Background

A software company CardCraft Studio based in Melbourne is exploring new markets, and they have decided to get into the gaming industry. They want to create a digital version of card games, and their first project is a basic version of the **Pinochle** card game. However, the game is far from perfect. The team of software engineers who initially worked on it was inexperienced and didn't follow good design principles. As a result, the game's system is poorly designed, and the code is difficult to change or improve. There are issues with the base version of the code, and the company realises that they need to improve the system to make it scalable, easy to maintain, and able to support more advanced features in the future. The company has decided to contract your team to help redesign the game.

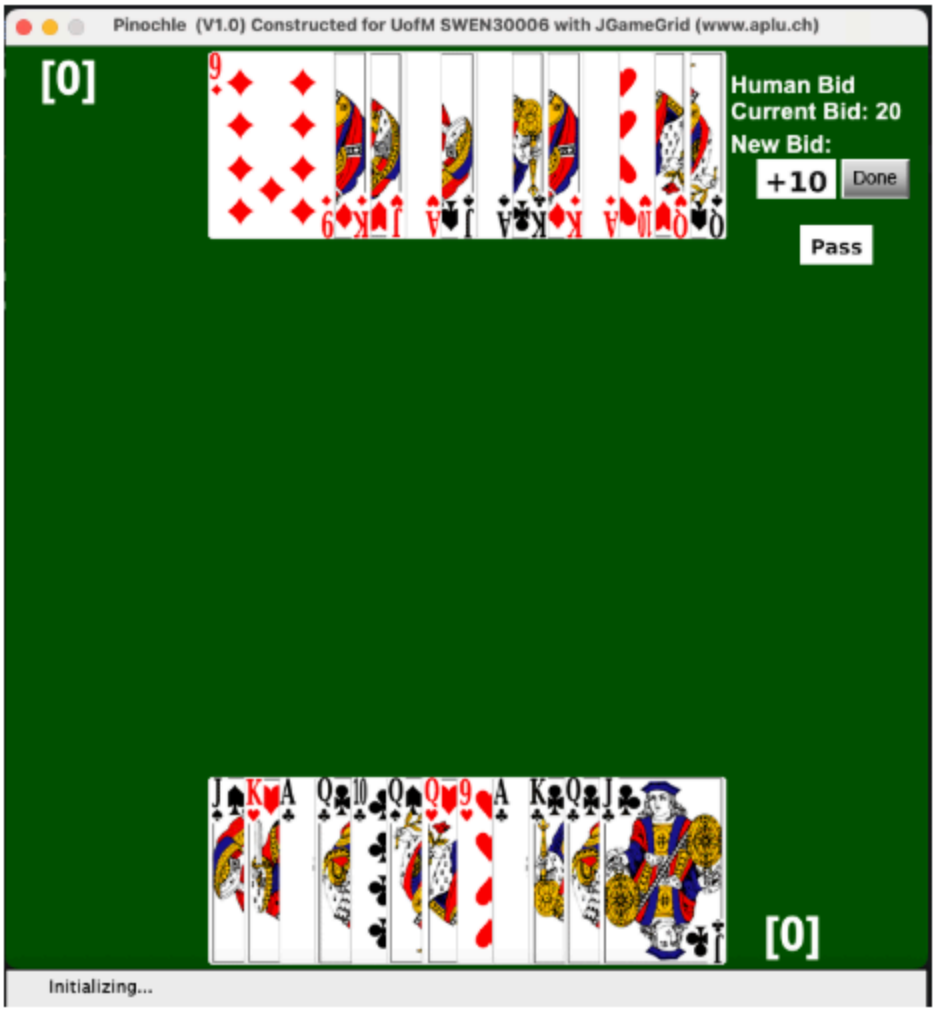

Figure 1: The GUI of the Pinochle game

What is Pinochle?

Pinochle is a trick-taking card game played with a special deck of **48 cards**. The deck consists of **two** copies of each card from 9 to Ace in all four suits (♠♣♦♥). Other cards are not used in the game.

Pinochle gameplay has the following steps:

1. **Dealing:** Each player gets a set number of cards from the deck.

2. **Bidding:** Players bid to decide who will set the trump suit (a powerful suit in the game).
3. **Melding:** Players show special card combinations called melds to earn points.
4. **Trick-Taking:** Players play a card in their hands to win the trick.
5. **Scoring**: Players earn points through melding and trick-taking. A penalty applies if the bid winner fails to earn the number of points they bid.

The game involves **two players**. The basic rules are explained below.

Dealing is the first step in Pinochle. This is when the cards are given to the players.

- Shuffle the cards thoroughly.
- Deal 12 cards to each player, three cards at a time, alternating between the two players.
- Remaining 24 cards form the stockpile (a face-down draw pile placed in the centre).

Bidding is when players take turns saying how many total points they believe they can earn. This is like making a promise. The points come from **melds** (card combinations shown before play) and **tricks won during play**. Players raise the bid or pass until one player remains—the bid winner. The bid winner gets the right to:

- **Declare the trump suit** (the powerful suit).
- **Lead the first trick** (start the first trick of the card played).

However, the bid winner must **earn at least as many points as they bid**. If they fall short, their **entire points will become zero and lose** the game.

For example, suppose Player 1 bids 300 points and Player 2 raises the bid to 320 points. Player 1 then passes, making Player 2 the bid winner. Player 2 now has the right to declare the trump suit and lead the first trick. However, if at the end of the round Player 2 earns only 300 points from melding and trick-taking—falling short of their 320-point bid—Player 2 earns 0 points and loses the game.

The bidding process is as follows:

- The first bidder is randomly chosen from the two players. There is no minimum bid number.
- The opponent **player** continues the bidding by either:
    - **Bid higher** than the previous bid by 10 points or by 20 points, or
    - **Pass** (say no, I won't bid anymore).
- This goes between the two players until one player has passed.
- If one of the players passes, the other player wins the bidding.

- The winning bidder (the player with the highest bid) declares the trump suit.

Melding is when players form **melds --** valid combinations of cards. After the bidding and declaring the trump suit, each player shows their valid melds to the other players to earn points. This is done before the start of the Trick Taking Phase. The table below shows a list of valid melds and their points. A player can win multiple melds.

- Each card in a hand can be used in only one meld, so the program should evaluate melds in order from highest to lowest scoring.
- If the cards in hand can form multiple identical melds, the points for each meld type are counted only once.
  For example, if a player holds **♥Q, ♥K, ♥Q, ♥K** (with ♥ as the trump suit), they can form two Royal Marriage melds. Therefore, the total score is 40.

Melds	**Points**
Ten to Ace Run of the trump suit. Ex: Suppose ♥ as the Trump Suit, then: **♥10, ♥J, ♥Q, ♥K, ♥A**	150
Ace Run + Extra King of the trump suit. Ex: Suppose ♥ as the Trump Suit, then: **♥10, ♥J, ♥Q, ♥K, ♥K, ♥A**	190
Ace Run + Extra Queen of the trump suit. Ex: Suppose ♥ as the Trump Suit, then: **♥10, ♥J, ♥Q, ♥Q, ♥K, ♥A**	190
Royal Marriage. King and Queen of the Trump Suit: Ex: Suppose ♥ as the Trump Suit, then: **♥Q, ♥K**	40

rick taking is when players play a card in their hands to earn points. The bid winner leads the first round (also called a trick), and then the winner of each trick leads the next one. A player wins a trick based on the trick rank: **Ace (1st) > 10 (2nd) > King (3rd) > Queen (4th) > Jack (5th) > 9 (6th).**

For each trick, a game is played as below:

- **The leading player** (the bid winner for the first trick) plays the first card called the "leading card".
- The opponent player plays a card in response.
- The trick is won by the player based on the following rules:
- If the opponent player has a higher trick rank in the same suit as the first player's card, they must play that higher trick rank. The opponent player wins the trick.
- Example:
    - Player 1 leads with Q♠.
    - Player 2 has K♠ and must play it.
    - Player 2 wins the trick because K♠ is higher than Q♠.
- If the opponent player does not have a card in the same suit as the leading player's card, they can:
    - Play a trump card (which always wins the trick, irrespective of the card);
    - Otherwise, play any other card in a different suit, and the leading player wins the trick.
    - Example 1:
        - Player 1 leads with 9♦.
        - Player 2 does not have any Diamonds, so they play A♠ (Trump suit).
        - Player 2 wins the trick because the Trump suit always wins.
    - Example 2:
        - Player 1 leads with 10♣ (Clubs suit, non-trump suit).
        - Player 2 has no Clubs, so they play A♥ (non-trump suit).
        - Player 1 wins the trick because Player 2 did not have a card in the same suit as Player 1 that was higher (Here – a Card in ♣ suit higher than 10). Even a higher card in a different suit would not count for a win if this card is not from the same suit (or any card from the trump suit). Here, even if A♥ is higher than 10♣, since the leading card was from ♣, Player 1 wins.
    - The trick winner takes the two cards that were played in a trick for points **and leads the next trick.**

Scoring: At the end of the 12 tricks, all cards at hand are exhausted. Each player counts all cards they won and earns points based on **the card values**, i.e.,

- Ace: 11 Points,
- 10: 10 Points,

- Jack: 2 Points,
- Queen: 3 Points,
- King: 4 Points, and
- 9: 0 Points. If the 9 Card is from the Trump Suit, it gets 10 points.

Trick points for each player are added along with their Meld Score for the game. **The player who has the higher score wins the game**. However, if the total points of the bidding winner are less than their bid, the total points will become zero, and loses the game.

Your Task

The current system can run the game with one human player and one computer player, following the rules described above. Right now, the computer player uses a random strategy (i.e., making a random decision to raise a bid or pass, and randomly selecting valid cards for trick-taking). Your task is to improve the design and extend the system to include the following additional capabilities.

1. **Support for Additional Melds:** Your extended system should include additional melds below and should be easily extended for additional melds in the future.

Additional Melds to be Supported	**Points**
Dix: 9 of the Trump Suit. Ex: Suppose ♥ as the Trump Suit, then: **♥9**	10
Ace Run + Royal Marriage (King and Queen of the Trump Suit). Ex: Suppose ♥ as the Trump Suit, then: ♥10, ♥J, ♥Q, ♥Q, ♥K, ♥K, ♥A.	230
Double Run (Two Runs in the Trump Suit). Ex: Suppose ♥ as the Trump Suit, then: ♥10, ♥10, ♥J, ♥J, ♥Q, ♥Q, ♥K, ♥K, ♥A, ♥A	1500
Common Marriage. King and Queen of the Non-Trump Suit:	

Ex: Suppose ♥ as the Trump Suit, then: ♦Q, ♦K	20
Pinochle – Jack of Diamonds and Queen of Spades. ♦J, ♠Q	40
Double Pinochle – Both Jacks of Diamonds and Queen of Spades. Ex: ♦J, ♦J, ♠Q, ♠Q	300
Aces Around – One Ace of each Suit. Ex: ♥A, ♠A, ♦A, ♣A	100
Jacks Abound – All 8 Jacks. Ex: ♥J, ♥J, ♠J, ♠J, ♦J, ♦J, ♣J, ♣J	400

2. **Smarter Bidding for a Computer Player**: Instead of making random bids, the computer player will bid based on a set of rules (smart bidding) as outlined below:
    - **If the computer player is the first bidder**, its opening bid will be equal to the total meld score of its hand.
    - **For subsequent bids** (or if the computer player is the second bidder), the computer will:
        - Prepare to raise the bid by **20 points** if it has **6 or more cards** in the same suit;
        - Otherwise, prepare to raise the bid by **10 points**.
        - The bid will only be made if the **new bid** (previous bid + planned increase) is **less than** *the total meld score, plus* the **maximum** of the following two values. Otherwise, it will pass.
            - The total card value of the **majority suit (**the **suit that has the most cards**), or
            - The total card value of the suit that contains the **most Aces, 10s, and Kings.**

During this process, the computer player assumes that the suit with the most cards in its hand will be the trump suit. If there is a tie (more than one suit has the same highest number of cards), it will randomly choose one of these as the trump suit.

3. **Support a cut-throat mode**: If a game is enabled in a cutthroat mode, an additional step after bidding will be in place in gameplay.
    - The dealer (the player who lost the bid) flips over the **top two cards** from the stockpile and places them face-up (shown in the middle of the GUI gameplay).
    - The non-dealer (winner of the bid) chooses **one** of these 2 face-up cards to add to their hand.
    - The dealer takes the **remaining** face-up card.
    - The rest of the cards in the stockpile are drawn automatically alternately between the two players (the GUI doesn't need to show this process):
        - The winner of the bid draws first, followed by the loser
        - Continue drawing one card each until the stockpile is empty.
    - After drawing all cards, each player will have 24 cards in their hands (shown in the GUI)
        - From these 24 cards, each player chooses the **best 12 cards** to keep for the rest of the game (melding and trick taking). For this extension, the computer player can choose to discard cards from the non-trump suit that has the fewest cards (proceeding by the next suit) till 12 cards are identified for discarding.
        - Discarded cards are not used in the game further for the Trick-Taking phase.
    - For this feature, you need to extend the auto code as well. The test5.properties contains extra properties for this mode (players.i.extra_cards, players.i.final_cards). The players.i.extra_cards dictate what cards each player will pick from the stockpile, and the final cards for the players.1 is predetermined. Your code needs to implement logic that the final cards for the computer players pass the test 5.
4. **Smart Trick-Taking for a Computer Player:** Instead of randomly selecting cards that follow the rules, your extended system should support a smart mode for the computer player. In a smart mode, the computer player must keep track of all the cards played and use this information together with the knowledge of the cards in its hand to try to maximise its score. The smart computer player must produce cleverer play than the random player. (You will have to explain how your design supports the smart mode.)

Note: We are aware that some edge cases may not be covered in this specification. In general, we expect the extended system to behave as described. Reasonable variations in handling edge cases are acceptable.

Provided Package

The package is provided as an IntelliJ project in a GitHub repository:

https://classroom.github.com/a/cV3KxRnB

Links to an external site.

Build and run the project: You will see the GUI appear and can play the current version of Pinochle using the mouse actions.

The classes in the provided 'app/src/main/java' package primarily handle the game behaviour where the GUI operates by using the JGameGrid

Links to an external site.

library. You can refer to the classes and their functions in this framework in Java Doc

Links to an external site.

.

- Most of the required changes for this project relate only to the game behaviour. You will need to modify the GUI code for the **cut-throat mode**
- Implementation hints: The JGameGrid library has provided many useful functions that you can reuse to implement the proposed extensions. The codebase should also provide examples for implementing the extensions.

The system reads a properties file that specifies the player type for each of the players, some initial cards that each player is dealt with, auto-play configuration (for testing purposes) and some of the cards the player will play in auto-play mode. You can edit the properties in these files for your own testing but make sure to preserve the default behaviour.

The 'logResult' string in the Pinochle class is for testing your system. Please see below Testing Your Solution for more details. Image files of characters and items are provided in the 'sprites' folder (app/src/main/resources/spites). The file names of these image files must not be changed.

Testing Your Solution

We will be testing your application programmatically, so we need to be able to build and run your program with a gradle build script. This is the same structure as the workshop 0. You need to follow the following guideline for gradle build to work:

- You should use the gradle script that we have set up for you to run the project.
- The entry point must remain at Driver and the Driver file needs to remain at `src/main/java`.
- Whenever you push to Github, Github should automatically build and run the test in your code with the test cases already provided by us in the project. You are not required to run the test in your local machine. When you are developing the app, please ensure that your code passes the testing when pushed to Github.
- Initially, your code will pass test case Original and fail other test cases. You need to implement the requirement and pass all test cases. Your code needs to pass the automatic tests as well as the manual tests that teaching staff will perform on your final submission. For manual tests, we will run the app as a human player to play against random, basic and clever players.
- We have provided the sample log that the project needs to generate to pass the test case. Your log needs to follow the format but does not need to match the sample log 100%.

You must not change anything in the test directory as we will overwrite all your changes with our default values when testing is executed. You must also not change the build.gradle and settings.gradle file as we will. It means that any other keys or values that you add into your properties files in the app directory will not work in testing.

Four properties files are provided as samples for development (gameX.properties) and five properties files are provided as the key resources for testing (testX.properties).

The auto mode may control some player for some card selections, and then it may stop after a few steps. When the auto mode stops, your program will need to take over. This is especially important for cut-throat mode.

You need to ensure that any modification still preserves the original behaviour, i.e., logResult generated by your extended version can pass the testing that we provide.

For your understanding, below are some key gradle build explanations. Note that you are not expected to change anything outside of your development directory and documentation directory:

- `settings.gradle`: to set up the name of the project and the main development directory

- `app/src`: where the development and testing code is
- `app/src/build.gradle`:set up the dependencies, the language version and the main class
- `app/src/main/java`: where your development code is
- `app/src/main/resources`: where your properties files and images are
- `app/src/test/java`: where the testing code is
- `app/src/test/resources`: where the properties files and images are
- To load a property or text file: you need to use `CLASS_NAME.class.getClassLoader().getResourceAsStream(propertiesFile)`
  This is already done for you, so you only need to be aware of it if you need to read any other text file.

Project Deliverables

1. Your version of the Pinochle game
   Submit your whole project (with any/all libraries used included, with all the gradle structure and gradle setting files)
   - Ensure that your code is well documented and includes your team's name and team members in all changed or new source code files.
2. Report
   A design analysis report detailing the changes made to the project and the design analysis including identifying design alternatives and justifying your design decisions with respect to design patterns and principles (GoF and GRASP). The report must also include your strategy for Clever computer player.
3. Software Models
   To facilitate the discussion of your design, the following software models should be provided in the report and used along with the discussion. You can use sub-diagrams or provide additional diagrams where appropriate.
   - A domain class diagram for capturing the covering the domain concepts relating to computer players
   - A static design model (i.e., a design class diagram) for documenting your design relating to computer players
   - Additional static and/or dynamic design models to support your explanation as you judge useful and appropriate.