Introduction to Computer Programming (CS 1323) Project 9

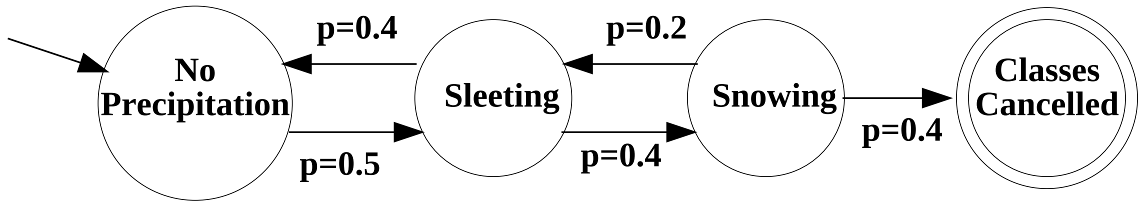
Instructor: Andrew H. Fagg

November 30, 2014

Introduction

In project 6, we developed a simple model of a Markov Chain. In this project, we will develop an object-based implementation that will allow multiple probabilistic transitions from any given state.

For example, we might want to represent the following Markov Chain model of Oklahoma weather and OU weather policy:



From the **Sleeting** state, we transition to the **No Precipitation** state with probability $p = 0.4$ and to the **Snowing** state with probability $p = 0.4$ (and stay in this state with probability $p = 0.2$). Likewise, from the **Snowing** state, we can transition to either the **Sleeting** or **Classes Cancelled** states.

Objectives

By the end of this project, you will be able to:

1. augment existing code to solve a problem,
2. implement parts of a class, and

3. write a loop that iterates until appropriate criteria are met.

State Class

State.java contains a partial implementation of the **State** class. Each state will contain the following information:

- The name of state (a String)
- A list of transition probabilities (an ArrayList)
- A list of cumulative probabilities (an ArrayList)
- A list of next states (an ArrayList)
- A terminal flag (a boolean)

The implementation of each of these instance variables is provided in the java file. Each ArrayList has one element for each transition that we will represent.

Entry i of the cumulative probability array will always be the sum of probabilities $0, 1, ..., i$ (your code will need to enforce this).

Project Requirements

1. Import State.java into your Eclipse environment. This can be found in the projects section of the class web page.
2. Complete the implementation of the State() constructor.
3. Provide an implementation of addTransition().
4. Complete the implementation of nextState().
5. Copy your implementation of getNonNegativeInt() from project 7.
6. In main(): add code that implements the state transitions and terminals for the above Markov Chain.
7. In main(): add the implementation for traversing the Markov Chain starting from $s0$.

Examples

Here are some example inputs and corresponding outputs from a properly executing program. You should carefully consider the cases with which you test your program. In particular, think about the key “boundary cases” (the cases that cause your code to do one thing or another, based on very small differences in input). User inputs are shown in bold.

Example 1

```
 No Precipitation:
                              0.5                 0.5                   Sleeting
 Sleeting:
                              0.4                 0.4                   No Precipitation
                              0.4                 0.8                   Snowing
 Snowing:
                              0.2                 0.2                   Sleeting
                              0.4                 0.6000000000000001    Classes Cancelled
 Classes Cancelled (Terminal):
                              NO TRANSITIONS
########################

Enter a seed: 2
No Precipitation
No Precipitation
No Precipitation
Sleeting
Sleeting
Sleeting
Sleeting
No Precipitation
Sleeting
Snowing
Snowing
Sleeting
Snowing
Snowing
Sleeting
Snowing
```

```
Classes Cancelled

    (program ends)
```

Example 2

```
 No Precipitation:
                                    0.5                   0.5                  Sleeting
 Sleeting:
                                    0.4                   0.4                  No Precipitation
                                    0.4                   0.8                  Snowing
 Snowing:
                                    0.2                   0.2                  Sleeting
                                    0.4                   0.6000000000000001   Classes Cancelled
 Classes Cancelled (Terminal):
                                    NO TRANSITIONS
######################

Enter a seed: done
Enter a seed: -5
Only non-negative values are allowed: done
Enter a seed: 4
No Precipitation
No Precipitation
No Precipitation
No Precipitation
No Precipitation
Sleeting
No Precipitation
No Precipitation
No Precipitation
Sleeting
Sleeting
Snowing
Sleeting
Snowing
Snowing
```

```
Snowing
Sleeting
No Precipitation
No Precipitation
Sleeting
No Precipitation
No Precipitation
No Precipitation
No Precipitation
No Precipitation
No Precipitation
Sleeting
No Precipitation
Sleeting
Sleeting
Snowing
Classes Cancelled
```

(program ends)

Project Documentation

Follow the same documentation procedures as we used in project 1. In particular, you must include:

- Java source file documentation (top of file)
- Method-level documentation
- Inline documentation

For full credit, you must execute Javadoc on your source file and include the results (in the *doc* directory) in your zip file.

Hints

In project 6, we had one probabilistic transition to consider at most for each state. Our code decided to perform a transition using logic such as this:

```
double value = random.nextDouble();
if(value < p)
  do_transition();
```

Here, if $0 \leq value < p$, then the transition is performed.

When we have multiple transitions to consider, we will still use a single sample (the value) from the random object.

For multiple transitions (with probabilities $p_0, p_1, ...p_{n-1}$), the transition to the k^{th} state in the list will happen if:

$p_0 + p_1 + ... + p_{k-1} \leq value < p_0 + p_1 + ... + p_k$

Think about how this relates to your cumulativeProbabity instance variable.

Project Submission

This project must be submitted no later than 2:00pm on Tuesday, December 2nd to the project 9 D2L dropbox. The file must be called *Project9.zip* and be generated in the same way as for prior projects.

Code Review

For the office hour sessions surrounding the project deadline, we will put together a “Doodle Poll” through which you can sign up for a 5-minute code review appointment. During this review, we will execute test examples and review the code with you. If you fail to show up for your reserved time, then you will forfeit your appointment and you must attend at a later time that is not already reserved by someone else.

If either the instructor or TA are free during office hours (or another mutually-agreed upon time), then we are happy to do code reviews, as well as provide general help.

We will only perform reviews on code that has already been submitted to the dropbox on D2L. Please prepare ahead of time for your code review by performing this submission step.

Code reviews must be completed no later than Tuesday, December 9th. If you fail to do a code review, then you will receive a score of zero for the project.

Anyone receiving a 90 or greater on Project 7 may opt to do an “offline” code review (which does not require your presence). Send the instructor email once you have submitted your project in order to schedule an offline review.

Rubric

The project will be graded out of 100 points. The distribution is as follows:

Implementation: 35 points

Program formatting: 15 points

(15) The program is properly formatted (including indentation, curly brace and semicolon locations).

(8) There is one problem with program formatting.

(0) The program is not properly formatted.

Data types and method calls: 10 points

(10) The program is using proper data types and method calls.

(5) There is one error in data type or method call selection.

(0) There are multiple errors in data type and method call selection.

Required Methods: 10 points

(10) All of the required methods are implemented correctly.

(0) The required methods are not implemented correctly.

Proper Execution: 30 points

Output: 15 points

(15) The program passes all tests.

(12) The program fails one test.

(9) The program fails two tests.

(6) The program fails three tests.

(3) The program fails four tests.

(0) The program fails five or more tests.

Execution: 15 points

(15) The program executes with no errors (thrown exceptions).

(8) The program executes, but there is one minor error.

(0) The program does not execute.

Documentation and Submission: 35 points

Project Documentation: 5 points

(5) The java file contains all of the required documentation elements at the top of the file.

(3) The java file is missing one of the required documentation elements.

(2) The java file is missing two of the required documentation elements.

(0) The java file is missing more than two of the required documentation elements.

Method-Level Documentation: 10 points

(10) Every method contains all of the required documentation elements ahead of the method prototype, and the Javadocs have been generated and included in the submission.

(7) The method documentation is missing one of the required documentation elements.

(3) The method documentation is missing two of the required documentation elements, or the Javadocs have not been submitted.

(0) The method documentation is missing more than two of the required documentation elements.

Inline: 10 points

(10) Every method contains appropriate inline documentation.

(7) There is one missing or incorrect line of inline documentation.

(3) There are two missing or incorrect lines of inline documentation.

(0) There are more than two missing or incorrect lines of inline documentation.

Submission: 10 points

(10) The correct zip file name is used.

(0) An incorrect zip file name is used.

Bonus: 6 points For each unique and meaningful error in this project description that is reported to the instructor, a student will receive an extra two points.