

Exercise_01

Ramona Walker, Dominik Johann Arnold, Mark Woolley, Otto Buck

November 2022

1 CRC Cards

These are the final CRC Cards.

Name: Card	Superclass:
Responsibilities: Return the card type	Collaborations: CardType

Name: CardType	Superclass:
Responsibilities: Return the ruleset of a card type	Collaborations: Ruleset

Name: Deck	Superclass:
Responsibilities: Shuffle	Collaborations: -
Create Deck	DecSpec
Draw	Card
Return number of cards left	-
Return if deck is empty	-

Name: DeckSpec	Superclass:
Responsibilities: Build DeckSpec	Collaborations: CardType
Get amount of card	CardType

Name: DiceCombo	Superclass:
Responsibilities: Store possible combos	Collaborations: -
Return points	-

Name: DiceSet	Superclass:
Responsibilities: Create set of dice	Collaborations: Die
Roll the remaining dice	Die
Pick up all the used dice (after tutto or lost turn)	Die
Show all valid dice combos	DiceCombo, DieValue
Return if set is empty	-
Return amount of dice left	-
Return total size	-

Name: Die	Superclass:
Responsibilities: Roll	Collaborations: DieValue
Get the value	DieValue

Name: DieValue	Superclass:
Responsibilities: An enum of all possible die values	Collaborations: -

Name: DefaultParser	Superclass:
Responsibilities: Filter non-valid answers from console	Collaborations: -
Translate answers (Y/N) to boolean	-

Name: Game	Superclass:
Responsibilities: Initializing the game	Collaborations: Deck, Tableau, DefaultParser, DeckSpec
Play the Game	Tableau, Card, Ruleset, Round, DefaultParser, Deck
Determine Winner	Tableau

Name: Round	Superclass:
Responsibilities: Enables rules of the given ruleset	Collaborations: Ruleset
Play round	InputParser, Ruleset, Diceset, DiceCombo
Return what to do next round (draw a new card or keep the same)	Ruleset

Name: Tableau	Superclass:
Responsibilities: Add player	Collaborations: -
Update points of players	-
Returns if a player has won	-
Return points of player	-
	-

2 Following the Responsibility Driven Design, describe the main classes you designed to be your project in terms of responsibilities and collaborations; also draw their class diagram.

- **Die**

A *Die* represents a physical game die.

Responsibilities: A *Die* has to be able to roll and should return a random integer between 1 and 6. **Collaborations:** *Die* collaborates with *DieValue*, an enum representing all values a *Die* can have.

- **DiceSet**

DiceSet is the collection of *dice* that are used to play the game.

Responsibilities: The DiceSet has to know, which dice are used. It should be able to roll the dice and show all valid combos.

Collaborations: DiceSet obviously collaborates with *Die*, since it consists of six dice. It also collaborates with *DiceCombo* and *DieValue* to show all valid combos.

- **Ruleset (and subclasses)**

Ruleset is a abstract class which has to decide how a round is played. There are subclasses of Ruleset, one for each card type.

Responsibilities: Each subclass of Ruleset represents a card type. Each of these has to know:

- When to draw a new *Card*
- Bonus points and special actions after Tutto.
- Display the rules of the ruleset.
- Which dicecombos are valid (i.e. for straight only singles of dice not used yet.)
- What happens when you throw a null?

Collaborations: Ruleset has no collaborations.

- **Card**

Card represents the cards of Tutto. Each card has a *CardType*, which determines its *Ruleset*. The difference between *Card* and *CardType* is, that there can be multiple instances of *Card* with the same *Cardtype*.

Responsibilities: A card has to know which *Cardtype* it has.

Collaborations: Card only collaborates with *Cardtype*.

- **CardType**

A CardType is an enum representing all the cards with the same *Ruleset*.

Responsibilities: A *CardType* must know which *Ruleset* applies.

Collaborations: *CardType* only collaborates with *Ruleset*.

- **Deck**

A *Deck* is a collection of *Cards*.

Responsibilities: A *Deck* has to be iterable, such that one can draw from the *Deck*. It has to know when it is empty and be able to reshuffle. It has to know how many of which *cards* it contains, when it is build.

Collaborations: *Deck* collaborates with *DeckSpec* to get the right amount of *Cards* of the right *CardType*. It has to collaborate with *Card*, since it consists of *Cards*.

- **Tableau**

Tableau represents the game table. It is a collection of Players, with their names, points and order. We decided that it is not necessary to have player as a separate class, since *Tableau* has already little responsibilities.

Responsibilities: *Tableau* should be able to add a new player, update their points and keep track of the order. *Tableau* can tell if a player has reached 6000 points.

Collaborations: *Tableau* has no collaborations.

- **Round**

Round splits the game in smaller bits, ending when a player rolled a Null, decides to stop or when he achieved Tutto.

This makes it easier to keep track of the points and what to do, when a Tutto happens.

Responsibilities: A *Round* has to enable the rules of the given *Ruleset*. This includes giving back, what happens in the next round (should a new card be drawn or keep the same one?). *Round* should be able to play a round, starting with a *Ruleset*, ending with Null, Tutto or when the player decides to stop.

Collaborations: *Round* collaborates with *Ruleset* to enables its rules including telling if a new card should be drawn. It collaborates with *InputParser*, *Ruleset*, *Diceset* and *DiceCombo* to play the round.

- **Game**

Game is the class connecting the main components to be able to play the game. It initializes a *Tableau*, a *Deck* and lets the players play their turn.

Responsibilities: *Game* is responsible to initialize the game, play the game and to declare the winner and end the game.

Collaborations: *Game* collaborates with *Deck*, *Tableau*, *DefaultParser* and *DeckSpec* to initialize the game. It further collaborates with *Tableau*, *Card*, *Ruleset*, *Round*, *DefaultParser* and *Deck* to play the game and it collaborates with *Tableau* to declare the winner.

- **DefaultParser**

The *DefaultParser* makes shure that only useful input gets through.

Responsibilities: The *DefaultParser* is responsible to filter all non valid answers from the console. The valid answers will be translated in more suitable inputs ($Y/N \rightarrow 1/0$).

Collaborations: none

3 Draw the class diagram of the aforementioned main elements of your game.

