SWEN30006 Software Modelling and Design
Project 2: Whist
- Project Specification -

School of Computing and Information Systems
University of Melbourne
Semester 2, 2020

Background: Whist

New, Exciting and Revolutionary Designer Games Inc. (aka *NERD Games Inc.*) has developed and released the GUI *Whist* card game (see Figure 1) to the market. Whist is played with a standard fifty-two card deck with four *suits* (i.e., Hearts ♥, Spades ♠, Diamonds ♦, and Clubs ♣) and thirteen *ranks* (i.e., 2-10, Jack, Queen, King, and Ace) in each suit. The game involves 4 *players* who play independently (there are no teams).

The Whist game play is briefly described below.

1. The game starts with a hand of thirteen cards being *dealt* to each player. Then, a ***trump*** suit being randomly selected (displayed in the upper left of Figure 1), and a player being randomly selected to start or ***lead***.
2. In each round, the game play proceeds as follows:
   i. The player taking the *lead* can play any card they wish from their hand to the centre; this card provides the basis for a *trick*.
   ii. Play proceeds clockwise from the lead with each player playing one card in turn to *follow* the lead.
   iii. Following players must play a card of the same suit as that lead or a card of the trump suit if they have one. If not, they may play any card they wish.
   iv. Once every player has played one card to the centre, the winner of the round is the player who has played the highest card of the lead suit (i.e., the suit of the card that the first played). If there are other players who played a card in the *trump* suit, the winner will be the player who played the highest card in the trump suit instead.
   v. The winner receives one point for winning the trick
   vi. The winner takes the *lead* for the next round starting a new trick.
   vii. The game play ends as soon as a player has received 24 points and that play wins the game. If the players have played all their cards without anyone winning, play continues exactly as described from 1 starting with new deal.

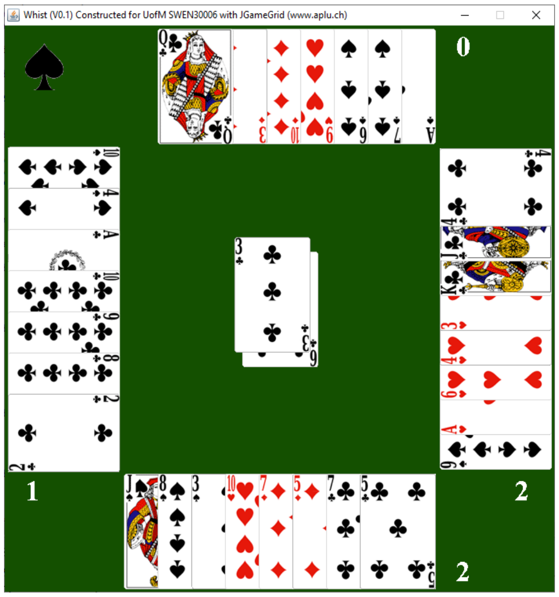

Whist (V0.1) Constructed for UofM SWEN30006 with JGameGrid (www.aplu.ch)
0
1
2
2

Figure 1: Whist GUI

NERD Games Inc. has a fully running version of the GUI Whist game. The Whist program currently supports two types of players:

1. **An interactive player:** a player is the user who plays the game through GUI, i.e., selecting a card by a double-left-mouse-click.
2. **A Non-Playable Cardplayer (NPC):** an automated player who plays the game by randomly selecting a card from the hand without regard for the rules.

The game works reasonably well but not well documented. Moreover, the current version is limited in terms of configurability and the available types of NPCs.

Your Task:

You and your team of independent Software Contractors have been hired by *NERD Games Inc.* to provide some much-needed assistance in improving their innovative *Whist* card game. Your task is simple. You have to modify the design and implementation to improve configurability, to add a new type of NPC, and to facilitate the addition of other future changes. The details of the enhancements are as follow:

- **Advanced NPC**: As described in the background, the current version has only a random NPC who plays the game by randomly selecting a card without regard for the rules. NERD Games Inc. wants a more intelligent NPC to make the game more challenging. To select a card to play, NERD Games Inc. designed that the advanced NPC will do 2 steps, i.e., *cards filtering* and *card selecting*. Figure 2 below provides an illustrative approach of the advanced NPC. The card filtering step is *optional* and the card selecting step is *mandatory*, meaning that the NPC can be configured to skip the card filtering step, but the NPC must always use one of the card selecting approaches.

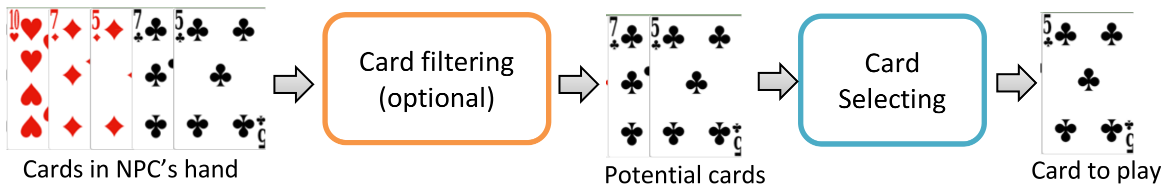


Figure 2: an illustrative example of how the advanced NPC select a card to play.

  - **Cards filtering approaches:** The advanced NPC will first select a set of cards in the hand that can be played. The NPC will use one of the following approaches to select a set of potential cards:
    - *Naïve legal approach:* The advanced NPC will attempt to select the cards that consistent with the rules, i.e., the cards in the lead suit and the cards in the trump suit. If the NPC does not have cards in the lead suit nor in the trump suit, it will select all cards in the hand (i.e., no filtering). If the NPC takes the lead, then no filtering.
    - *Trump saving approach*: This approach is similar to the naive legal approach, but the NPC will try to save the card in the trump suit. To do so, the NPC will attempt to select the cards in the lead suit. If the NPC does not have the cards in the lead suit, then the NPC will select the cards in the trump suit. Same as the naïve legal approach, if the NPC does not have cards in the lead suit nor the trump suit, it will select all cards in the hand (i.e., no filtering). If the NPC takes the lead, then no filtering.
  - **Card selecting approaches:** The advanced NPC will select the card to play using one of the following approaches.
    - *Random selection:* The NPC will randomly select the card to play.
    - *Highest rank selection:* The NPC will select the card with the highest rank to play. If many cards are applicable for a selection, NERD Games Inc. is open to your idea on how to select the card.
    - *Smart selection*: The advanced NPC will record the relevant information (e.g., collecting the cards that been already played) and makes a reasonable selection. For example, the NPC may consider the chance whether it will win or not. If not, it will violate the rules by selecting card with the lowest rank. For this approach, NERD Games Inc. is open to your ideas for the smart selection approach. However, NERD Games Inc. does not expect a sophisticated algorithm. At least, this approach should make a decision based on the collected information and it should look smarter than the combination of pre-defined approaches above.
- **Configurability**: Currently all elements of Whist are set in the code. The system should be made more configurable through a property file. *You will have to make a judgement as to which parameters are worth making configurable*. Your system will read the property file named "whist.properties". In your deliverables, you must provide the following ***three property files*** to demonstrate your system with the specified variations. The system should support repeatable runs (subject to interactive player choice) with a fixed random seed of "30006". All the properties make the card face up (like Figure 1).

- `whist.properties`: one interactive player (Player 0) and three random NPCs, and game settings as per the original system.
- `legal.properties`:
  - Each player has 6 cards instead of 13 (nbStartCards = 6)
  - The game ends when the player receive a score of 6 (winningScore = 6)
  - *Three* players are advanced NPCs with the following configs:
    a) **Player 0** uses the naïve legal approach + random selection.
    b) **Player 1** uses the trump saving approach + highest rank selection.
    c) **Player 2** uses the highest rank selection (not using a filtering approach).
- `smart.properties`**:** Game setting as per the original system. *Four* players are as follow:
    a) **Player 0** is an interactive player
    b) **Player 1** is an NPC using the smart selection (not using a filtering approach).
    c) **Player 2 and 3** are random NPCs.

Remember that there might be more features in the future. Hence, you should apply your software engineering and design knowledge to make the system extensible and facilitate the addition of other future changes.

Note: Despite the graphics in Figure 1, the advanced NPC (especially for the smart selection) must not violate the principle of the game by using information that would not be visible to the NPC. For example, an NPC must not be able to see the cards in another players' hand.

The Base Package

You have been provided with an Eclipse project export representing the current system. This system runs in a configuration similar to the whist.properties file (but without a fixed seed).

To begin:

1. Import the project zip file as you did for Workshop 1.
2. The system uses the JGameCard library which is based on the JGameGrid framework. Make sure that the JGameGrid.jar file (which is in the dist.lib folder) is in the Java Build Path for the project (Project > Properties > Java Build Path > Libraries); if not, re-add it (using Add JARs …).
3. Try running by right clicking on the project and selecting **Run as... Java Application**.
4. You should see a window like that in Figure 1; you can then play the game as the interactive player.

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.

Note: By default, the simulation should run without a fixed seed, meaning the results will be random on every run. To have a consistent test outcome, you **need** to be able to specify the seed in the configuration file.

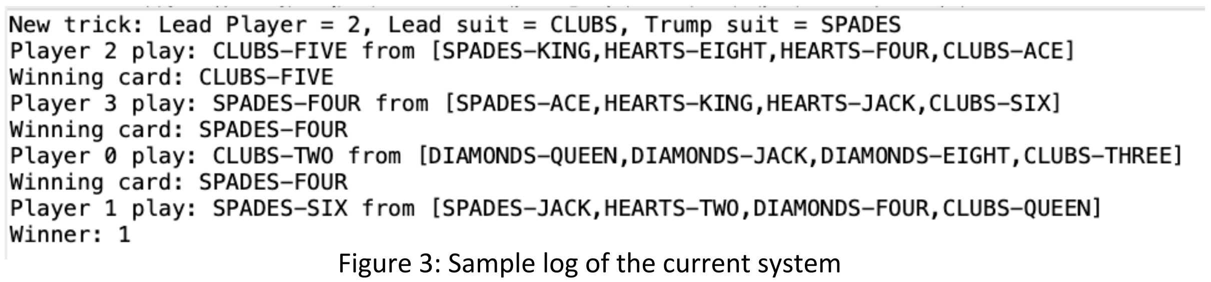
Testing Your Solution

We will build and run your program without using an integrated development environment. So, you must:

- Not change the entry point (i.e., it should remain as "game.Whist.main()"),
- Make your system read the property file named "whist.properties" for a configuration,
- Use the correct property file names (whist.properties, legal.properties, smart.properties), and
- follow the submission structure as described in the submission page in LMS.

We will test manually and analyse the log to pre-check the behaviour the system. So, you must preserve of the log output as the original system (see Figure 3). You can add more log if you wish but you should use different format from the original log output.

Note: It is **your responsibility** to ensure that you have thoroughly tested your software before you submit it. You may be penalised if your system cannot be built or run properly.



```
New trick: Lead Player = 2, Lead suit = CLUBS, Trump suit = SPADES
Player 2 play: CLUBS–FIVE from [SPADES–KING,HEARTS–EIGHT,HEARTS–FOUR,CLUBS–ACE]
Winning card: CLUBS–FIVE
Player 3 play: SPADES–FOUR from [SPADES–ACE,HEARTS–KING,HEARTS–JACK,CLUBS–SIX]
Winning card: SPADES–FOUR
Player 0 play: CLUBS–TWO from [DIAMONDS–QUEEN,DIAMONDS–JACK,DIAMONDS–EIGHT,CLUBS–THREE]
Winning card: SPADES–FOUR
Player 1 play: SPADES–SIX from [SPADES–JACK,HEARTS–TWO,DIAMONDS–FOUR,CLUBS–QUEEN]
Winner: 1
```

Figure 3: Sample log of the current system

Project Deliverables

As discussed above, you have to provide your system with a new design that support advanced NPC, support configurability, and facilitate the addition of other future changes. In particular, your system should

- provide an advanced NPC which uses a combination of one (or none) card filtering approach and one card selecting approach.
- be configurable according the parameters specified in the three property files. Three property files must be included (whist.properties, legal.properties, smart.properties) in your deliverables.

You are free to make whatever changes you wish, as long as the game play is preserved and still easily recognisable. If you have any questions, please make use of the discussion board, or ask your tutor directly as soon as possible; it is assumed that you will be comfortable with the package provided.