Homework 3 Testing Strategy
Greg Attra
CS 5010 Prof. Freifield

DeadEndNode
This is the “empty node” in the linked maze. This class has no properties and therefore no state. All methods have a hard-coded response that ignores the input passed to them. So a single test suffices for the following methods:
- getGoldCount()
- getThiefPenalty()
- getCoordinates()
- getNode(Direction dir)
- canReach(Coordinates c)
- wealthiestPathTo(Coordinates c)
- exploreTo(Coordinates c)
- toString()

RoomNode - Standard, Gold, Thief
The Node is the core of the maze program. Much of the functionality for how the maze is built and how a player moves through it resides in the Node classes.

a) A Node has four exits: North, South, East and West. When a node is initialized, it has 4 empty nodes at each exit.
b) A node can accept another node and direction as input and set that node at the corresponding exit.
c) A node can get another node that it connects to given the target node’s coordinates.
d) A node can take a target coordinate set and find the wealthiest path through the linked maze to that target node. If the node is out of bounds of the maze, a DeadEndNode() is returned.
e) A node can traverse the entire linked maze reaching every node.
f) A node can take a coordinate set and return a boolean indicating whether the node is connected to a node at that coordinate set.
g) A node can take a coordinate set and find a random path to that coordinate set.
h) A node can recursively grow() itself into a maze, given a configuration object.
    i) A node can produce a wrapping maze or non-wrapping maze
    ii) A node can produce a perfect maze or a room maze
i) A node can have gold, which the player picks up upon entering the node.
j) A node can have a thief, which steals some of the player’s gold upon entering the node.

Path
The path object is used to track the nodes traversed when navigating the maze. It is used by the nodes as well to avoid infinite loops in their recursive traversals of the linked maze.
a) The path is instantiated with a target coordinate set
b) The path has a boolean attribute indicating whether the path reaches the target

c) The path keeps a running list of coordinates of the nodes it traverses
d) The path keeps a tally of the gold acquired and stolen by thieves as it traverses the maze
e) The path exposes its current state through getter methods

Edge

The Edge object captures a wall between two nodes. A node produces an edge whenever it attempts to grow into a coordinate where there already exists another node. These objects are provided to the Kruskal's algorithm which is used to make a room maze.

a) An Edge object takes a "tail" node and a "head" node, as well as the respective Direction relative to the center of the Edge (the center of the wall)

Maze

The Maze object is a wrapper ADT around the maze Node's. It has a start and an end Node, and keeps track of the current node (where the "pointer" is located).

a) Test the getters to make sure the maze is instantiated properly
b) Test move() which takes a Direction and moves the pointer in that direction.
    i) The current node is updated to the new node if the move was successful.
    ii) If the node at the specified direction is a dead end, returns false to the caller

Configuration

The MazeConfiguration object is used by the nodes to determine how they should grow() themselves into the maze. It also tracks the current state of the maze as it is being built by the nodes.

a) A MazeConfiguration holds the following config information:
    i) ColumnCount
    ii) RowCount
    iii) Start coordinates
    iv) Goal coordinates
    v) The random seed used to generate the maze
    vi) The random object used to generate the maze
    vii) ThiefPenalty
    viii) ThiefFrequency
    ix) GoldFrequency
    x) GoldAmount
    xi) IsWrappingMaze?
    xii) IsRoomMaze?
    xiii) The number of edges that would result in a perfect maze
    xiv) The target edge count for Kruskal's algorithm (for room mazes)
b) A MazeConfiguration also tracks the state of the maze as it is being built:
    i) Visited (a 2D array of Node objects - Node[rowCount][columnCount])
    ii) A list of Edges produced by the nodes
    iii) The running number of exits set on the nodes

    - iv) The number of nodes with gold
    - v) The number of nodes with a thief
- c) The Configuration object exposes functionality to assist the nodes in building the maze:
    - i) A method to add an edge to a maze
    - ii) A method to generate a new node at a coordinate set
    - iii) A method to apply Kruskal's algorithm to create a room maze

Builder
The maze builder is a helper class which simplifies the maze construction process. It is recommended to use this class to instantiate a maze as it abstracts much of the complexity.

- a) The builder allows the user to specify the following maze configurations:
    - i) ColumnCount
    - ii) RowCount
    - iii) Start coordinates
    - iv) Goal coordinates
    - v) The random seed used to generate the maze
    - vi) The random object used to generate the maze
    - vii) ThiefPenalty
    - viii) ThiefFrequency
    - ix) GoldFrequency
    - x) GoldAmount
    - xi) IsWrappingMaze?
    - xii) IsRoomMaze?
    - xiii) The number of edges that would result in a perfect maze
    - xiv) The target edge count for Kruskal's algorithm (for room mazes)
- b) The builder's core method build() uses those configurations to create a Maze. The tests should assert that the final maze matches the expected / configurations.

Player
The Player object represents the real-world player of this maze program. The player moves through the maze and collects gold or gets robbed by thieves until they reach the goal.

- a) A player is constructed with a name
- b) When a player enters a node, the player's gold count is either incremented if the node is a gold node, decremented if the node is a thief node, or unchanged if it is a standard node

Game
The Game object translates the inputs from the user to move the maze pointer and update the player state. It also uses a Path object to determine where the player has already traversed.

- a) Instantiated with a Player instance and a Maze instance
- b) MovePlayer() method attempts to move the maze pointer in the specified direction
    - i) If success, player enters room and loots it / gets robbed / nothing
    - ii) If fail, return false to caller / user
- c) Start() method places the player in the starting node and adds that node to the Path