Chance-It

Module Guide

Version 1.0

CMMS Systems

Chris Wong
Matthew Casiro
Melissa Page
Sheryll Tabamo

Table of Contents

Module Breakdown

Module Name	Owner	Tester	Intended Abstraction
App Driver	Chris W	Sheryll T	Hides how the program is initialized
Game	Matthew C	Melissa P	Hides how game moves between states based on user selections and game progress
Input/Output	Sheryll T	Melissa P	Hides all screen output logic as well as user input collection and validation
Local Turn	Melissa P	Chris W	Hides logic for walking through a turn and determining the turn score
Network Turn	Melissa P	Sheryll T	Hides network communications for interacting with a server for online play
Dice	Chris W	Matthew C	Hides dice attributes and dice roll functionality
Random	Matthew C	Sheryll T	Hides how random numbers are generated
Probability	Sheryll T	Matthew C	Hides probability calculations for chance to re-roll the first turn sum
Network Protocol	Chris W	Matthew C	Hides connection protocols to create connection with the server
Computer Player	Matthew C Melissa P	Chris W Sheryll T	Hides decision making algorithms for the computer player
High Score	Sheryll T	Chris W	Hides storage and retrieval method for high score data as well as score comparisons

Module Interfaces

Game:

Pre: randomInit() has been called once
Post: N/A
Clean-Up: N/A
Param: player is a pointer to an unsigned variable
Return: the winning score of the game, or zero if a computer or network player won
unsigned gameInit();

Input/Output:

// Pre: N/A
// Post: The screen was-Updated to display new information
// Clean-Up: N/A
// Param name is the player's name
// Param userSelection is the user's choice
// Returns N/A
void displayGamePlay(char* name, int userSelection);

// Pre: A highscore file exists
// Post: The highscore was displayed on the screen
// Clean-Up: N/A
// Param N/A
// Returns N/A
void displayHighScore();

// Pre: N/A
// Post: The main menu was displayed on the screen
// Clean-Up: N/A
// Param N/A
// Returns N/A
void displayMainMenu();

// Pre: N/A
// Post: The help menu was displayed on the screen
// Clean-Up: N/A
// Param N/A
// Returns N/A
void displayHelpMenu();

// Pre: N/A
// Post: The Network Play Mode was displayed on screen
// Clean-Up: N/A
// Param N/A
// Returns N/A
void displayNetworkPlayMode();

// Pre: N/A
// Post: The prompt for network information was displayed on screen
// Clean-Up: N/A
// Param N/A
// Returns N/A
void displayNetWorkPlayInput();

// Pre: N/A
// Post: The Local Play mode was displayed on screen
// Clean-Up: N/A
// Param N/A
// Returns N/A
void displayLocalPlayMode();

// Pre: N/A
// Post: The menu for in-game help was displayed on screen
// Clean-Up: N/A
// Param N/A
// Returns N/A
void displayInGameHelpMenu();

Local Turn:

// Pre: N/A
// Post: N/A
// Clean-Up: N/A
// Returns: the final turn score
unsigned localTurn();

Network Turn:

// Pre: N/A
// Post: N/A
// Cleanup N/A
// Return an unsigned of the turn score
unsigned networkTurn();

Dice:

// Pre-Conditions: Must be during a turn
// Post-Conditions: A random number was generated
// Clean-Up: N/A
// Returns: The random number
int rollDie();

Random:

// Generate a randomized integer value between the given parameters
// Pre: randomInit has been called once, and min < max
// Post: N/A
// Clean-Up: N/A
// Returns: an integer in the set [min,max]
int getRandomInt(int min, int max);

// Initializes the random module, must be called before any other functions
// Pre: N/A
// Post: N/A
// Clean-Up: N/A
void randomInit();

Probability:

// Pre: N/A
// Post: N/A
// Clean-Up: N/A
// Param sum is the number to check the probability of re-rolling
// Returns: the probability of re-rolling sum
double getProbability(int sum);

Network Protocol:

// Pre: A network game is chosen
// Post: A connection to a server was made
// Clean-Up: Close connection after the game is finished
void connectInit(char* IPaddress, int port);

Computer Player:

// Determines the computer player's decision to roll or stop.
// Pre: N/A
// Post: N/A
// Return 0 for stop, or 1 for roll again
// Cleanup N/A
unsigned getDecision(unsigned roundNumber, unsigned turnNumber,
unsigned turnScore, unsigned p1Score,
unsigned p2Score, unsigned probability);

High Score:

// Pre: a file for highscore exists
// Post: The highscore was displayed on screen
// Clean up: N/A
// Param N/A
// returns N/A
void getHighScore();

// Pre: a file for highscore exists
// Post: The highscore has been amended with new information, if necessary
// Clean up: N/A
// Param info
// returns N/A
void amendHighScore(char* name, char* date, unsigned score);

Uses Hierarchy