Introduction

In this assignment you will write a game data analyzer to spot potential faults in a game's logics. The game you will be using is a custom version of blackjack (rules defined below).

Our blackjack rules

1. This version of blackjack is played only with 2 people – the player and the dealer.
2. There is no money involved, people play for fun and glory.
3. The game is played with one deck of cards (no jokers).
4. The goal is to have a hand total of 21 or as close to 21 as possible without going over (going bust).
5. The player is dealt two cards face up, while the dealer gets one card face up and one card face down.
6. Cards 2-10 are worth their face value, while face cards (Jack, Queen, King) are worth 10 points, and Ace is worth 11 points.
7. After receiving their initial two cards, the player can choose to "hit" and receive additional cards to improve their hand, or "stand" and keep their current hand.
8. If the player's hand exceeds 21, they bust and lose the game automatically.
9. Once the player decides to stand, the dealer reveals their face-down card and hits until their hand totals 17 or more.
10. If the dealer busts, the player wins. Otherwise, the dealer compares their hand with the players hand, and the one with a higher hand total wins.
11. If the players hand total equals the dealer's hand total, the player wins.

Assignment rules

1. The program must read in the game data from a .txt file with the name game_data.txt (data format explained later) from resources folder in your project.
2. The program must write the output to analyzer_results.txt (data format explained later) to project root.
3. The output must be sorted by game session ID.
4. The program must be written in Java versions between(including) 8 and 19.
5. No third party libraries are allowed, only base Java.

Input

The game data is presented in a text file with each line describing one turn of a game session. The line will look like this:

```
1678346026,123456789,987654321,P Hit,2S-?,AS-8C
```

The breakdown is as follows:

1. Timestamp: 1678346026
   a. Timestamp of the turn in UNIX format
2. Game Session ID: 123456789
   a. Unique identifier for the session

3. Player ID: 987654321
    a. Unique identifier for the player
4. Action: Player Hit (P Hit)
    a. The action that will happen this turn
5. Dealer's Hand: Two of spades and a face down card (2S-?)
    a. Dealers hand before the action is put into effect. **EXCEPTION!** When player joins a table, his cards are already dealt. Same when game resets and new cards are dealt
6. Player's Hand: Ace of Spades and 8 of Clubs (AS-8C)
    a. Player's hand before the action is put into effect. **EXCEPTION!** When player joins a table, his cards are already dealt. Same when game resets and new cards are dealt

The input may be a full game session, many different game sessions, or half a game session. One game session is at least one line long.

Output

Output is a text file with a list of the first faulty move of a game session. This means that for the game session to be faulty, you only need to find one flaw in the session. We assume this because if the game logic is broken at one point, the logic might not be correct moving on.

The output result format will be the same as the input game data format. A line of game data, each faulty move on a new line. Example:
1678346026,123456789,987654321,P Hit,?-AH,AS-8C-KH-KC

In this example the player had an ability to hit even though he had bust, and dealer should have won.

Things we can assume

There are some things we can assume that are correct for simplicity's sake.

1. The game session will always start with a player joining a table.
2. Once the cards have been dealt, they will not change in the future.

General notes

Example inputs and outputs have been included with this assignment.

Some more helpful information:

1. The logs are not sequential
2. Player can have multiple wins and losses within one game session
3. Java objects are your friends
4. ? in hand represents hidden cards
5. The card names are in short format:
    a. S – spade
    b. H – heart
    c. C – clubs
    d. D – diamond

6. Your analyzer logic will only need to check information that is directly related to blackjack itself, not the game session creation
7. For the game session to be faulty, you only need to find one flaw in the session

Submitting

For submitting you will need to upload you project to a website where we could download it for reviewing. After uploading you will need to provide the link to the project via Smartrecruiters.