Programming task: card game

Read the enclosed paper *Functional Software Architecture*, which describes a design and implementation strategy for the card game *Hearts*. We are working on a modification of the code base that comes with this article that is supplied as a file `task-template.zip`. Unpack this file into a fresh directory. This code base contains preliminary modifications to implement the card game *Oh Hell*, but there are some problems for you to fix.

```
$ mkdir fpexam
$ cd fpexam
$ unzip /path/to/task-template.zip
```

Your task consists of

- documenting your work in a file `LOG.md` that you create in the `fpexam` directory; this file should explanaing what you implemented for each of the following subtasks (which modules did you change, which functions, data structures, type classes, etc are affected).
- implementing the subtasks listed below. Make sure you do not introduce new errors by appropriate testing.

You final submission consists of a single zip file `final.zip` obtained by

```
$ # in directory fpexam
$ zip -r final LOG.md ohhell.cabal src stack.yaml stack.yaml.lock
```

The code in the zip file must build and run using stack or cabal as described in the original `README`. Do not submit binaries as they will be ignored. Do not change the underlying architecture as it is explained in the paper. Make sure that the submitted program compiles as a whole, even if you do not finish all subtasks. The rules of the game are attached at the end of this document.

1. (10 points: show of cards) The current implementation asks for a bid before showing the cards of the interactive player. Fix this problem.

2. (20 points: dealer rotation) For each round, the dealer moves to the left of the dealer of the previous round. Fix this problem by adding a suitable component to the `GameState`.

3. (10 points: trick taking logic) There is an error in the trick taking logic. Here is an excerpt from the game that demonstrates the problem:

   ```
   Trump Card is : 10♡
   ...
   Your turn, Robo-3!
   Robo-3 plays 5♣
   Your turn, PETER!
   Cards on table:
     Robo-3: 5♣
   Your hand:
     9♡
   You have no choice.
   JIMBO plays 9♡
   Your turn, Robo-1!
   Robo-1 plays A♣
   Your turn, Robo-2!
   Robo-2 plays 2♡
   Robo-2 takes the trick:
   2♡, A♣, 9♡, 5♣
   ```

The problem is that the trick should be Jimbo's: ♡ is trump, Jimbo played 9♡, which is higher than Robo-2's 2♡.

Identify the source of the problem and fix it.

4. (30 points: interactive dealer) If the interactive player is also the dealer, then interactive bidding starts before the cards are dealt and before trump is announced.

   Fix this problem.

5. (20 points: selective rounds) Playing the game to the end can be time consuming. Extend the argument parser and the program logic to take a specification for the number of rounds and the cards to play in each round as a parameter. The program argument should have the form

   ```
   $ cabal run ohhell -- --rounds=[1,2,3] # three rounds with 1-3 cards
   $ cabal run ohhell -- --rounds=[10,10,10] # three rounds with 10 cards
   $ cabal run ohhell -- --rounds=[1,2,3,4,5,6,7,8,9,10,11,12,13] # play original game
   ```

   where single rounds and ranges can be mixed in arbitrary order.

   The command line parser is based on the library `optparse-applicative`. Consider using `option auto`. This subtask is independent of the previous ones.

6. (10 points: announcement) If you completed the previous task, the count of rounds no longer matches the number of cards dealt in this round. Besides the round number, announce the number of cards dealt at the beginning of each round. (only counts in connection with the completion of the selective rounds task)

7. (20 points: global bookkeeping) Currently, there is no bookkeeping across rounds of the game. Implement the following functionality by extending the `GameState` with appropriate maps:

   - at the end of each round display the stats: for each player, the bid, the number of tricks won, and the number of points earned in this round
   - at the end of the game, print the final stats and announce the winner.

8. (30 points: clever robo) The automatic player implemented in `Gameplay.strategyPlayer` plays according to the rules, but without any ambition to win. Define a module `M0000000` (replace) that implements an improved player that performs significantly better than the predefined `strategyPlayer`.

   - Place the bid `n` according to the estimated probability that the player takes `n` tricks in this round.
   - Take all available information into account: the cards in hand, whether you lead the first trick, etc.
   - When choosing a card, take into account your current bid, how many tricks you already scored, and what cards may likely be still in the game.

   We will evaluate your strategy using the framework implemented in the `Tournament` module. You don't have to do that yourself (you are free to try). Marks are given on the basis of the implemented strategy and your explanation of it, not on the performance in the tournament.

Oh Hell (Blackout) Simplified Game Rules

By ERIK ARNESON Updated on 10/20/19 https://www.thesprucecrafts.com/oh-hell-card-game-rules-411138

Oh Hell is a simple trick-taking card game, such as regular or two-player Spades, that presents as much strategic play as any popular board game. The game Wizard, published by US Games, is based on Oh Hell. This game is sometimes referred to as Oh Pshaw or Blackout.

Players

Four players.

Deck

Standard 52-card deck. Ace is high; two is low.

Goal

To score the most points by accurately predicting how many tricks you'll win.

Setup

Shuffle the cards. Choose a dealer. For each subsequent hand, the player to the left of the previous dealer becomes the new dealer.

The four player game lasts for 13 hands. For the first hand of the game, each player receives one card. For the second hand, each player receives two cards. The third hand, three cards, and so on until the end of the game.

After the cards are dealt, the dealer turns the next card face up. The suit of this card establishes the trump suit (in the last hand of the game, there is no trump suit, so this card is not turned face up).

Remaining cards are set aside and not used in that hand.

Bidding

The player to the dealer's left bids first. Each player must bid; no one may pass. Legal bids range from 0 to the number of cards dealt for that round. For example, if four cards are dealt, legal bids range from 0 to 4.

Players are bidding on the number of tricks they think they'll win in that hand.

Gameplay

The player to the dealer's left plays first, or "leads." Play continues clockwise. Each player must follow suit (i.e. play the same suit that was led), if possible.

Generally, each trick is won by the player who played the highest rank in the suit led. However, if the suit led was not trump, and one or more players played a trump card, then the trick is won by the player who played the highest rank of trump.

When a trick is won, the winning player sets the trick in front of himself so that it's easy to tell how many tricks each player has won.

Scoring

One player serves as the scorekeeper. As each player makes his bid, the scorekeeper writes them down. All information about the bids is open, and any player can ask for a reminder of who bid what at any time during the game.

Players only score points by precisely predicting the number of hands they would win. A bid that's either too high or too low scores zero points.

Each player who makes his bid exactly scores 10 points plus the number of tricks won. For example, Evelyn bid four and won four tricks. She scores 14 points (10+4). Frank bid zero and won zero tricks. He scores 10 points (10+0).

Winning

The player with the highest total score at the end of the game is the winner.