

# anOtherGame

## Requirements and Analysis Document

Version 1.0

2017-05-28

**Miranda Bånnsgård**

**Amanda Jonsson**

**Maja Nyberg**

**Allex Nordgren**

This version overrides all previous versions.

# 1 Introduction

The project aims to create a prototype desktop application of the board game *The Lost Diamond* which is designed by the board game designer Kari Mannerla<sup>1</sup>. To revive this well-selling board game this project will create a Gothenburg edition of the *The Lost Diamond*. In this game, instead of a diamond, the search will be for a kitten. And, for example, instead of ‘move by flight’ there is ‘move by tram’. It will be a standalone and multi-player application.

General characteristics of the application:

- The game is turn-based. When a player has completed his or her turn the turn proceeds to the next player. The order of priority is the same order as the players type in their names on the start pane.
- The game can be canceled, or it will end according to the rules.
- There is no time constraint for a round.
- The application will handle the rules.
- The application does not include a computer-player. It is not possible to play the game alone.
- The application does not save interrupted games or collect any statistics.

## 1.1 Definitions, acronyms and abbreviations

GUI - Graphical User Interface

Java - platform independent language

JRE - Java Runtime Environment. Additional software needed to run a Java application.

Host - a computer where the game will run.

Station - a station is a tram station.

Marker - a marker is located on stations. The player can get money, get robbed or find either a cat or tram-card after flipping a marker.

Round - one complete game ending in a winner or possible canceled.

Turn - the turn of each player. The player can only act during his or her turn.

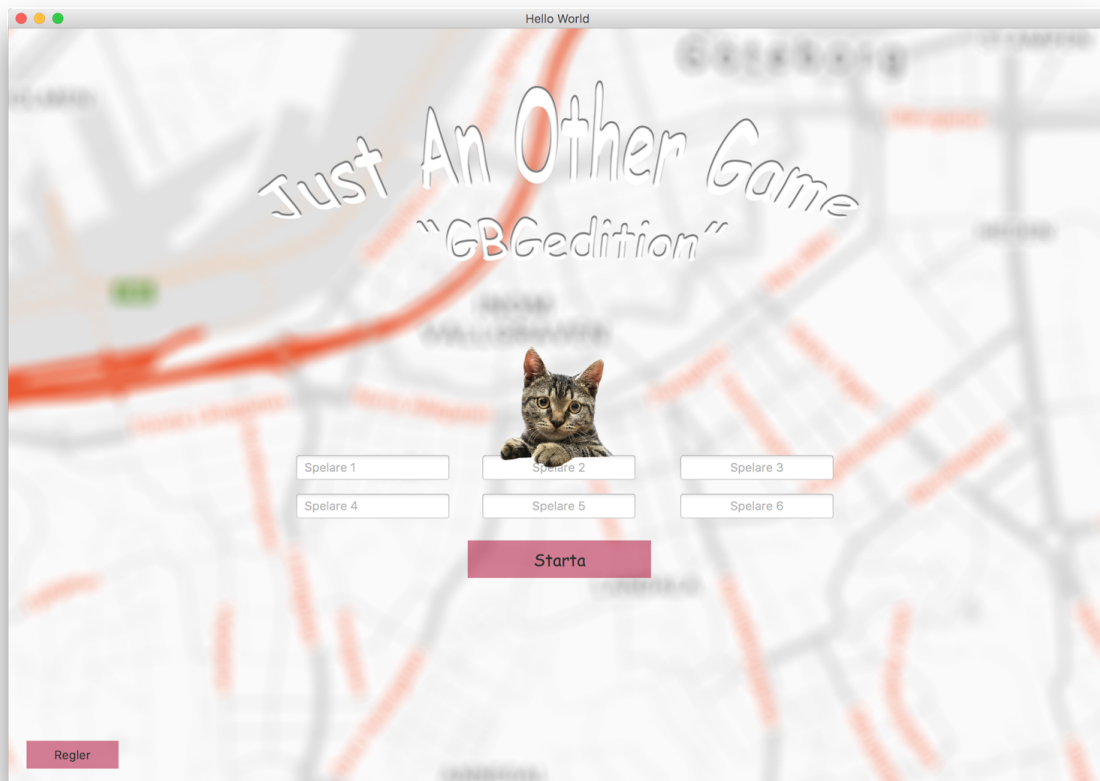
---

<sup>1</sup> Afrikan tähti, Wikipedia, [https://en.wikipedia.org/wiki/Afrikan\\_tähti](https://en.wikipedia.org/wiki/Afrikan_tähti)

## 2 Requirements

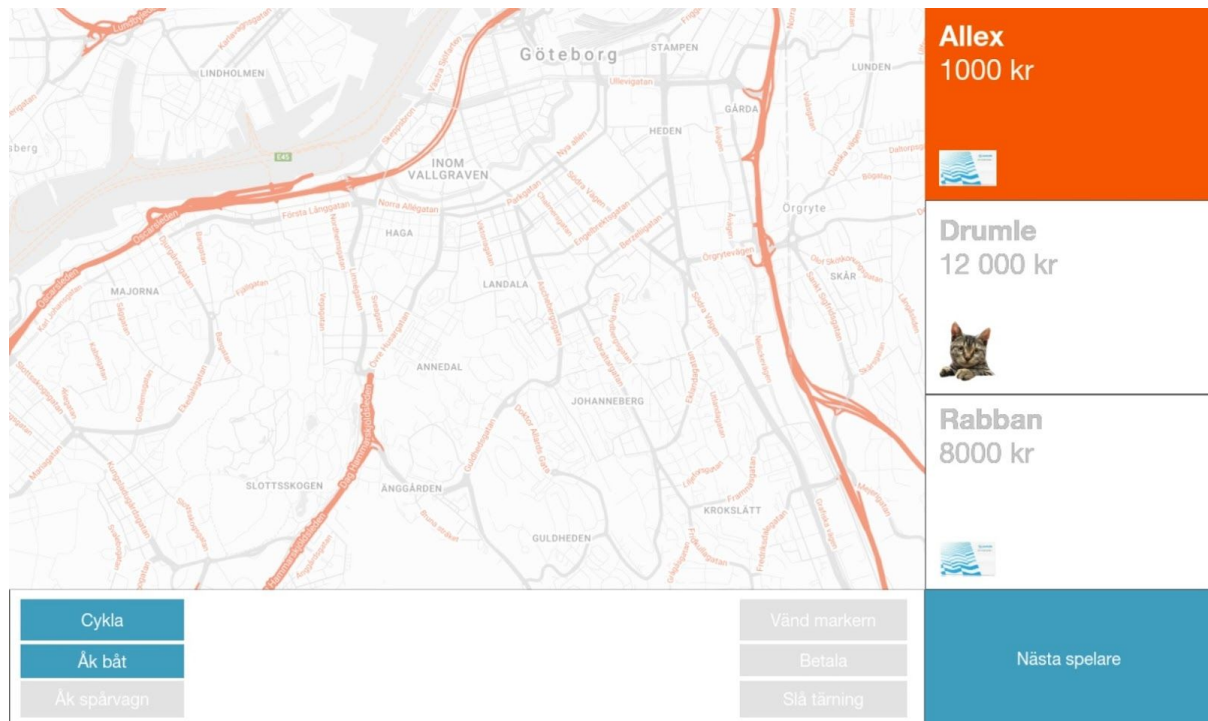
### 2.1 User interface

When the game is started, the first view that appears is the view below. It has text fields for the players to type in their names as well as two button - one for reading the rules and one for starting the game.



*The start view.*

The game view will look similar to the ordinary game. It will have a map with places to go to, specified start location and routes between the places. All players will be visible at all time, together with their balance and if they have a tram card or the cat. The lower pane with buttons will show the player what action are available, as well as specifying whose turn it is.



*The game view.*

## 2.2 Functional requirements

The player should be able to:

1. Select how many players should play the game (between 1 and 6).
2. Start a new game by pressing the "Starta spel" button.
3. Do a turn. During the turn the player should be able to:
  - a. Roll dices. This should trigger a response from the application showing where the player can go on the map.
  - b. Travel by bike or tram.
  - c. Flip markers.
  - d. End the turn.
4. See the 'game over' screen when the round is finished
  - a. From here either start a new game or exit the application.
  - b. See which player won the game.
5. Exit the application.

Ordering of use cases by priority

1. Start game
2. Move by bike
3. Player's turn
4. Flip marker
5. Do marker
6. Win game
7. Game over
8. New game
9. Move by tram

## 2.3 Non-functional requirements

Usability is high priority. The game should be user friendly and easy to understand. Normal users should be able to play the game within a very short period. The player is able to read the rules in the *Rules* screen.

The player should be able to recognize the map and street names from the city Gothenburg.

It should be possible to play a complete round without a crash.

Any actions initiated by the player should not exceed a 2 second response time in worst case. The application's response should take 2 seconds at most.

The game will not be implemented for other platforms than computer. The game should be supported fully by operative systems Windows, Mac and Linux.

The game will be runnable through Gradle.

The application uses images which we have not been given permission to use.

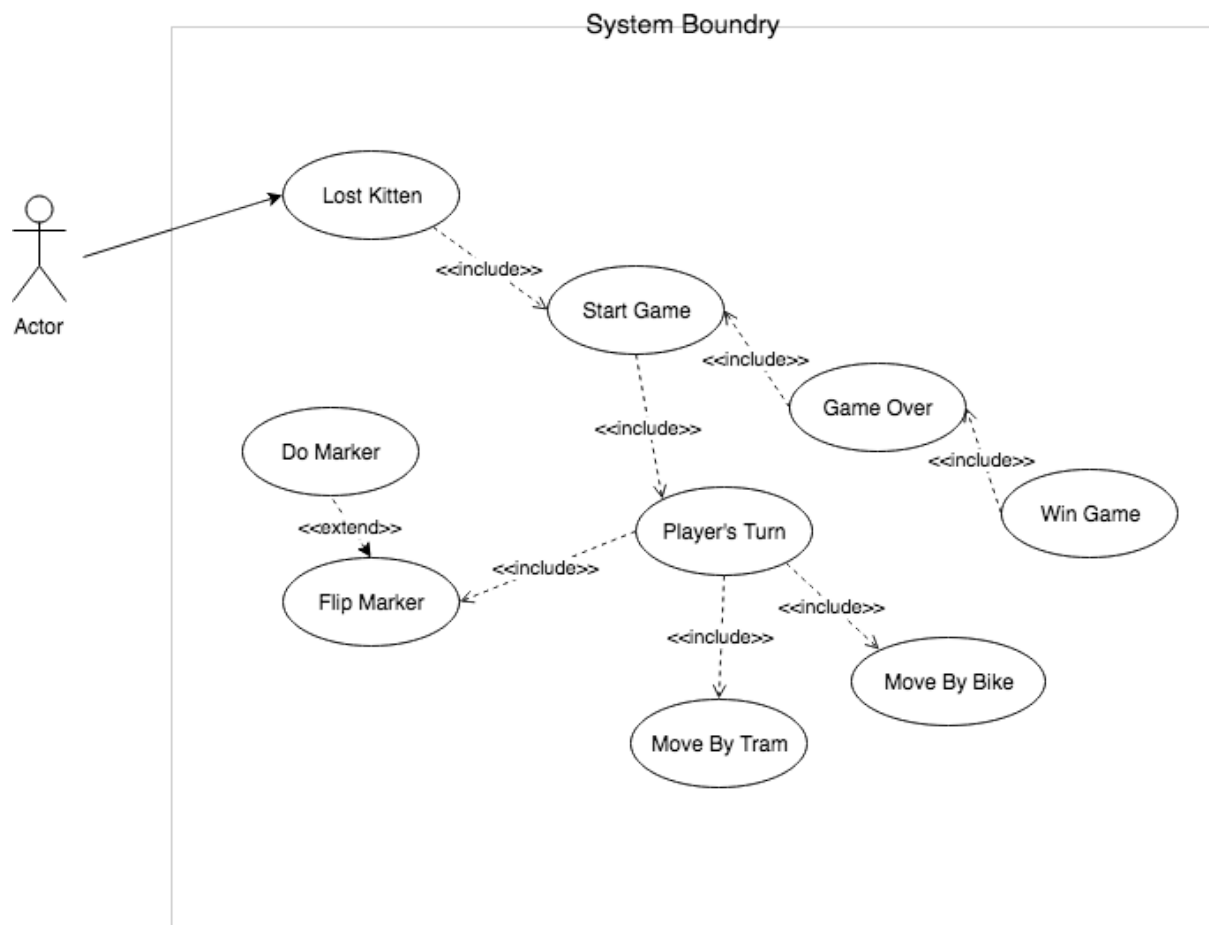
The program will be delivered as a zip-archive containing:

- All needed resources involved in the application.
- A README-file.

To achieve platform independence, the application will use Java environment. All hosts must have JRE installed and configured. The application needs to be installed on all hosts where it will run.

There should be automated tests for each class. The GUI should be tested manually.

### 3 Use cases



## 3.1 Use case listing

### 1. Use Case: Lost Kitten

**Summary:** User opens application Lost Kitten and the start menu is displayed.

**Priority:** High

**Extends:** NA

**Includes:** Start Game

**Participators:** The player

**Normal flow of events:** Player opens the desktop application Lost Kitten.

	User	System
1	Chooses to open the application Lost Kitten.	
2		Opens application and displays the start menu.

## 2. Use Case: Start Game

**Summary:** The start menu is displayed and the player fills out the name of the different players that will participate in a game of Lost Kitten. The player clicks on the “Starta” button on the start menu. A new window with the game board is shown.

**Priority:** High

**Extends:** Lost Kitten

**Includes:** Player’s Turn

**Participants:** The user

**Normal flow of events:** The user starts a game without reading the rules first.

	User	System
1	User writes the names of the players (between 2 and 6).	
2	Clicks on button “Starta”.	
3		Sets up a new window with the game board containing the map of Gothenburg and the number of players chosen. Places markers randomly on the stations, except for the start stations. The chosen number of players are set up with 5000:- each.

**Alternate flow:** The user starts the game after reading the rules.

	User	System
1.1.1	Clicks on the “Regler” button.	
1.1.2		Shows the rules in a pop-up window.
1.1.3	Exits the pop-up with rules by pressing the “Tillbaka till start”-button.	
1.1.4		Closes the pop-up window.

**Alternate flow:** The user starts the game after reading the rules.

	User	System
1.2.1	Clicks on the “Regler” button.	
1.2.2		Shows the rules in a pop-up window.



1.2.2	Exits the pop-up with rules by pressing the outside the pop-up	
1.2.3		Closes the pop-up window.

### 3. Use Case: Player's Turn

**Summary:** This case explains how the user moves by bike, which is the standard move. The cycle path is marked by black lines and black dots.

**Priority:** High

**Extends:** Start Game

**Includes:** Move by Bike, Move By Tram, Flip Marker

**Participants:** The user

**Normal flow of events:** The turn before has just ended. The user moves by bike. This requires that the user's jack is placed on land. It does not cost to move by bike. The player can be placed either on the cycle path or a station.

	Actor	System
1	See UC <i>Move By Bike</i>	
2		Player is on the cycle path, nothing happens. Next player's turn
2.1 Player stays at a station with a marker	See UC Flip Marker	

Alternative flow: The player is on a station and can choose to move by tram.

	Actor	System
1.1.1	See UC Move By Tram	
1.1.2 If player is on a station with a marker	See UC Flip Marker	

Alternative flow: The player is already on a station with a marker (perhaps player did not succeed to flip marker previous turn).

	Actor	System
1.3.1	See UC Flip Marker	

## 4. Use Case: Move by bike

**Summary:** The standard way for a move is by bike, it is also free of charge. The cycle path is marked by black lines and black dots.

**Priority:** high

**Extends:** Player's Turn

**Includes:**

**Participators:** The user

**Normal flow of events:** The user moves by bike. It does not cost to move by bike. The player can move as many step shown by the dice, but the player can also choose to stop earlier if there is a station on the way.

	Actor	System
1	The user clicks on the "Cykla" button	
2		Shows the different path the player can take, by lightning up the positions the player would be placed at.
3	Picks a path by clicking on the position.	
4		Moves the jack to the chosen position.
4.1 Player chooses to stay at the station		Moves the jack to a station, which could be fewer steps away than the number of steps shown by the dice.

## 5. Use Case: Flip Marker

**Summary:** The player has landed/chosen to stop on a tram station with a marker. The player can choose to flip the marker in the same move.

**Priority:** High

**Extends:** Player's Turn

**Includes:** Do Marker

**Participants:** Actual player

**Normal flow of events:** The player chooses to try to get the marker and pays 1000:- for it. Only works if player's balance is over 1000:-. If balance is below 1000:- the player can only roll the dice to flip the marker.

	User	System
1	Chooses to flip the marker on the stop by clicking at "Vänd marker".	
2		Displays different buttons in the "Alternative-window" below.
3	Clicks the "Betala"-button	
4		Updates the player's balance
6		See 'Do Marker'
7		Next player's turn

**Alternative flow:** The player can choose to try to get a 4, 5 or a 6 with the dice instead of paying 1000:-.

	User	System
3.1	Clicks on the "Slå tärning" button.	
3.2		Displays the number, and how successful it was.
3.2.1 If dice shows 4, 5 or a 6		See 'Do Marker'
3.2.2		Next player's turn.
3.2.2 If the dice displays 1, 2 or 3		Next player's turn

Alternative flow: The player lands on a marker but chooses not to pay or roll the dice in order to get the marker.

	User	System
1.1	Chooses not to turn marker	
1.2	Clicks on “Nästa spelare!”	
1.3		Next player’s turn.

## 6. Use Case: Do Marker

**Summary:** The player has flipped a marker and the system performs the action of the marker. There are seven different kind of markers with different actions.

**Priority:** High

**Extends:** Flip Marker

**Includes:**

**Participants:** System only

**Normal flow of events:** The player turns a blank marker.

	User	System
1		Nothing happens.

**Alternative flow:** The player turns a marker with a gem worth 3000:-

	User	System
1.1.1		Displays “Du fick vända markern”.
1.1.2		Updates balance of the player with 3000:-.

**Alternative flow:** The player turns a marker with a gem worth 4000:-

	User	System
1.2.1		Displays “Du fick vända markern”.
1.2.2		Updates balance of the player with 4000:-.

**Alternative flow:** The player turns a marker with a gem worth 5000:-

	User	System
1.3.1		Displays “Du fick vända markern”.
1.3.2		Updates balance of the player with 5000:-.

**Alternative flow:** The player turns a marker with a bandit/pickpocket.

	User	System
1.4.1		Updates the balance to 0.

**Alternative flow:** The player turns a marker with a tram ticket.

	User	System
1.5.1		Displays this ticket in the player’s pane.

**Alternative flow:** The player turns a marker with the kitten!

	User	System
1.6.1		Displays this cat in the player's pane.

## 7. Use Case: Move by tram

**Summary:** This case explains how the user moves by tram, which costs 3000 kr. The tram path is marked by lines between bigger tram stations. To move by tram your jack needs to be placed at a tram station.

**Priority:** high

**Extends:** Player's Turn

**Includes:**

**Participants:** The user

**Normal flow of events:** The user pays 3000 kr to move by tram.

	Actor	System
1	The user clicks on the "Åk spårvagn" button	
2		Shows the different path the player can take, by lightning up the positions the player would be placed at.
3	Picks a path by clicking on the position.	
4		Moves the jack to the chosen position.
5		Updates player's balance.



## 8. Use Case: Win Game

**Summary:** A player collects the kitten and hurry to get home first in order to win before some other player gets a Västtrafik-card and gets home before the first player.

**Priority:** High

**Extends:**

**Includes:** New game, Game over.

**Participators:** The user

**Normal flow of events:** The user collects the cat, wins the game and chooses to play again.

	User	System
1	User collects the cat and goes to either “Redbergsplatsen” or “Lundby” before any other user collects a Västtrafik-card and goes to their starting place.	
2		Show pop-up with the text “Spelare, “namn” vann!”. See UC Game Over.

**Alternate flow:** The user collects the cat, but another user collects a Västtrafik-card, wins the game and chooses to quit game.

	User	System
1.1	A user collects the cat, but another user collects a Västtrafik-card shortly after and goes to either “Lundby” or “Redbergsplatsen” before the user with the cat manage to get there.	
		Show pop-up with the text “Spelare, “namn” vann!”. See UC Game Over.

## 9. Use Case: Game over

**Summary:** A player has just won and can choose to play the game again or finish the application.

**Priority:** High

**Extends:** Win game

**Includes:** Start Game

**Participators:** The user

**Normal flow of events:** The game is finished. The user want to play again.

	Actor	System
1	Clicks button “Spela igen!”.	
		See UC Start Game.

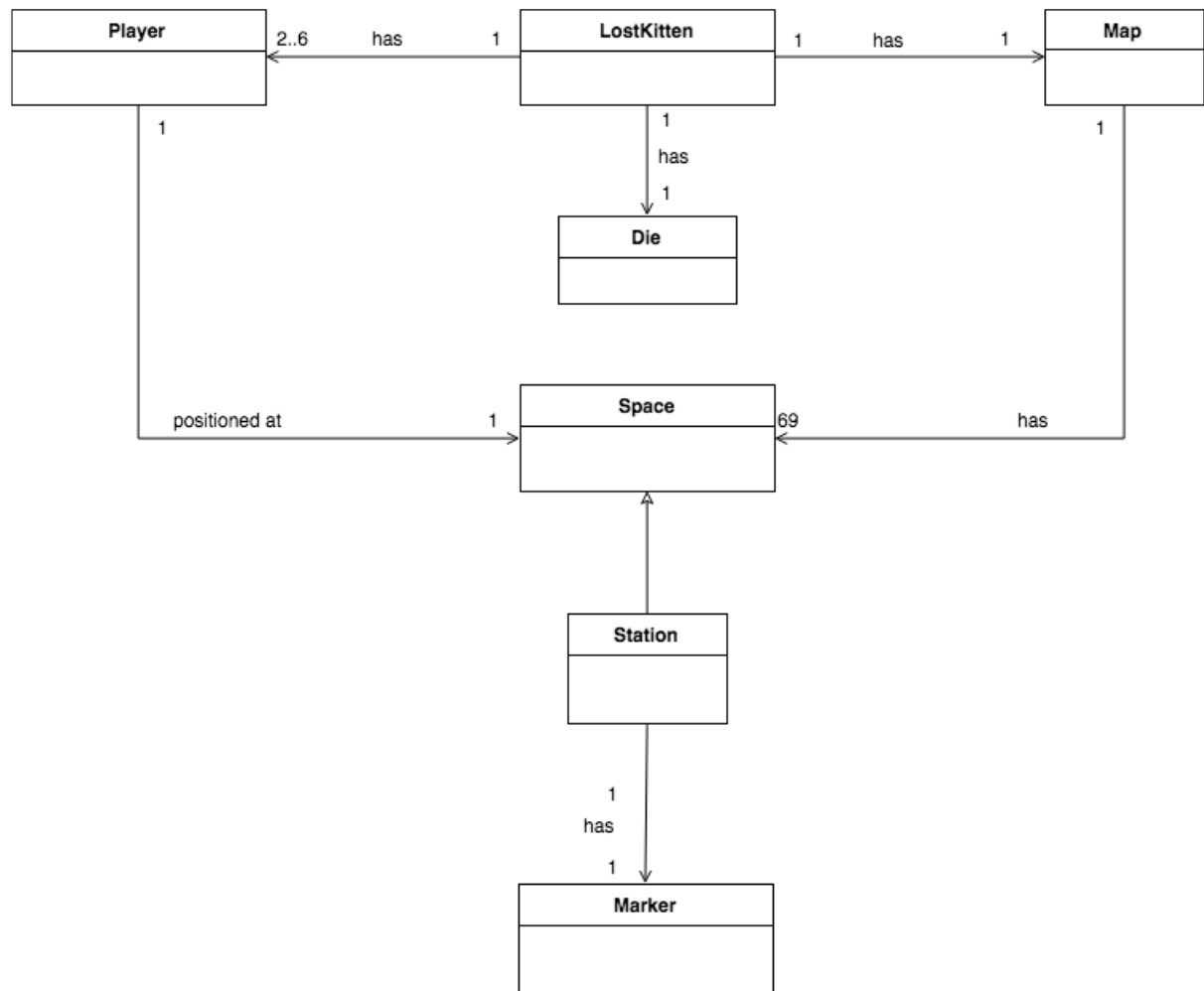
**Alternate flow:** The game is finished. The user wants to quit the application.

	Actor	System
2.1	Clicks button “Avsluta”.	
		Closes application.

**Alternate flow:** The game is finished. The user wants to quit the application.

	Actor	System
2.1	Clicks the red cross-button in the corner.	
		Closes application.

## 4 Domain model



## 4.1 Class responsibilities

### **LostKitten**

- the overall representation of the game

### **Player**

- There can be 2 to 6 players who play the game. A player is always positioned at a space on the map and the player class knows which space the player is positioned at.
- The player class has variables that knows the players budget and if the player has a tram card or have find the lost kitten.

### **Map**

- The map contains a fixed number of spaces and represent the game board.

### **Die**

- Represents a dice, gives a random number between 1 and 6.

### **Space**

- A space represent a location on the board, which may hold a player.

### **Station**

- A station is a special space which contains a marker with some kind of aciton.

### **Marker**

- Contains different actions that can occur for a player.

#### **MoneyMarker**

- Contains gems with a value which will be given to the player if he or she flips a MoneyMarker.

#### **OtherMarker**

- Contain pickpockets, blank, cat or tram card.

## 5 Future work

To extend the project, a high score and an achievements feature could be implemented. The high score could save the names and for example the time it took to finish that round. Achievements could represent different kind of actions that are repeated throughout a round. For example who went by tram the most, who got robbed the most, who flipped most blank markers etcetera.

One feature that could be implemented is the possibility of only one player playing the game, ergo playing towards the computer.

Another feature that could be implemented is the possibility of other ways to travel, for example by boat.

Finally, being able to choose a different map than the one we have hardcoded was something that we discussed. There could be maps over different cities, so the player can choose whether to play TheLostKitten in Gothenburg, Stockholm or for example Malmö.

## 6 References

Afrikan tähti, Wikipedia, [https://en.wikipedia.org/wiki/Afrikan\\_tähti](https://en.wikipedia.org/wiki/Afrikan_tähti), 2017-04-01