Team PJ-b

CODENAMES GAME

Team Members

Name	ID Number
Benjamin Thérien	40034572
Bilal Rana	40013408
Christophe Savard	40017812
Michael Wilgus	29206388
Mordechai Zirkind	27206151
Rezza-Zairan Zaharin	40003377
Shereece Victor	40105094
Steven Zanga	40000797

Contents

Project Analysis and Development Plan

1.1 Introduction

The purpose of this document is to detail the high-level requirements and features of the Codenames Game developed by team PJ-B. Codenames is a game with four roles. These roles can be played by either humans or AIs. This project is an adaptation of the Codenames Game designed by Vlada Chvátil and published by Czech games.

The specifics of how the Codenames Game fulfills these needs will be detailed in the use cases which more will be detailed in the upcoming design phase.

1.2 Purpose

This document will describe the specifications entailed by the development of Codenames in compliance with the requirements of COMP 354. It will outline the high-level requirements encompassing user interfaces, product functions, user descriptions, assumptions and dependencies, constraints, specific requirements, and an analysis model. The analysis model will hold use case diagrams, class diagrams, sequence diagrams, and state transition diagrams.

1.3 Scope

This document only addresses the high level requirements of Codenames that the design phase entails. The analysis model will contain UML diagrams used to serve multiple purposes. The use case diagrams will give an overview of the functions of the project and how users will interact with it. The class diagrams will show the interactions between different objects in the game.

1.4 Definitions and Abbreviations

1.4.1 Definitions

Assassin
The Assassin is a card type that when revealed causes the team that revealed it to lose.

Board

The main playing area will be composed of a 5 by 5 grid of cells with words. Each cell of the grid represents the codename of an agent. Each cell in the grid will be defined in this document as a *Card*.

Card

The card will hold two values and can be in one of two states: hidden or revealed. In its hidden state the card will present a word. In its revealed state the card can either be a blue or red team's *Spy* card, a *Civilian* card or the **Assassin** card.

Civilian
The Civilian is a card type that occupies spaces on the board that not occupied by *Spy* cards or the

Assassin card. When revealed, they cause the revealing team's turn to end.

Model View Controller

The architecture used in the Codenames game, consisting of three individual components, the model, view and controller, which can be developed separately

Operative

An Operative is a team member who uses the hints given by the Spy Master to try and determine where their team's Spies are located. Both the red and blue teams have operatives.

Phase

A phase is a part of a round during which either the *Operative* or *Spy Master* of a particular team do what they're supposed to.

Round

A round is a phase of the game during which each team has a turn to both give a hint and guess. A round is made up of four *Phases*.

Spy

The Spy is a card type that belongs to either the red or blue team. Once all spy cards of a team are revealed, that team wins.

Spy Master

The Spy Master is the player whose job it is to give hints to their team's *Operatives*. Both the red and blue teams have a Spy Master.

1.4.2 Abbreviations

MVC - Model View Controller

1.5 Overview

The rest of this document outlines the problem description and the development plan.

The problem description will describe the game user's interfaces, product functions, user descriptions, assumptions and dependencies, constraints, specification requirements, and the analysis model.

Problem Description

2.1 Project Purpose, Scope, and Objectives

The objective of this project is to simulate a multi-user tabletop game named Codenames game by Vlada Chvátil and published by Czech games. The rules of the game will follow Vlada Chvátil with slight variations to accommodate the digital conversion. This project will be a multi-user game of up to four players. These players can be either humans or AIs.

By working on this project, Team PJ-b can experience the software engineering process and the difficulties that arise when managing a group project.

The project will be worked on over 3 iterative phases: the requirement phase, the design phase, and the test phase.

2.1.1 User interface

As this is the requirement phase, we are keeping the interface to a minimum. With only a single screen that displays the information required for a game. The visual representation of the game is as shown below:

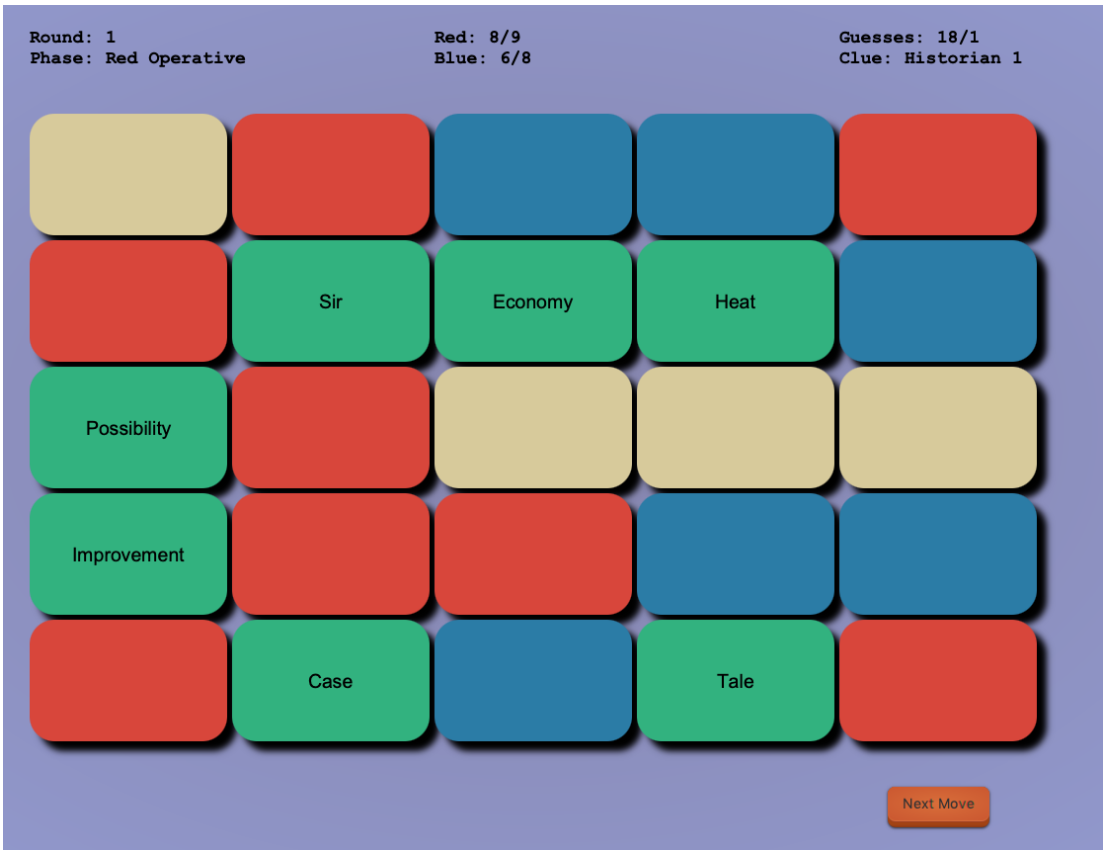

Figure 1: The playable menu of the current game version

Board

The board is represented by a 5 by 5 set of cards, each possessing a word that, when revealed, will change in color to present what is the card type:

- Red is a Red Spy.
- Blue is a Blue Spy.
- Black is the Assassin.
- Beige is a Civilian.

Round

On the top left, the "*Round*" text represents the number of rounds that have passed in the game.

Phase

On the top left, the "*Phase*" text details the current player. This value would be either one of these at a time:

- Blue Spy Master
- Blue Operative
- Red Spy Master
- Red Operative

Score-Keeping

In the middle top, Two sections with the text "*Red*" and "*Blue*" denotes the number of spies revealed for each team to the number of cards left to revealed. This is done in the format of: "Number of spies revealed / Total Number of spies"

Guesses

On the top right, the text "*Guesses*" denotes the number of guesses left to be made by the *Operative* given to by the *Spy Master*. The value is formatted in: "Number of Guesses Left / Maximum Number of Guesses"

Clue

On the top right, below the guesses, the clue given by the *Spy Master* is displayed.

Next Move

On the bottom right is "*Next Move*" button. This is how the human watching the game played by two AIs makes the game progress.

2.2 Constraints

Given that some members of our team use Macs and others use Windows machines, the game must compile and run on both. Due to class requirements we are using the following technologies:

- JAVA as the programming language.
- SQLite as chosen for data storage.
- JUnit for unit testing.
- JavaFX for the GUI.

2.3 Analysis Models

2.3.1 Use Case Diagrams

The following diagrams will help provide an overview of the functions in the game. They describe the action that a player can perform, as well as the interaction between some of the system functions which are not directly controlled by the player.

These use case diagrams are included due to their importance in defining the user-to-software interactions, the requirements, and the scope of the system.

The section below will detail the actors involved in the game:

User

The User represents the human player. For this iteration there is only one User. The User is able to control the chronological flow of the game via the "*Start Game*" use case detailed below.

Spy Master

The Spy Master is involved in the game by doling out clues. As there are only two teams, there will only be two Spymasters in the game. The Spy Master hands out clues via the "*Reveal Clue*" use case detailed below.

Operative

The Operative is involved in the game by revealing cards in accordance to the clue and number of guesses given to by the Spy Master actor. As there are two teams, there will be only two Operatives in the game. The operative reveals cards through the use case "*Reveal CARD*" use case as detailed below.

User Use Case

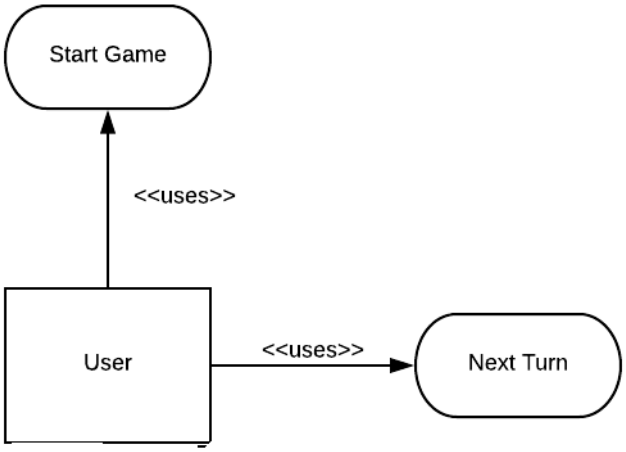

Figure 2: A use case diagram showing the use case concerning the actor "User"

Spy Master and Operative Use Case

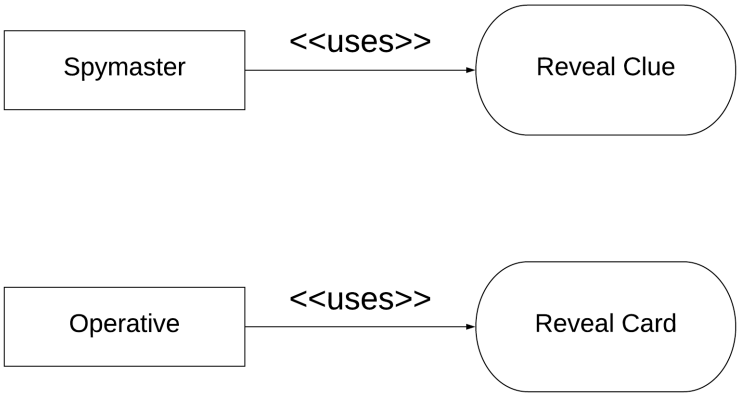

Figure 3: A use case diagram showing the use case concerning the actors "Spy Master" and "Operative"

2.3.2 Use Cases Details

Use Case 1: Start Game

Description	The Player starts the game
Actors	User
Pre-Conditions	None
Basic Path	1. The User clicks "Start Game". 2. 4 Dummy players are generated and each are: • Deposited into a team. • Distributed a role. 3. The board is initialized. 4. The card layout is randomized. 5. The team turn order is randomized. 6. The first turn order is given to the leading team's Spy Master.
Alternative Paths	None
Post-Conditions	• The board has been initialized. • Four AI players are generated with their own team and role. • The first player can play their turn.
Related Use Cases	
Used Use Cases	None
Extending Use Cases	None

Use Case 2: Next Turn

Description	The game moves forward a turn.
Actors	User
Pre-Conditions	• The Board is initialized. • The game has not ended.
Basic Path	1. The user clicks the button, "Next Turn". 2. The current acting player plays their turn. 3. The turn passes to the next acting player.
Alternative Paths	None
Post-Conditions	A player has acted on their turn.
Related Use Cases	
Used Use Cases	None
Extending Use Cases	None

Use Case 3: Reveal Clue

Description	The Spy Master gives a clue.
Actors	Spy Master
Pre-Conditions	• The Board is initialized. • The turn phase belongs to the Spy Master. • The team Operative has not started their turn. • There are still cards on the board to reveal.
Basic Path	1. The Spy Master issues a clue with a number of guesses. 2. The turn is passed to the team's Operative as detailed in the use case, "Reveal Card".
Alternative Paths	Alternative 1: 1. After step 1, if the clue not considered valid by the system (e.g. a word visible on the board contain the clue word), play passed to the alternate team as detailed in the "End Turn" use case.
Post-Conditions	• A clue has been revealed. • The Spy Master's turn has ended. • The turn is passed to the team's Operative.
Related Use Cases	
Used Use Cases	Reveal card, End Team's turn.
Extending Use Cases	None

Use Case 4: Reveal Card

Description	The Operative reveals a card
Actors	Spy Master
Pre-Conditions	• The Board is initialized. • The team's Spy Master has given a clue. • The turn belongs to the Operative. • The team Operative has not started their turn. • There are still cards on the board to reveal. • The number of guesses given by Spy Master is at least one.

Basic Path	1. The Operative chooses to either:
	  • Continue to make guesses, proceed to step 2.
	  • End their turn, proceed to step 4.
	2. The Operative reveals a card on the board.
	3. The number of guesses for the Operative is depleted by one.
	4. If there are still more than one guess left and there are cards to be revealed, repeat step 1.
	5. The turn is passed to the enemy Spy Master detailed in the "Reveal Clue" use case.

Alternative Paths	Alternative 1: • After step 2, if the card revealed is an *Enemy Spy* or *Civilian*, the turn is passed to the enemy Spy Master detailed in the "Reveal clue" use case. • If all of the enemy team's *Spy* cards are revealed, the game ends with the enemy team winning. Alternative 2: • After step 2, If card revealed is the *Assassin* card, the game ends with the enemy team winning. Alternative 3: • After step 2, If all of the Operative's *Spy* cards are revealed, the game ends with the Operative's team winning.
Post-Conditions	• A card is revealed on the board. • The turn is passed to the enemy Spy.
Related Use Cases	
Used Use Cases	Reveal Clue
Extending Use Cases	None

2.3.3 Class Diagrams

Simplified Domain Model

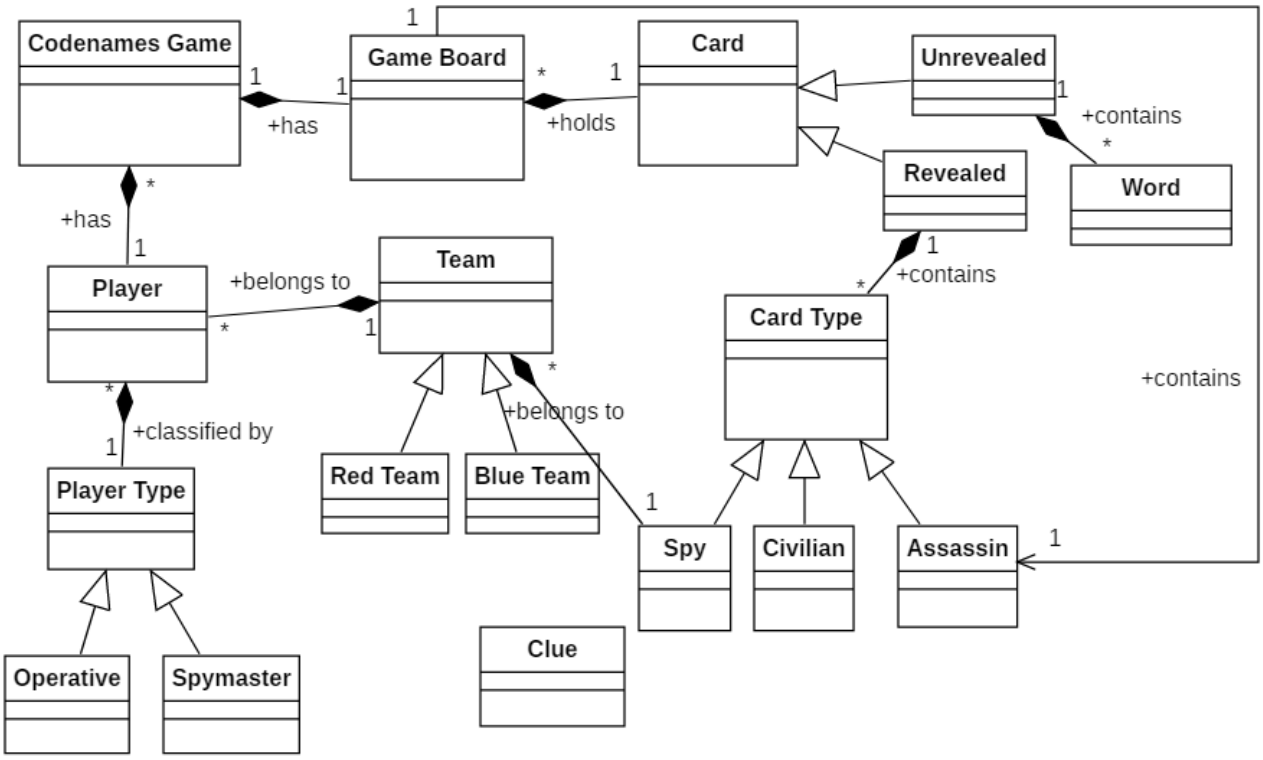

Figure 4: A domain model of the Codenames game.

The Game consists of a game board with cards which can either be revealed or hidden during play. When cards are hidden they show a word. Each card has a type: spy, civilian, the Assassin. When cards are revealed they show their type. There is only one assassin card per game. Each spy belongs to a team, red or blue.

The game also contains players which interact with the game. Each player, like the spies, belong to either the red or blue team.

Class Diagram

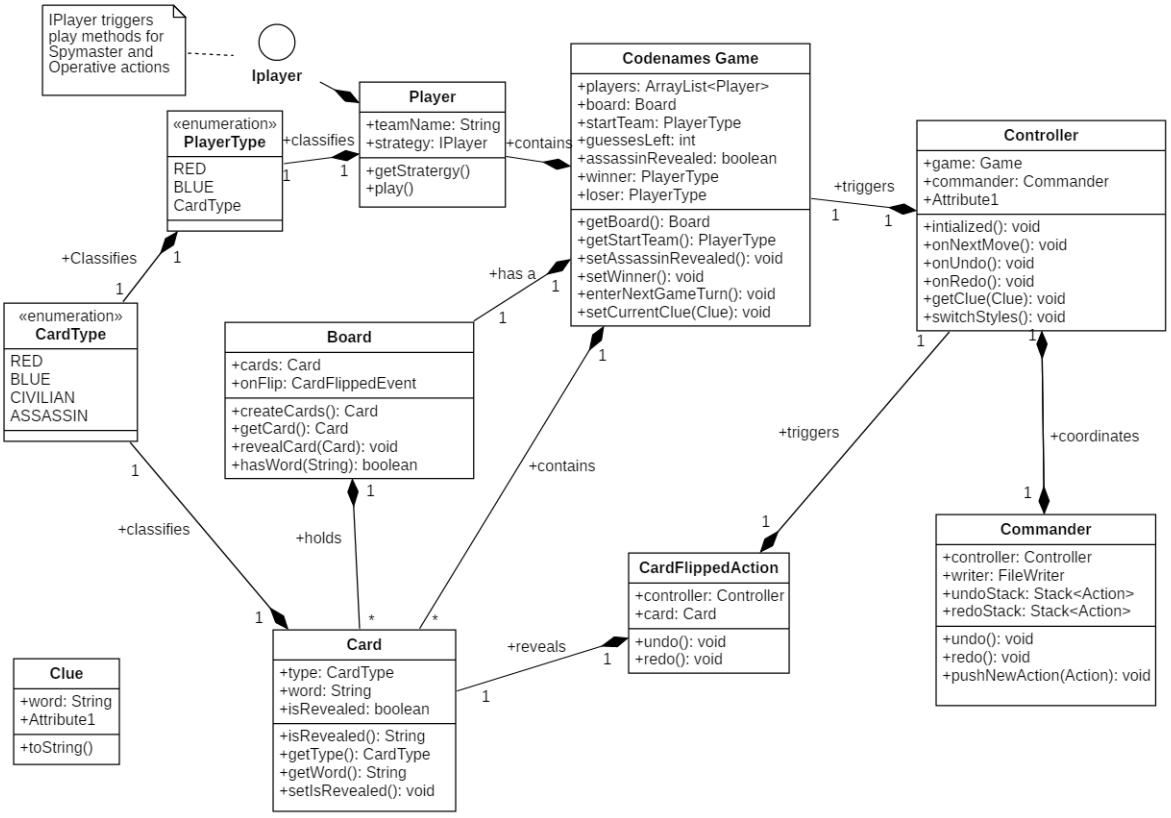

Figure 5: A class diagram of the Codenames game.

This class diagram for the Codenames game does not show all the components of the final classes and interfaces. Instead it shows how some of the concepts presented in the domain model can be implemented.

The commander contains most of the functions the user interacts with. The controller carries out those functions by triggering methods in other parts of the system. The card flipped event for example, can be triggered when the user presses the 'next move' button. This is received by the commander, which passes it to the controller, which triggers the card flipped event, revealing or hiding a card on the game board within the game.