Trianta Ena

- Object design (emphasis on the extendibility of TTT game infrastructure)
- Implementation

Classes:

- Game.java — interface for games (TTT, OAC, Trianta Ena)
- Main.java — main class to run the whole package.

Players interface

- Player.java
  - should have attribute "money" to record current money
  - should have attribute "bet" to record current bet
  - should have attribute "hand" to record current card values in hand

- Dealer.java
  - attribute "hand"
  - "money"
  - *: Automation of playing game

→ Combine to one class: TE-Player.java and set up a boolean for that

- Hand.java — class of recording the current values in player's hand
  - should have attribute "Card[] theHand"
  - attribute "number of cards"

- Card.java — class of recording value and suit of cards
  - attribute "suit"
  - attribute "value"

- Deck2.java — class of two decks of cards
  - attribute "int nextCardIndex"
  - contains two decks of cards and shuffling functionality

- Exception.java — hold any exception

- TriantaEna.java class of the whole game

- Table.java. records all the methods used in TriantaEna.java

  start-deal
  ask-bet
  deal_two_cards
  one_round

△: Players: TE_Player[]
  CurrentPlayers: ArrayList<TE_Player>

Extendibility of PA1:

- TriantaEna class is implemented from Game.java which is from Jay's PA1. It is one other game than TTT or DAC. It overrides three methods from the interface: check_winner(), one_round(), and play_game() play_game() is to start the game. one_round() function is to run one single round of Trianta Ena. check_winner() checks if players or dealer win at each round.

- TE_Player class is extended from Player.java which is from Jessica's PA1. Player.java class has several variables like playerNum name, piece, and winNum. We create TE_Player class for our PA2 to describe the two kinds of players: players and dealer. We set up the parameters one-by-one.

- There are three classes implementing Exception class which is served for error checking of cards/decks classes. This is beneficial since we can avoid writing duplicate codes to throw same exception in different situations.

To give a graph of the classes: