SWEN30006 Software Modelling and Design
Project 2: Pasur Trainer

School of Computing and Information Systems
University of Melbourne
Semester 2, 2021

1. Overview

You and your team of independent Software Contractors have been hired by *New, Exciting and Revolutionary Designer Games* (aka *NERD Games*) to provide some much-needed assistance in developing their innovative *Pasur* card game trainer (see details below) to be market ready.

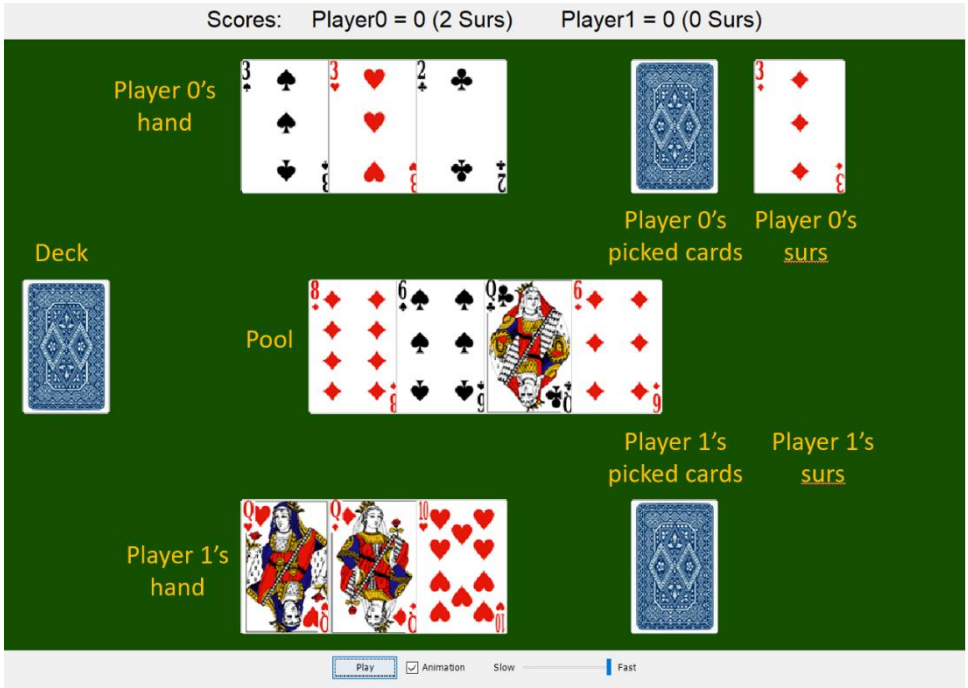

Figure 1: Pasur Snapshot

NERD Games have a configurable, running version of the trainer, but it does not provide any scoring, or log the run sessions.

Your task is simple: add scoring and logging to the trainer, with a flexible design which will allow for future maintenance and modification by the NERD team. You are free to modify the existing design and implementation to facilitate these changes, with some limitations (see below). You will also need to provide a report to the NERD team explaining and justifying your changes to the design (details below).

2. Playing the Pasur Card Game

2.1 Introduction

- Pasur is played with a standard fifty-two card deck with four *suits* and thirteen *ranks* in each suit.
  - The suits are hearts, diamonds, clubs, and spades.
- The game involves 2, 3 or 4 *players*.

    - Players take turns being the dealer.
    - In the provided implementation, there are 2 players.
- The objective of the game is to win the most points
- "Numbered" cards are Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10. Ace is assigned the value of 1. "Face" cards are King, Queen, and Jack.
- Players will have cards which are placed in front of them.
- There is a *pool* in the middle of the table made of cards.

In summary, each game includes the following sequence of activities: A *round* consists of steps 1-8.

1. Dealing 4 cards to each player
2. Play a card
3. Picking up cards from pool (may happen)
4. Scoring a Sur (may happen)
5. Players alternate and do steps 3-5 till no card is left in their hands
6. Repeat steps 1-6 till no card is left in the deck
7. End of this round of the game
8. Tallying the points scored by each player
9. Resetting the deck to 52 cards and starting a new round of the game from step 1 till a player scores 62 or more points, who will be the winner.
    a. See section 2.7 for more details, including what happens for a draw.

2.2 Dealing for a round

- Each player is dealt four cards, which are placed face-down in front of them (named *hand*).
- In the first round, four cards are placed face-up in the middle of the table. These four cards form the *pool*.
    - If any <u>one</u> card in the pool is Jack rank, this card is placed randomly back into the middle of deck and is replaced by a new card from the top of the deck.
        - If the replacement card is also a Jack rank the dealer must shuffle the deck and deal again.
    - If any <u>two or more</u> cards in the pool are Jack rank the dealer must shuffle the deck and deal again.

2.3 Play

Players take turns to *play* one card to the pool at a time. This continues until each player has no cards remaining in their hands. This is known as a *round of play*.

A *play* involves playing one card in one of two possible ways:

1. Adding the card to the pool of face-up cards in the middle of the table.
   <u>or</u>
2. Using the card to pick up one or more cards from the pool in the middle of the table.
    - A player cannot add a card to the pool if that card can pick up one or more cards from the pool.

Once each player has no cards remaining in their hands, the dealer will deal four additional cards to each player (but not the pool). This repeats until the deck is empty. When this happens a *round of the game* is completed.

2.4 Picking up cards

During a play, a player can pick up one or more cards from the pool in the middle of the table.

The cards which can be picked up from the pool depend on the value of the card which is being played.

- If a King is played, one King is picked up from the pool.
    - You can play a King if there is no King in the pool. But, clearly you won't pick up anything in this case.
- If a Queen is played, one Queen is picked up from the pool.
    - You can play a Queen if there is no Queen in the pool. But, clearly you won't pick up anything in this case.
- If a Jack is played, all Jacks and number cards are picked up from the pool.
    - King and Queen cards remain in the pool.
- If a Number card is played, you can pick up one or more cards from the pool, if the combined value of the Number card and the cards from the pool sums to 11.
    - e.g., The pool contains Ace, 4, 4, 9.
      A player would need to play a 10 to pick up the Ace, as 1+10=11.
      A player would need to play a 7 to pick up either of the 4's, as 4+7=11.
      A player would need to play a 2 to pick up the 9, as 2+9=11.
      A player would need to play a 3 to pick up both of the 4's, as 3+4+4=11.

On the last hand of the deck, whoever has picked up the last card gets to pick up all remaining unmatched cards from the table at the end of the round.

When a player picks up cards from the pool, these are placed in a separate pile face-down in front of them. This is explained in more detail below in section 2.6 Scoring.

2.5 Surs

A *Sur* is a specific event which occurs during the game when a player plays a card that picks up all the cards from the pool. There are two exceptions.

- If a Jack card is played and used to clear the pool, a Sur does not occur/is not scored.
- A Sur cannot occur/is not scored during the last round of play.

Players keep track of the number of Surs which they score during the game by placing a card involved in scoring the sur face-up in a separate pile and will be awarded points for it during the game (see below). If other players have previously scored a Sur, the player uses his/her scored Sur to offset one of the Surs of other players. In the latter case, the player will not receive any points for his/her scored sur, and hence, no card will be added to his/her sur pile.

2.6 Scoring each round of the game

The purpose of the game is to collect as many points as possible.

- Each player calculates their *round score* once the deck is empty (i.e., at the end of each round of the game).

Each player looks at the cards they picked up (which were placed in a separate pile face-down) as well their sur cards (which were placed in a separate pile face-up) in front of them. Points are awarded as follows:

- Player who has 7 or more clubs: 7 points (there are only 13 clubs so they must have the most if they have 7 or more)
- Player who has the 10 of diamonds: 3 points
- Player who has the 2 of clubs: 2 points
- Each Ace: 1 point
- Each Jack: 1 point
- Each *Sur*: 5 points.

This means that there are 20 total points which are available each round that are distributed between players (in addition to several 5-point bonuses for each *Sur* that occurs).
i.e., 20 points available each round consists of:

- Most clubs (i.e., at least seven clubs): 7 points
- 10 of diamonds: 3 points
- 2 of clubs: 2 points
- 4 Aces: 4 points
- 4 Jacks: 4 points

Note that if a point rewarding card such as a club card, 10 of Diamonds, ... is added to the sur pile, it should still be counted towards the regular 20 points. In addition, it also awards the player 5 points (as a sur).

2.7 Ending the game (scoring the total)

At the end of each round of the game, players calculate their round score and add it to their *total running score*, which is the sum of all their *round scores*.

E.g. For a player the round scores may look like this after three rounds of games:

Round of game	**Round Score**	**Total Running Score**
1	9	9
2	13	22
3	10	32

When a player's *total running score* is 62 or more the game ends, and the player has won the game.

- If two or more players obtain a *total running score* of 62 or more points after the same round, and their *total running scores* are the same, the game is currently tied. In this case, the game continues with another round (steps 1-8) until a player has a higher score.

E.g., For a winning player the round scores may look like this:

Round	**Round Score**	**Total Running Score**	**>= 62?**	**Has player won?**
1	9	9	No	No
2	13	22	No	No
3	10	32	No	No
4	8	40	No	No
5	8	48	No	No
6	22 (had a Sur)	70	Yes	**Yes**
Stop.		Game Ended.		

3. Required Changes and Reporting

3.1 Scoring

The Pasur trainer does not provide any scoring. You need to add scoring as per the rules above.

Note that there are some variations to the rules above, for example, some give the player with the most clubs 13 points instead of 7. While you do not need to implement any variations, your design should account for the situation, in the future, where the system may be extended to be configurable to handle scoring variations. In a real Pasur game, the scores are calculated at the end of each round of game. *However, to make the game more exciting, in our version of the game, the round score of each player must be calculated and the total running score of each player must be updated and visualized during the game every time a player plays a card.* The scoring of the current ongoing round of game should include all the scoring rules mentioned in Section 2.6 including surs. Therefore, it may happen that the score of a player increases and then decreases if he/she scores a sur and later another player offsets his/her sur as explained in Section 2.5.

3.2 Logging

To help players study the impact of their choices on the play and score[1], the activity of the Pasur trainer needs to be logged to a file called "*pasur.log*" stored in the project directory (next to the src folder).

Cards are to be referred to by two letters separated by '-', the first for rank (one of: A, K,Q,J,10,9,8,7,6,5,4,3,2) and the second for suit (one of: C,D,H,S). A hand is shown by listing comma separated cards between square brackets in descending order of rank and alpha order of suit within rank.

Events are logged, one per line in the order they occur without whitespace as per the example below of the expected log file contents for 2 players of type "pasur.RandomPlayer" with seed "30006" and animation set to true:

```
#Seed: 30006
#Animate: true
#Player0: pasur.RandomPlayer
#Player1: pasur.RandomPlayer
Game starts...
Round 1 of the game starts...
Total Running Scores: Player0 = 0 (0 Surs)        Player1 = 0 (0 Surs)
Dealing out to players...
Player0 hand: [J-S, 10-S, 4-D, 3-H]
Player1 hand: [A-H, A-S, 4-S, 3-S]
Dealing out to pool...
Pool: [9-S, 8-C, 7-D, 6-S]
Player0 plays J-S
Player0 picks [J-S, 9-S, 8-C, 7-D, 6-S]
Total Running Scores: Player0 = 0 (0 Surs)        Player1 = 0 (0 Surs)
Player1 plays A-S
```

[1] And to help us assess the correctness of your implementation by automatically processing and comparing the output your code produces.

```
Player1 picks []
Total Running Scores: Player0 = 0 (0 Surs)        Player1 = 0 (0 Surs)
Player0 plays 10-S
Player0 picks [A-S, 10-S]
Player0 scores a sur
Total Running Scores: Player0 = 0 (1 Surs)        Player1 = 0 (0 Surs)
Player1 plays 3-S
Player1 picks []
Total Running Scores: Player0 = 0 (1 Surs)        Player1 = 0 (0 Surs)
Player0 plays 4-D
Player0 picks []
Total Running Scores: Player0 = 0 (1 Surs)        Player1 = 0 (0 Surs)
Player1 plays A-H
Player1 picks []
Total Running Scores: Player0 = 0 (1 Surs)        Player1 = 0 (0 Surs)
Player0 plays 3-H
Player0 picks [A-H, 4-D, 3-H, 3-S]
Player0 scores a sur
Total Running Scores: Player0 = 0 (2 Surs)        Player1 = 0 (0 Surs)
...
```

The current version of the game logs everything on the terminal. This needs to be replaced by logging everything in the *"pasur.log"* file. Hence, *in the new version of the game nothing must be printed on the terminal*. Note that you should not add (resp. Remove) anything to (resp. from) the current log and must preserve the records of the current log and their order.

4. Report

You are required to produce a **report** describing the changes you have made to the system and providing a rationale for these changes **using design patterns and principles**. Note that to do this well, you should include discussion of other reasonable options which you considered, and why you did not choose these options. You should clearly identify the points of evolution/variation supported by your design with appropriate justification.

Note that you should draw a class diagram for the whole project (not only the changes, which was the case of project 1), and also draw a design sequence diagram for computing the score of a player.

Around 4 pages should be enough to provide a concise rationale. You should include and refer to diagrams in your report to help illustrate aspects of the changes.

5. The Base Package

You have been provided with an Eclipse project export representing the current system.

To begin:

1. Import the project zip file as you did for Workshop 1.
2. Add the JGameGrid.jar file (in dist/lib) to the Java Build Path for the project: Project > Properties > Java Build Path > Add Jars …. You may need to remove it from the build path and re-add it if the path is not correct.

3. Try running by right clicking on the project and selecting **Run as... Java Application**.
4. You should see a window like the one in Figure 1; you can then play the game by pressing the play button.

The current system should be used as a starting point. Please carefully study it and ensure that you are confident you understand how it is set up and functions before continuing. You will need to preserve the game play behaviour, including the use of randomisation. However, you are free to make whatever changes to the code you wish, given the game play behaviour is preserved and you can justify your changes in your report. If you have any questions, please make use of the discussion board; it is assumed that you will be comfortable with the package provided.

Note: You may modify the *seed* of the random number generator to see a different run of the game and the *automate* flag to toggle the automation on/off in the pasur.properties file. Also, note that, currently, the only implemented player is a RandomPlayer that every time plays a random card. In the future, other types of players such as a SmartPlayer or a HumanPlayer may be added to the game.

6. Marking and Submission

Note: There are three required tasks here – design, code, and report – which are interrelated. You should work on these tasks **as a team**, not separately. Every team member needs to be able to demonstrate their understanding of and contributions to all deliverables.

Testing Your Solution

6.1 Testing Your Solution

We need to be able to build and run your program without using an integrated development environment. We will be running your application via script with appropriate property files and applying automated checks to the output. The entry point must remain as "Pasur.main()". Other than JGameGrid, you should use only standard Java libraries (using other libraries will result in a failure of your system to build).

Note: It is **your responsibility** to ensure that you have thoroughly tested your software before you submit it.

6.2 Marking Criteria

This project will account for 20 marks out of the total 100 available for this subject. The rubric for the project is available on the LMS.

Note: We reserve the right to award or deduct marks for clever or very poor code quality on a case-by-case basis outside of the prescribed marking scheme. Further, we expect to see good variable names, well commented functions, inline comments for complicated code. We also expect good object-oriented design principles and functional decomposition. For UML diagrams you are expected to use UML 2+ notation.

On Plagiarism: We take plagiarism very seriously in this subject. You are not permitted to submit the work of others under your own name. This is a **team** project. More information can be found here: https://academichonesty.unimelb.edu.au/advice.html.

6.3 Submission

Detailed submission instructions will be posted on the LMS. You should submit all items shown in the checklist below. You must include your team number in all your pdf submissions, and as a comment in all changed or new source code files provided as part of your submission.

Submission Checklist

1. Your complete updated source code.
2. Design Analysis Report (pdf) including diagrams. Diagrams need to be placed inside the pdf document.

Note: Only one member of the team to submit.

Submission Date

This project is due at **11:59pm (Melbourne time) on Friday October 22nd 2021**. Any late submissions will incur a 1-mark penalty per day unless you have supporting documents. If you have any issues with submission, please email Alireza at aostovar@unimelb.edu.au **before** the submission deadline.