Trivial Pursuit Java Edition

Technical Report

Catherine Matulis and Jinglan Wang

Wellesley College

December 2, 2013

1 ADTs

1.1 Linked Lists

We will use linked lists to store the questions and answer choices for the game. We have six text files corresponding to different question categories (which correspond to tile colors on the game board in the GUI), and will load the contents of these files into six linked lists. When a player selects a tile of a certain color on the game board, a question from the corresponding linked list will be called.

We chose to use linked lists to store the questions and answers because we are loading questions and answers from text files of unknown size, and also because we want to somewhat randomize the order in which questions are presented, and a queue or stack would make it difficult to present questions in a non-fixed order (in that case, it would be easiest to always take questions from the top or bottom of the stack or queue).

1.2 Graph

We will use a modified version of the AdjMatGraphPlus implementation seen in Exam 3 to keep track of which tiles on the game board in the GUI have been clicked and which tiles a player is allowed to select or move to. Each vertex on the adjacency matrix will correspond to a tile on the game board.

For the version of the game which involves rolling a dice (where the winner is the first to correctly answer a question from each category), the adjacency matrix will be used to determine legal moves. We will add a method to the AdjMatGraphPlus class to determine the distance between vertices, and this will determine where a user is allowed to move based on the result of their die roll.

For the version of the game which does not involve rolling a die (where the winner is the user who answers the most questions correctly), the users must attempt a question for each tile on the game board, so when a tile is selected, the vertex corresponding to the tile will be removed from the AdjMatGraph. When no vertices remain, the game is over.

We chose to use a graph because our gameplay depends on either paths or the number of unselected tiles, which is best represented by a graph.

2 Classes

2.1 TrivialPursuitGUI

This class will create a two-tab GUI using the files corresponding to the "About" and "Play!" panels.

2.2 AboutPanel

This class will create the panel that contains instructions for the game.

2.3 GamePanel

This class will contain the code that allows the user to interact with the GUI to play a game of Trivial Pursuit. This class uses code from the TrivialPursuit class.

2.4 TPQuestions

This class loads the game questions from text files and stores them in six linked lists.

2.5 TPGraph extends AdjMatGraphPlus

This file will keep track of the game board and will allow TrivialPursuit to determine if a move is legal (for the version of the game using dice) or if the game is over (for the version of the game not using dice). This class will be almost identical to AdjMatGraphPlus, but will include one additional method.

3 Actions

3.1 TrivialPursuitGUI, AboutPanel, GamePanel

These classes will include methods to set up the GUI and to allow it to respond to user actions.

3.2 TPQuestions

This class will include a method to read information from a text file into a linked list.

3.3 TrivialPursuit

This class will include methods to call questions corresponding to a specific category, roll a die, determine legal moves, keep track of player names and scores, determine if a game is over, and determine the winner of a game.

3.4 TPGraph extends AdjMatGraphPlus

This class will be almost identical to AdjMatGraphPlus, but will include one additional method, which will determine the number of steps between each vertex. This will allow the TrivialPursuit class to determine if a user's move is legal based on their die roll.