

## Assignment 1

Team number: 3

Team members

Name	Student Nr.	Email
Muhammad Ahsan	2663138	m.ahsan@student.vu.nl
Hao Qin	2658357	h.qin@student.vu.nl
Björn Oosterwijk	2651414	b.w.oosterwijk@student.vu.nl
Ashish upadhaya	2635493	a.upadhaya@student.vu.nl

## Introduction Author(s): Björn

The goal of this project is to design and implement a software system which enables a user to play the Exploding Kittens game against other (AI) players in a command-line interface.



### The game

Exploding Kittens is a turn-based card game in which the objective is to remain the last player in the game. Although the game dynamics are rather intricate, the general concept of the game is easy and explained next.

The deck of cards that is used consists of multiple types of cards, which all have (potential) abilities that can alter the flow of the game. These cards will be elaborated on in the next section. Each player starts with four random cards in hand, and one “defuse” card. Each turn follows the same pattern. First, a player has the option to play one or multiple cards from its hand. If a card is played, it is put on a discard pile and its effect takes place immediately. After a player is done playing cards, the turn is ended by drawing a card from the common draw pile and the play continues clockwise around the table. Players can be eliminated from the game when the card they draw at the end of their turn is an “exploding kitten” and they have no “defuse” card to counter this. The game continues until eventually one player remains.

The original, physical Exploding Kittens card game’s identity and popularity are derived for a large part from its funny card designs and witty card descriptions. The focus of this project is however on the game play and the functionalities of the game and it will therefore leave the witty gimmicks out of scope.



### The cards

Since Exploding Kittens is a card game, its gameplay is determined by the content of the cards in the deck. The type of cards used, and their corresponding abilities, will be briefly summarized here. It is important to first note that the cards explained are the essential cards, which are part of the original Exploding Kittens game. However, the makers of the game have designed numerous expansion packs with extra cards which have different abilities, which can be added to the original deck for a different experience. Our implementation should provide for these cards too and the next paragraph shortly touches on those. However, the indispensable cards that **shall** be included, and their corresponding actions from the players view, are the following:



**Exploding Kitten** - When you draw this card you are removed from the game unless you have a defuse card.

**Defuse** - If you drew an exploding kitten, you can play this card instead of dying.

**Nope** - Can be played anytime another player plays a card. It can cancel out any action, except for an exploding kitten or a defuse card.

**Attack** - End your turn without drawing a card and force the next player to take two turns in a row.

**Skip** - Immediately end your turn without drawing a card.

**Favor** - Force another player to give you one card from their hand.

**Shuffle** - Shuffle the draw-pile without looking at the cards.

**See the future** - Look at the top three cards of the deck.

**“Powerless” cards** – Play two equal cards and pick a random card from another player’s hand. Play three equal cards and tell a player which card you want from them; if they have it they need to give it to you.

The functional design of the expansion cards is similar to that of the original cards. That is, when a card is played, an action takes place. Therefore, the design of this project, and more specific the back-end representation of the cards, will be so that different types of cards can be added effortlessly to the software system. In this way, the design of the game shall be as independent as possible from the specific types of cards and will therefore allow for flexible adjusting of card properties. If time allows these expansion packs should be included in this project, to enable an even more profound game.

## The implementation Author(s): Ashish

First of all, we shall make a UML diagram and other helpful diagrams that will represent the workflow of the system. These diagrams will give the developers a certain idea about what they have to build and how it should communicate with other classes/systems. When these helpful diagrams are done, the programming of the system can start.

During this project we will be working according to the agile method. This means that everyone in the group is going to work on small parts of the system. Throughout the project these smaller parts will be merged to become one full system. This method has a huge advantage in comparison with other methods like the waterfall approach. The main advantages are the speed and flexibility that the agile approach enables. For example, if there is something wrong with the number of cards, you just need to go to the part of the system that deals with the cards and change what is needed. This adjusting can be done throughout the project and not only when the testing phase is done. Also using this method we will continuously work together and test the system every time a subpart is done and improve accordingly. This method requires a lot of collaboration and multiple iterations to successfully develop the system.

## Challenges and potential pitfalls Author(s): Ashish, Ahsan

The main technical challenge of this project is that the game must be as independent as possible from the specific type of cards used in the game. It should be easy to adapt/configure the game to new decks. This is in line with the philosophy of Exploding Kittens, which can be expanded with several expansion packs. This idea can be extended that our project can be adapted in other card games as well. To resolve this challenge, most features used in the project can be added (easily modified) to be used in other card games. For example, the time limit feature is part of the majority of card games. This is useful as it allows reuse of our system.




For this project, the potential pitfalls will occur during the programming phase. Each person in the group is assigned to a few tasks which he/she will be responsible for. Reasonable deadlines will be made in the first week where everyone needs to hold onto. As already mentioned, tasks are divided however everyone in the group needs to help each other in need. At the end we will have to merge all the functions together to one working system. Everything should then work properly, but it is not easy as imagined. But the group will do their best to prevent things not working together at the end by having weekly meetings with

each other and look at each other's code and make changes if needed. This will not lead to surprises at the end. So communication and working together as a group is the challenge to success for this project.

## Features Author(s): Hao Qin, Björn, Ahsan

### Functional features

ID	Short name	Description	Champion
F1	Game cards	The system shall include and provide for all basic cards (and card-effects) which are included in the original game. All types of cards shall be defined and represented in terms of card name and action (example: the action for skipping the turn when playing a skip card). Normal cards can be defined without an effect.	Hao Qin
F2	Card actions	The system shall perform the action of the game cards that were defined by F1 when a card is played (example: when a skip card is played, the turn is moved to the next player).	Hao Qin
F3	Game set up	The system shall initiate a new game upon startup. The player should first have the option to choose against how many AI players he/she wants to compete. Furthermore, upon startup the system shall create a new draw-pile and discard-pile and deal all players their initial hands.	Björn
F4	Commands	The player can play cards by issuing command-line commands following this syntax: command-name [target-cards]*. The available command-names are the following: - draw: this function draws a card for the current player - play: use the function to play a card in hand - pick: the player can pick another player to perform an action on, or one of the cards from another player - quit: player can leave in the middle of the game	Ashish
F5	Time limit	Each player has a time-limit for each turn, for example 1 minute. The time limit starts counting when the turn begins, and after the time-limit ends the system auto-ends the current player's turn and makes the player draw a card.	Ashish

F6	Game controller	The system shall control the flow and order of the game. Turns shall follow the same structure of playing cards and drawing cards and players' turns shall follow each other clockwise. When an AI player draws an exploding kitten card without a defuse card at hand, it is eliminated from the game. When the human player draws an exploding kitten card without a defuse card at hand, the game ends.	Björn
F7	AI Player Bots	The system shall have a number of AI player bots participate in the game by playing random cards against the human player. The amount of player bots that will participate in a game will be specified in the initial game setup.	Ahsan
F8	Interface	The system should have a functional and appealing interface through which the player can interact with the system. The relevant game aspects, such as the cards in the players hand and the amount of cards in the deck, should be displayed during the game at all times. Also, the player can perform game-actions, such as playing cards, through this command-line based interface.	Ahsan
F9	Player hand manager	The system shall keep track of the cards in hand for each player throughout the game. Cards that are drawn are added to the hand and cards that are played are removed from the hand. The hand will be represented in a practical way so the player can think about which moves to perform next.	Together 
F10	Multiplayer (bonus)	The system should allow human players to play together over the same network. The player should be able to choose multiplayer mode in the initial game setup and wait for other players to join on the same network.	Together

## Quality requirements

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. // specify Implement the game card feature independent, and then we can add more cards with effect just in the game cards feature easily.
QR 3	Overhead time	Responsiveness	All game action (e.g. card effects, drawing a card etc.) should follow each other instantly, without overhead time in between actions.
QR4	Time-limit	Responsiveness	The time limit for each turn should be easily specified by the user
QR5	Interface	Usability	The game should be represented in an appealing and functional way
QR6	Game running control	Reliability	The game shall run consistently under the set up game rule, and follow the players turn order.
QR7*	On-line multiple human players(bonus)	Availability	The host player can invite other human players to join the game on the network.
QR8*	On-line game join password. (bonus)	Security	The players can only join the on-line game when they input the correct password provided by the host player.



## Java libraries Author(s): Ahsan, Hao Qin, Ashish

### Fastjson

We will use fastjson library to convert our Java Objects, from our game set up process, into some JSON representation, which could be very useful for the function realization of



commands functions during the game running. And we can also use this same library to reverse convert a JSON string to an equivalent object.

## Java.util

We will use this library for the timer of each player. We want players to play as fast as possible because otherwise, the game might take too long.

## Java.io

According to the web this library is very useful and contains the classes that handle fundamental input and output operations in Java. The I/O classes in this package can be grouped as follows:

- Classes for reading input from a stream of data.
- Classes for writing output to a stream of data.
- Classes that manipulate files on the local filesystem.
- Classes that handle object serialization.

This system shall use this library to interface with human players using the command line input.

## JavaFX

For the bonus part of this project, if we have enough time, we may build an interface for this card game. We could use this JavaFX library to build this modern graphical user interface(GUI. JavaFX library is an embedded system for use with the JDK and can be used for free. It is very easy to be used to create user interfaces and turn our design into an interactive prototype.

## Time logs

Exploding kittens	3							
Ashish	Activity	Week number	Hours		Björn	Activity	Week number	Hours
	Define functional features	1	2			Define functiona	1	2
	Search Java libraries	1	2			Search Java libr	1	2
	Define quality	1	2			Define quality	1	2
	Define functional features	2	2			Define functiona	2	2
	Search Java libraries	2	2			Search Java libr	2	2
	Define quality	2	2			Define quality	2	2
	Total		12			Total		12
Hao	Activity	Week number	Hours		Muhammad	Activity	Week number	Hours
	Define functional features	1	2			Define functiona	1	2
	Search Java libraries	1	2			Search Java libr	1	2
	Define quality	1	2			Define quality	1	2
	Define functional features	2	2			Define functiona	2	2
	Search Java libraries	2	2			Search Java libr	2	2
	Define quality	2	2			Define quality	2	2
	Total		12			Total		12