Thread Games

ISTE-121 – Alternate Homework 04

Problem Statement:

The purpose of this assignment is to create a simple card game that requires loose synchronization (not actual synchronization) among three threads. Since you have not learned about Java synchronization yet, this will be done manually through global variables, similar to Lab 4's use of the *keepGoing* Boolean variable. The card game will be implemented using text only. No GUI is required.

Here is how the game is played: The game consists of 10 rounds. During each round, Player 1 and Player 2 randomly draw a card. A Referee determines which player's card has the greatest value (see table below) and assigns a winner and loser for that round. The player with the card with the highest value wins that round, while the player with the lowest valued card loses that round. If each player draws a card with the same value, the round is considered a tie. At the end of 10 rounds, the player with the most wins is considered the winner. If the players have the same number of wins, the entire game is considered a tie.

Card	Value
Joker	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
Jack	11
Queen	12
King	13
Ace	14

Requirements / Specifications:

Player 1, Player 2, and the Referee will be implemented as their own respective threads, using inner classes, that are in a class called CardGame.

In CardGame, there will be private global members to represent each player's card, playerOneCard and playerTwoCard.

At the start of each round, the Referee will set each player's card to 0 and declare that the players are free to randomly draw a card (using random numbers between 1 and 14, as shown in the table above). The players may not draw a card for that round unless the Referee says they can. If a Player thread runs before the Referee says they can draw a card, the Player must go to sleep for 100 milliseconds.

The Referee is not allowed to declare the winner of a round until both players have drawn a card. If the Referee thread runs before both players have drawn a card, the Referee must go to sleep for 100 milliseconds.

Once both players draw a card, the Referee will determine the winner of the round and print out a message with the round number that states the cards each player drew and who won the round (or if it was a tie).

Examples:
Round 1 – Player 1 drew King, Player 2 drew 6. Player 1 wins!
Round 2 – Player 1 drew Joker, Player 2 drew Ace. Player 2 wins!
Round 3 – Player 1 drew 9, Player 2 drew 9. Tie!

A global variable should be used to keep track of which round it is.

The Referee has a Scoreboard object (implement this as an inner class or create a class in a separate file called Scoreboard.java) with private members (and related accessors and mutators) as follows:

- playerOneWins
- playerOneLosses
- playerOneTies
- playerTwoWins
- playerTwoLosses
- playerTwoTies

At the end of the game, the Referee will declare a winner (either Player 1 or Player 2) or say the game was a tie and print out the win, loss, and ties statistics for each player.

Example:
Player 1 wins the game!
Player 1: 5 W, 4 L, 1 T
Player 2: 4 W, 5 L, 1 T

121 Homework 4 – Thread Games Name:

Item	Possible points	Earned points
Thread Behavior:		
Players: (30%) Players wait until the referee has said they can draw a card Players draw cards as specified	 20 10	
Referee: (50%) Referee arbitrates each round as specified Referee properly determines the winner of each round and saves statistics Referee properly determines winner of the game and outputs statistics	 30 15 5	
Coding: (20%) Class members are private Threads are implemented as inner classes Scoreboard class implemented as specified	 5 5 10	
Deductions: Late deduction Coding standard violations, etc. Submitted something other than a zip file, or not all code supplied (-5) Other violations:		
Total:	100	

Comments: