Exploding Kittens Card game
Java implementation
Team 12
Assignment 1

Team members:

Name	Student Nr.	Email
Andrei Dragomir	2669304	a.dragomir@student.vu.nl
Francesco Marciani	2677750	f.marciani@student.vu.nl
Adam Ristov	2659483	a.ristov@student.vu.nl
Sander Vrzina	2678335	s.vrzina@student.vu.nl

Introduction

All team members participated in writing this section.

We are going to implement the game Exploding kittens. The rules we are going to use are taken directly from the official website, and we aim to implement all types of cards, if time allows it. The project will start out as a command line game, and later

- The game can be played with human players as well as a.i.
- The game works clockwise, each player playing all the actions cards he wants, then finishes his turn by drawing a card (unless a card used allows avoidance of this action). The goal of the game is to be the last one that remains alive, “death” occurring when you pick an Exploding Kitten card from the deck and you have no way of defusing this certain card (with a Defuse card).
- We will implement the full functionality of the real card game (all action cards, all special combos).
- Server-Client Structure: The players will act as clients, sending their actions to a server which processes all the information from all the players and then sends the updated version of the game contents to all players. By working in this manner, we leave in the possibility of integrating over the internet multiplayer without having to change the infrastructure of our project.

Don’t forget to use links to external URLs (e.g., the direct link to the project manager tool you are getting inspiration from, the link to the original video game you are getting inspiration from, etc.), if applicable.
https://explodingkittens.com/how-to-play/exploding-kittens

Features

All team members participated in writing this section.

Functional features

We chose each champion based on a random shuffle and decided to assign the same champion to F1 and F2 due to the intertwined nature of the two.

ID	Short name	Description	Champion
F1	Server - Networking	The networking aspect of the server, responsible for the communication with the client. It has to manage connections to multiple clients (possibly multiple games). Parsing messages to and from the clients, and connection stability (halt game in case of connection loss).	Andrei
F2	Client - Networking	Much like the server side of networking, it handles the connection to the server, and parses incoming messages / sends action messages when a user - Plays a card - Ends their turn - Draw a card	Andrei
F3	Server - Game state	The part of the server responsible for the game state. It tracks the status of all the decks and player hands, and updates such statuses when required (ex: a player plays a Shuffle card). This is basically the game itself. At its most basic iteration it should be able to handle: - Game setup - All types of cards - At least 2 players - Game ending	Adam
F4	Extensible deck	The deck size and specific cards will be specified in a JSON file that the user can modify, and gets loaded by the server at start-up. We are going to include presets for slightly modified versions such as - Highly explosive (extra Exploding kittens) - Time traveller (extra See the future) and similar.	Sander
F5	Client - Command line interface	The client side of the game will solely act as the interface through which the user selects and requests a certain card in their hand to be played. On top of that, it will present a function which will act as a filter in order to prevent the user from trying to request a card to be played which would not be possible (because of its incompatibility with the rules of the game).	Francesco

If time allows it:

ID	Short name	Description	Champion
F6	Terminal UI/ GUI	The game has a graphical interface through which the player can play the game. During the development we will choose whether to go for a Graphical interface or a Terminal interface. This will depend on the complexity and how well documented the relative libraries are.	To be determined*
F7	Online Compatibility	The game has the option to be played over the internet in lobbies.	To be determined*

* Since we don't know for sure if we are going to implement these features yet, we have made them T.B.D. Considering the fact that some members might finish sooner, they could start working on this (if time allows it), but since we don't know who finishes first it will be determined later.

Quality requirements

All team members participated in writing this section.

In order to keep the game as reliable and playable as possible, whilst making the possibility of adding new features, i.e. adding new cards fairly easy, we came up with these qualitative requirements for the game.

ID	Short name	Quality attribute	Description
QR1	Commands sanity checks	Reliability	When the player issues a command, the syntax of the command shall always get validated against the format specified in F2.
QR2	Extensible deck	Maintainability	The card game shall be easily extendable in terms of types of cards, using a JSON file.
QR3	Instantaneous results	Performance	Once a player makes a move, it is instantly relayed to the server.
QR4	Concurrency of clients' statuses	Reliability	The game should be updated on all the players' screens before anyone can make a move.

Java libraries

All team members participated in writing this section.

As mentioned in the features, if time allows it we will develop a more complex interface for the game. We'll go for either a Terminal based UI or a Graphical interface. Here's a list of candidate libraries:

- If we go for a GUI, we'll use FXGL or Mini2DX. We are debating between the two as FXGL was mentioned in the list and it has many features, however, Mini2DX seems easier to use. This will be given more thought if the GUI path is taken, likely picking the one with easier and clearer documentation.
- Lanterna (if we choose the terminal-based UI path), due to the fact that it was recommended in the list and it seems fitting to our needs.
- Fastjson, which we'll use for reading and writing JSON configuration files containing the description of the cards of the deck. We chose it because it is easy to configure and use from Java code.
- KryoNet or Netty for networking, again depending on which one is more suited for our purposes and is documented better.

Time logs

Team number	12		
Member	**Activity**	**Week number**	**Hours**
All members	Define functional features	1	2
All members	Search Java libraries	1	1
All members	Discussing features	1	2
All members	Writing on the doc	1	3
		TOTAL	8