# Design choices for the coursework

For the java coursework java was used with the code editor visual studio code, Intelij and eclipse. Eclipse was used to develop the tests and visual studio code and intellij was used to work on the main program. The design has changed a lot during the planning phase.

A screenshot of a computer program

Description automatically generated

## Card game class

In the above diagram we have decided to use arraylist to store the player objects and cards. We decided to use ArrayList<String> to store the cards since this will be needed to get written to the files and no mathematical operations needed to get done so using a string makes more sense. The Pack contains a list of string and that pack will get split into decks. The amount of decks depends on the number of players. This class inherits all the other classes.

## Player class

The player class consists of an ID a starting hand and their preference. The preference of the player is the same as their ID. It can be quite hard to change the player card preference since a loop is used to assign the players their ID with their preference. The starting hand is an arraylist object which is a string so no conversion is needed.

## Pack class

The pack class is simple since it goes through the file and rads it all into an arraylist.

## Deck class

The deck class is simple since each deck has it’s own ID because the ID is needed to know which deck each player needs to draw and discard to. Each deck also has 4 cards since each pack has 8n players and each player gets 4 cards.

# How the code works and design choices.

## Inputs

After the code program is run the file is read and the input is taken for number of players, the error handling is applied on the pack and player to make sure it’s in the right format.

## Deck and player objects created

After the inputs are taken the deck objects are created  according to the input and the player object is also created according to the input with the preferred card being their ID. The ID of the decks and players is usually their number e.g. first player has ID 1 and first deck has ID1 and so on.

The hand is assigned to the players in a round robin fashion using 2 loops e.g. in a game of 3 players player draws cards: 1,4,7,10. Player 2 draws 2,5,8,11. Player 3 draws 3,6,9,12. This is also how the decks get their cards in a round robin fashion. But this happens after the player gets their starting hand. So the decks get their cards from the second half of the pack.

## Threads

Each thread then starts simultaneously for the player. Each player has one thread. This thread draws from the deck with the same ID as the player then discards to the one above. If there is no deckID one above the player. The player discards to deckID 1. If the first card is not the Preferred card for the player (with ID1). The player then draws a card from deck ID1. This drawn card is placed between the first and second card in the player's hand. (This is so a player does not hold on to a card indefinitely). The player then discards the first card to the button of the deck. Then the player goes on to the next card in their hand. If the card is preferred, the player goes on to the next card. This cycle repeats until a player has won.

This thread then writes to the player files accordingly.

## Outputs

A file is created for each player and each deck. When a player does and action it will say which deck he drew from and which deck he discarded to. At the end of the game the winning player will say that they won in their file. This will cause the shared flag to be true stopping the game and every player shows their final hand and every deck outputs their final cards.

## Known issues

The game can not determine if a deck has not got a winning solution. This was not put in the game as the specification says that it would not need to be. In such an instance the game would crash. The game may also crash if their preferred card choice is not in the deck they they will constantly discard their whole hand. In rare situation there can be 2 or more winners. This can happen when the players draw the card they need at the exact same time. In this situation the game will show more than one winner. But we will never know who drew the winning cards first.

# Tests

For the tests junit 4 was used. In the program there is a separate file detailing the methods we have tested. Some of these methods were hard to test since they can not be static. To test these other methods, error handling was added to the main program to make sure everything was working.

In the test file you can specify the amount of players in the game and deck that is being used. This will then run the specified classes to make sure the inputs you entered are correct

## testLoadDeckFileFormatted.

This makes sure the deck that is being used is formatted correctly. E.g. they are all integers. The deck is not empty and the deck is not a multiple of 8. This ensures each deck is in a valid format and means no future errors will occur

## testNumberOfPlayers

This test makes sure the size of the pack is a multiple of 8 since this is what the specification says. Also makes sure number of players is not negative.

## testSplitPack

Makes sure pack is split into the same number of decks as there is players.

## testCardsInEachDeck

Each deck is supposed to have 4 cards to this method is used to test that.

Development Log

# Error handling

Simple Errror handling has been done so the user can’t input an invalid deck and also user will have to input a positive number that is not 0.

# CommandLine Testing

The commandline was used to see what part of the code runs and if the code runs and outputs the files a intended. The commandline will say who won and will also give information on what file was written to and when. This was a very good tool to test the program.

# Development log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Time:start | Duration: Time end | Roles | Signed |
| 26/10/24 | 12.30 | 14.00 | Planning | 022132  030834 |
| 04/11/24 | 13:30 | 15.30 | Student:030834 coded while the other watched | 030834  022132 |
| 11/11/24 | 15:30 | 18:30 | Student:030834 coded the main system and developed some tests. | 030834  022132 |
| 16/11/24 | 17:30 | 20:30 | Student:030834 coded the main system | 030834  022132 |
| 24/11/24 | 17:30 | 18:30 | Student:030834 coded on the coursework and added comments. | 030834  022132 |