Markov Chain Simulation Library

Specifications

Background:

This library is intended to simulate and visualize a Markov decision process represented as a Markov chain. To develop, test, and understand this library you do not need in-depth knowledge of what a Markov chain is. Although, a primer on Markov chains is also provided alongside this specification.

The key idea is that a Markov chain has one "active state" among a group of "possible states". At each time step of the simulation (discrete steps, the actual time units does not matter), the chain randomly transitions from its active state into a new state, which becomes the new active state. While the transition is random, the probability of transitioning to each state is encoded as the Markov chain's transition matrix.

Goal:

The ultimate goal is to visualize a *random walk* generated by a Markov chain. Details on the walking, simulation, and visualization will come with Assignment 3. For now, the (potentially buggy) classes in the library merely encode a Markov chain along with a two-dimensional Cartesian coordinate pair.

Class Summary:

- `FloatMatrix`: a class for matrices of floating point numbers
- `MarkovChain`: a class for encoding a Markov chain
- `Coordinate`: a class for encoding two-dimensional Cartesian coordinates

FloatMatrix

The `FloatMatrix` class represents a matrix of floating point numbers. Each entry of the matrix is encoded as a float. Matrices must have at least one row and at least one column. A `FloatMatrix` can be created from an array of floating point numbers. This class allows for getting and setting individual entries of the matrix. It also provides functionality for multiplying matrices together, where the dimensions of the input matrices are compatible for multiplication.

MarkovChain

The `MarkovChain` class represents a Markov chain. It is encoded as a transition matrix, implemented as a `FloatMatrix`, and a "current state" instance variable. The transition matrix must be square and must have the entries of each row sum to 1. The number of states in the chain is equal to the number of

rows in its transition matrix. A class method is provided to validate if a `FloatMatrix` is a valid transition matrix.

The chain transitions to its next state only when a client prompts it to do so by calling its `nextState()` instance method. Getter methods are provided for its current state and number of possible states. Clients can manually set the active state by calling `setState().`

Coordinate

The Coordinate class encodes a pair of integers representing the possible integer coordinates on a two-dimensional Cartesian plane. Because the coordinates are integers, this is really coordinates of a two-dimensional *integer lattice* (fancy words for 2D points with only integer numbers). This class provides functionality for adding and subtracting coordinates. It also provides the ability to scale (multiply by a constant) a coordinate by a scalar number.