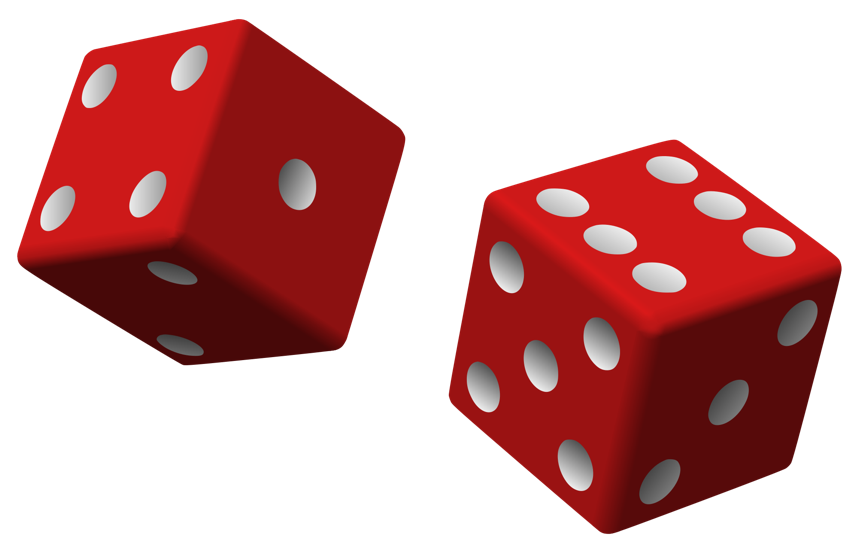
Chance-It

Module Guide

Version 1.0

**CMMS Systems**

Chris Wong

Matthew Casiro

Melissa Page

Sheryll Tabamo

Table of Contents

[Module Breakdown 1](#_Toc451206669)

[Module Interfaces 1](#_Toc451206670)

[Game: 1](#_Toc451206671)

[Input/Output: 1](#_Toc451206672)

[Local Turn: 3](#_Toc451206673)

[Network Turn: 3](#_Toc451206674)

[Dice: 3](#_Toc451206675)

[Random: 3](#_Toc451206676)

[Probability: 4](#_Toc451206677)

[Network Protocol: 4](#_Toc451206678)

[Computer Player: 4](#_Toc451206679)

[High Score: 4](#_Toc451206680)

[Uses Hierarchy 5](#_Toc451206681)

# 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 a pointer to player's name

// Param: firstRoll displays the value of the firstRoll

// Param: round displays the number of round

// Param: roundScore displays the current round score

// Param: die1 displays the first die

// Param: die1 displays the second die

// Param: score displays the player's score

// Param: opponentScore displays the opponent's score

// Returns: 1 for roll, 2 for stop, 3 for probably, 4 for help, 0 for forfeit

**void displayTurn(char\* name, unsigned firstRoll, unsigned round,**

**unsigned roundScore, unsigned die1, unsigned die2,**

**unsigned gameScore, unsigned opponentScore);**

// Pre: N/A

// Post: The rules was displayed on the screen

// Clean-Up: N/A

// Param N/A

// Returns N/A

**void displayRules();**

// 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 and the user selected a game type

// Clean-Up: N/A

// Param N/A

// Returns: 1 for local play, 2 for network play

**unsigned displayMainMenu();**

// Pre: N/A

// Post: The Network Selection Mode was displayed on screen

// Clean-Up: N/A

// Param: N/A

// Returns: 1 for human player, 2 for computer player, 0 to previous menu

**unsigned displayNetworkSelectMode();**

// Pre: N/A

// Post: The prompt for network information was displayed on screen, and IPaddress and

// port variables were updated

// Clean-Up: N/A

// Param: IPaddress pointer to the variable holding the ipaddress

// Param: port pointer to the variable holding the port

// Returns: N/A

**void displayNetWorkPlayInput(char\* IPaddress, unsigned\* port);**

// Pre: N/A

// Post: The Local Play mode was displayed on screen

// Clean-Up: N/A

// Param: N/A

// Returns: 1 for single player, 2 for multiplayer, 0 to return to previous menu

**unsigned displayLocalSelectOpponent();**

// Pre: N/A

// Post: The player names were assigned to the given char\*

// Clean-Up: N/A

// Param: player1 is a pointer to player 1's name

// Param: player2 is a pointer to player 2's name

// Returns: N/A

**void displayLocalPlayGetName(char\* player1, char\* player2);**

// 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: name a pointer to the player's name

# // Param: date a pointer to the date of the game play

# // Param: score takes in the value of the player’s score

# // Returns: N/A

# void amendHighScore(char\* name, char\* date, unsigned score);

# Uses Hierarchy

