this class is testable on the AP CS A and AB exams. |
|  | \*/ |
|  | public class Actor |
|  | { |
|  | private Grid<Actor> grid; |
|  | private Location location; |
|  | private int direction; |
|  | private Color color; |
|  |  |
|  | /\*\* |
|  | \* Constructs a blue actor that is facing north. |
|  | \*/ |
|  | public Actor() |
|  | { |
|  | color = Color.BLUE; |
|  | direction = Location.NORTH; |
|  | grid = null; |
|  | location = null; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* Gets the color of this actor. |
|  | \* @return the color of this actor |
|  | \*/ |
|  | public Color getColor() |
|  | { |
|  | return color; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* Sets the color of this actor. |
|  | \* @param newColor the new color |
|  | \*/ |
|  | public void setColor(Color newColor) |
|  | { |
|  | color = newColor; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* Gets the current direction of this actor. |
|  | \* @return the direction of this actor, an angle between 0 and 359 degrees |
|  | \*/ |
|  | public int getDirection() |
|  | { |
|  | return direction; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* Sets the current direction of this actor. |
|  | \* @param newDirection the new direction. The direction of this actor is set |
|  | \* to the angle between 0 and 359 degrees that is equivalent to |
|  | \* <code>newDirection</code>. |
|  | \*/ |
|  | public void setDirection(int newDirection) |
|  | { |
|  | direction = newDirection % Location.FULL\_CIRCLE; |
|  | if (direction < 0) |
|  | direction += Location.FULL\_CIRCLE; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* Gets the grid in which this actor is located. |
|  | \* @return the grid of this actor, or <code>null</code> if this actor is |
|  | \* not contained in a grid |
|  | \*/ |
|  | public Grid<Actor> getGrid() |
|  | { |
|  | return grid; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* Gets the location of this actor. |
|  | \* @return the location of this actor, or <code>null</code> if this actor is |
|  | \* not contained in a grid |
|  | \*/ |
|  | public Location getLocation() |
|  | { |
|  | return location; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* Puts this actor into a grid. If there is another actor at the given |
|  | \* location, it is removed. <br /> |
|  | \* Precondition: (1) This actor is not contained in a grid (2) |
|  | \* <code>loc</code> is valid in <code>gr</code> |
|  | \* @param gr the grid into which this actor should be placed |
|  | \* @param loc the location into which the actor should be placed |
|  | \*/ |
|  | public void putSelfInGrid(Grid<Actor> gr, Location loc) |
|  | { |
|  | if (grid != null) |
|  | throw new IllegalStateException( |
|  | "This actor is already contained in a grid."); |
|  |  |
|  | Actor actor = gr.get(loc); |
|  | if (actor != null) |
|  | actor.removeSelfFromGrid(); |
|  | gr.put(loc, this);r |
|  | grid = gr; |
|  | location = loc; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* Removes this actor from its grid. <br /> |
|  | \* Precondition: This actor is contained in a grid |
|  | \*/ |
|  | public void removeSelfFromGrid() |
|  | { |
|  | if (grid == null) |
|  | throw new IllegalStateException( |
|  | "This actor is not contained in a grid."); |
|  | if (grid.get(location) != this) |
|  | throw new IllegalStateException( |
|  | "The grid contains a different actor at location " |
|  | + location + "."); |
|  |  |
|  | grid.remove(location); |
|  | grid = null; |
|  | location = null; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* Moves this actor to a new location. If there is another actor at the |
|  | \* given location, it is removed. <br /> |
|  | \* Precondition: (1) This actor is contained in a grid (2) |
|  | \* <code>newLocation</code> is valid in the grid of this actor |
|  | \* @param newLocation the new location |
|  | \*/ |
|  | public void moveTo(Location newLocation) |
|  | { |
|  | if (grid == null) |
|  | throw new IllegalStateException("This actor is not in a grid."); |
|  | if (grid.get(location) != this) |
|  | throw new IllegalStateException( |
|  | "The grid contains a different actor at location " |
|  | + location + "."); |
|  | if (!grid.isValid(newLocation)) |
|  | throw new IllegalArgumentException("Location " + newLocation |
|  | + " is not valid."); |
|  |  |
|  | if (newLocation.equals(location)) |
|  | return; |
|  | grid.remove(location); |
|  | Actor other = grid.get(newLocation); |
|  | if (other != null) |
|  | other.removeSelfFromGrid(); |
|  | location = newLocation; |
|  | grid.put(location, this); |
|  | } |
|  |  |
|  | /\*\* |
|  | \* Reverses the direction of this actor. Override this method in subclasses |
|  | \* of <code>Actor</code> to define types of actors with different behavior |
|  | \* |
|  | \*/ |
|  | public void act() |
|  | { |
|  | setDirection(getDirection() + Location.HALF\_CIRCLE); |
|  | } |
|  |  |
|  | /\*\* |
|  | \* Creates a string that describes this actor. |
|  | \* @return a string with the location, direction, and color of this actor |
|  | \*/ |
|  | public String toString() |
|  | { |
|  | return getClass().getName() + "[location=" + location + ",direction=" |
|  | + direction + ",color=" + color + "]"; |
|  | } |
|  | } |