Assignment 1

Team number: 2
Team members

Name	Student Nr.	Email
Sebastian Fredrik	2669832	s.r.fredrik@student.vu.nl
Federico Giaj Levra	2674188	f.giajlevra@student.vu.nl
Kevin Koeks	2680522	k.a.b.koeks@student.vu.nl
Jelena Masic	2645593	j.masic@student.vu.nl

Introduction **2**
Player types 2
User Interface 3
Bonus feature - Networking 3
Bonus feature - Scoreboard 3
Bonus feature - Customizable Deck 4
Features **4**
Functional features 4
Quality requirements 5
Java libraries **6**

Introduction

Jelena Masic

Our system implements the Exploding Kittens original edition game. The base deck is composed of 56 cards: 20x *cat* (i.e., 5 sets of 4 matching cards), 4x *exploding kitten*, 6x *defuse*, 5x *nope*, 4x *attack*, 4x *skip*, 4x *favor*, 4x *shuffle*, 5x *see the future* cards.

A n-player game evolves as follows:

- Remove all *exploding kittens* and *defuse* cards from the deck
- Shuffle the deck of cards
- Give 7 face-down cards and a *defuse* card to each player
- Insert the remaining *defuse* cards by the following criteria:
  - for a 2- or 3-player game, insert only 2 *defuse* cards in the deck
  - otherwise, insert all the remaining *defuse* cards in the deck
- Insert n - 1 *exploding kittens* in the deck
- Shuffle the deck of cards
- Choose randomly the player that goes first
- Until two players are alive in the game, play turns in a clockwise order

A turn consists of playing any number of cards and all turns end by drawing a card.

The different types of cards have the following effects:

- *exploding kitten*: if drawn, the current player loses unless they play one *defuse* card
- *defuse*: counters the effect of an *exploding kitten*
- *shuffle*: the deck is shuffled until told to stop
- *favor*: one player receives a card from a selected opponent
- *skip*: ends the turn of the current player without drawing a card
- *attack*: ends the current turn and causes the next player to play 2 turns (stackable)
- *see the future*: the current player sees the 3 cards on top of the deck
- *nope*: counters any action (except *exploding kitten* and *defuse*) from an opponent
- *cat*: no effect if played alone

The game also has *special combos* defined as follows:

- the current player can play 2 matching cards to steal a card from an opponent
- the current player can play 3 cards with the same title to demand a specific card from an opponent (nothing happens if the opponent doesn't have the requested card)
- the current player can play 5 different cards to draw a card from the discard pile

Player types

Our system allows matches with human and/or computer players; the former play by interacting with a user interface, the latter play by means of artificial intelligence based on random choices. The number of players can vary from 2 to 5 and it's possible to have more than one human player.

This means that, in case we would have followed the original rules of the game, it would be cumbersome to play *nope* cards because all human players should be able to play them in response to the action of an opponent. We thought of several ways of addressing this issue:

1) use hotkeys, i.e., a different hotkey for each human player, to play *nope* cards, but this would imply that all human players would look at the screen at the same time to decide whether to play a *nope*, hence we didn't choose this solution;

2) after every action, poll every active player to check whether they want to play a *nope* card with a timeout, but this solution would have made the game not enjoyable;
3) automatically play a *nope* card in response to an action among *shuffle*, *skip*, *see the feature*, or an *attack* or *special combo* against the player holding the *nope* card. We decided not to opt for this solution since this is a strategy game, and all human players should be able to play the *nope* card consciously; or
4) <u>in a local environment having just one player against all AI players, while a network environment has only human players.</u> We use **this** solution because we consider it a clean and simple way to solve the issue.

This setup also avoids having an opponent watching someone else's cards, or coordinating interaction, e.g., stealing cards and activating the *favor* card between human players.

Finally, *nope* cards are played automatically by AI players, while human players can decide to play a *nope* card within a tunable amount of seconds from the last action in a turn.

User Interface
The game has an initial screen, where it's possible to decide how many AI players are in the game, with this number limited to 4. In this screen, human players can choose a username for the match.

From the initial screen, it's possible to start a match. Since there is only one human player per application instance, human players always see their hand of cards on the screen on a UI.

When defusing an *exploding kitten*, a human player can decide at which position the kitten will be placed. This happens by means of a prompt with a number selector that allows the player to choose the new position of the exploding kitten in the deck. Alternatives to this approach could have been forcing the reinsertion of the kitten at a given or randomly position, or a more elaborate UI to reinsert the card. These alternatives have intrinsic limitations or are just too complicated. Hence, we tried to strike a balance between the two extremes.

Finally, after the game ends, the player can play a new game or they can go back to the initial screen.

Bonus feature - Networking
To make the system more interesting and appealing, we decided to implement a networked mode for it. This choice allows us to make the game more interactive, so that human players can also compete with each other. In a networked setup, up to 5 players (both human and AI) can connect to the same match. Also in this case, only one human player is allowed for each instance of the application.

Bonus feature - Scoreboard
Regardless of the mode used to play the game (i.e., local or networked), results of past matches are stored in a file, so that players can see their amount of wins and losses. The file (if present) is loaded at startup and the initial screen features an option to display the scoreboard.

Bonus feature - Customizable Deck
Another extension we thought of is loading a deck from a file. From the initial screen, players can decide to use an alternative deck to the original base deck. In case a player tries to load an invalid deck, there will appear an error message and the deck will not be loaded. For example, a deck with more *exploding kitten* cards than *defuse* cards is invalid. In the networked mode, it's only possible to use the original base deck.

Finally, the game Exploding Kittens had success in the real world and, along with other games such as Bears vs Babies, the game producers released several expansions such as Imploding Kittens and Streaking Kittens. Our implementation will *not* include cards from these expansions, but will be designed to easily integrate new card types.

Features

Author(s): Federico Giaj Levra

Functional features

The game and its mechanics are centered on the deck. Most of the features are therefore based on implementing the deck, while keeping in mind that it should be easy for one to add an expansion pack to the initial deck. The game will be played through a user interface locally.

ID	Short name	Description	Champion
F1	Cards Functionalities	The functionalities for each group of cards has to be implemented as described in the introduction. (here)	Jelena Masic
F2	Deck Functionalities	The original Deck is composed by 56 cards, divided as explained in the introduction (here), we would like to implemented so that it can be instantiated with any combination of the cards and it will have the following functionalities which allow for the use of it: 1. Pick a card 2. Shuffle the deck 3. Insert a card in the deck in a given position	Kevin Koeks
F3	Discard Deck	When a card is played it is inserted in the discard deck. The discard deck allows for the following functionalities: 1. Add card(s) to the top. 2. Take a card from the discard deck	Kevin Koeks
F4	Combos	The following combos are available: ● Two-card combo, three-card combo, five-card combo. There is a detailed description in the introduction (here)	Jelena Masic
F5	Game mechanics	Different phases of the game have to be developed, such as: ● Game start, switching turns, attack turns, playing a card, keeping track of a players hand, Game end	Federico Giaj Levra
F6	User Interface	We choose to implement a CLI as a method to play the game. There is detailed description in the introduction (here)	Sebastian Fredrik
F7	Basic AI	In order to allow a player to play locally, we will implement a Basic AI which one can play against. The basic AI will randomly choose a card in its current hand, with an exception for the Defuse and	Federico Giaj Levra

		Nope cards, which it will only be able to play in certain conditions. The Defuse card will only be played if the AI picks an exploding kitten, while the Nope card will only be played when it is not its turn and when it is possible. Nope cards are randomly played (with a given probability) by AI players.	

Quality requirements

Author(s): Sebastian Fredrik & Kevin Koeks

ID	Short name	Quality attribute	Description
QR1	Keeping your cards	Reliability	If a player doesn't discard or play a card, the card remains in the player's hand, including in possible edge cases.
QR2	Extensible deck	Maintainability	The card game shall be easily extendable in terms of types of cards.
QR3	Fast actions	Performance	Once the player's turn starts it should at most take 1 second for all cards to load in and afterwards 1 second to use a card, to avoid making the wait irritating.
QR5	Fast next turn	Performance	Once the previous player's turn has ended, your turn should start within 5 seconds
QR6	Accessible parts	Usability	The players should be able to make actions with one command. Contextual text is present to help the players identify which cards they can play from their hand.
QR7	Responsiveness of features	Availability	We expect the features to complete successfully at least 95% of the times.
QR8	Player's privacy	Security	All the players should be secured from being accessed by an unauthorized person.

Java libraries

Jelena Masic, Sebastian Fredrik

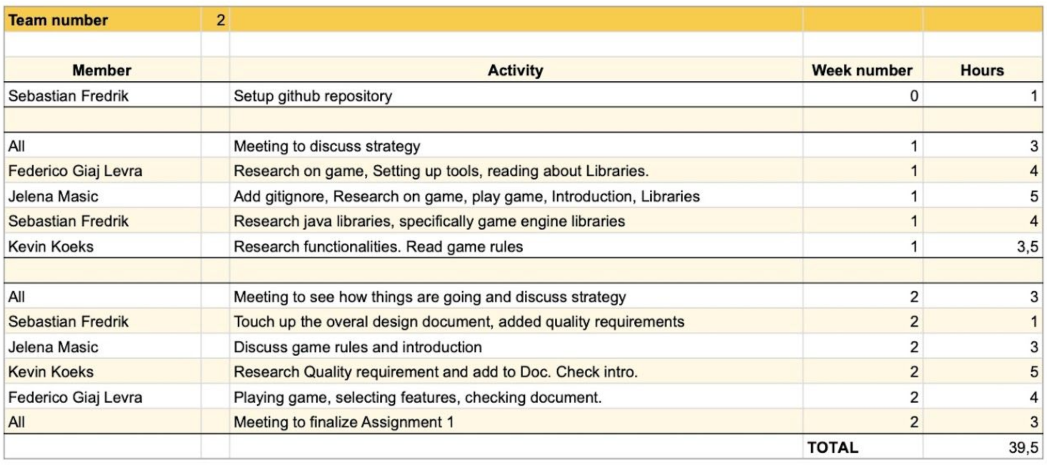
Text IO
This lightweight library for CLIs (Command Line Interfaces) provides useful classes such as TextTerminal or TextIO to interact in a type-safe and high-level manner with the terminal. Thanks to this library, we don't have to write a dedicated set of classes for parsing ourselves.

Fastjson
We decided to use this library from Alibaba to have access to high-speed API to marshal to and unmarshal from JSON strings. This can be useful in several of the bonus features, for example updating and reading the scoreboards, loading a deck from a file, or in the networked mode. Preliminary experiments, the company that backs the library, and the popularity on GitHub make us think that this library is a reasonable choice for our project rather than other alternatives/formats such as protobuf.

Apache Commons Lang
Manipulating Strings by hand and writing custom-made functions is error-prone compared to using a well-maintained Apache library, hence we decided to include Apache Commons Lang3. These API are null-safe , while Java's API are not, e.g., invoking equals on a String.
Furthermore, this library also contains classes such as Pair or EnumUtils, so that we don't have to reinvent the wheel for common utility methods.

SLF4J
This logging library has widespread adoption in the industry and it is easy to integrate with Lombok. Thanks to this library, we can resolve in one sweep a number of SonarLint issues by replacing prints to STDOUT, but also decide to (for example) redirect logs to file. Despite we don't strictly need logging, this library will be convenient to gather some metrics on quality requirements such as responsiveness and amount of errors.

Time Log link



Team number	2			
Member		**Activity**	**Week number**	**Hours**
Sebastian Fredrik		Setup github repository	0	1
All		Meeting to discuss strategy	1	3
Federico Giaj Levra		Research on game, Setting up tools, reading about Libraries.	1	4
Jelena Masic		Add gitignore, Research on game, play game, Introduction, Libraries	1	5
Sebastian Fredrik		Research java libraries, specifically game engine libraries	1	4
Kevin Koeks		Research functionalities. Read game rules	1	3,5
All		Meeting to see how things are going and discuss strategy	2	3
Sebastian Fredrik		Touch up the overal design document, added quality requirements	2	1
Jelena Masic		Discuss game rules and introduction	2	3
Kevin Koeks		Research Quality requirement and add to Doc. Check intro.	2	5
Federico Giaj Levra		Playing game, selecting features, checking document.	2	4
All		Meeting to finalize Assignment 1	2	3
			TOTAL	39,5