Liar’s Dice Competition Instructions

Liar's Dice is a group dice game heavy on the bluff. The rules are simple, but the strategy can be quite complex. In this tournament, the players are Bots (Java classes written by programmers), and they will play against each other with no human intervention. May the best Bot win!

Competition Time Frame:

The competition is currently open, and submissions will be accepted through the end of Wednesday, April 10th. We will announce the tournament winners at the ACM closing social on Friday, April 12th.

Rules of the Liar’s Dice Bot Tournament:

Each human competitor may enter one bot in the tournament. (Resubmissions are allowed until the deadline, but they will overwrite previous submissions.) Each bot will play a prespecified number of games, with four bots in each game. The winner of each game will get a point, and the bot with the most points wins the tournament.

Each bot gets five randomized dice to start a game. The first bot begins bidding, picking a face (2,3,4,5,or 6) and a quantity (1 to 20). The quantity is the claim of how many of the chosen face have been rolled in total on the table. All 1’s are wild and count as the face of the current bid, no matter who has them.

Each bot has two choices during his turn: make a higher bid, or challenge the previous bid. Raising the bid means either making a new bid with a higher quantity, or keeping the same quantity but raising the face value. (2 5’s < 2 6’s < 3 2’s < 3 3’s)

When a bot challenges the current bid, the round ends and the bid is compared against all the dice. If the bid is valid, the challenger loses a die. Otherwise, the bidder loses a die. The dice are again randomized (re-rolled) and the player who lost a die in the last round starts the next round.

When a bot loses its last die, it leaves play and the next round starts with the next bot in line. The last bot with dice wins the game and gains a point in the overall tournament.

Requirements for a bot file:

A bot is defined in a class that inherits from LiarsDiceBot. (See “LiarsDiceBot.java” for details.)

All code must be in one .java file. Use internal classes if you need more than one class.

To be recognized, the bot class must have a line that starts with “public class botNameHere extends LiarsDiceBot ”

Only two methods need to be implemented: getDecision() and getName(). The reportGameResults() method is optional.

No reflection allowed. No file access allowed. We can’t have bots accessing other bots’ files or files local to the tournament computer.

No multithreading allowed, since it could hog resources or be impossible to interrupt when a bot is out of time.

Bots that don't follow these requirements will be disqualified.

Bot development tips:

You don't need to read through *all* the code that runs the tournament. Essential classes to understand are: LiarsDiceBot, Bid, Challenge, and GameInfo. The GameHistory, Round, Turn, and Result classes form a hierarchy that may become useful if you need more information about what happened more than one turn ago.

Bot objects will persist from game to game in the tournament. The best bots could learn and adapt to the strategies their competitor bots use. reportGameResults() is called on your bot at the end of each game it plays in so that it can review the choices and their results in that game.

Bots will lose a round (and therefore a die) automatically in three circumstances: Timeouts, Exceptions, and Invalid Decisions. A good bot need never make these mistakes. It is recommended that you test your bot against the sample bots in the Tournament View to see how often your bot produces these problems. Each problem is explained below.

Timeouts: Bots are given limited time (50 ms) to make each of their decisions. A slow bot will be interrupted and lose a die whenever it takes longer than the allotted time to return a decision from getDecision(). reportGameResults() will also be interrupted if it takes too long, but there won't be other consequences.

Exceptions: Bots should handle all their own exceptions, but if one slips out of their control, it will be caught by the tournament and the bot will be penalized by losing a die.

Invalid Decisions: Bots need to follow the game rules when they choose their decision. A bid that is too low (or impossibly high), or a challenge on the first turn of a round will automatically lose the round. For your convenience, the checkValidDecision() method on the LiarsDiceBot class is available to ensure that a decision is valid. It is strongly recommended that each bot use this to test their chosen decision to avoid unnecessary loss.