01 Meeting 22.04.2020

Anwesend: Erman Zankov, Nikita Smailov, Oliver Corrodi, Tobias Ritscher.
Protokollführer: N. Smailov.

Agenda:

1. Was sollen wir machen?
2. Wie wollen wir vorgehen?
3. Welche Aufgaben gibt es?
4. Wer ist für was verantwortlich?

Ideen zum Vorgehen:

Q1.	Wie Wollen wir vorgehen?	
	O1.	Jeder bekommt Arbeit und denkt selbst nach wie er es implementier. Bsp: GUI, Logik,
		<ablehnt>
	O2.	Klassendiagramm bestimmen -> daraus die Issues abzuleiten.
	O3.	Zuerst Gui-Layout bestimmen-> daraus Anforderungen ableiten
	O4.	Mock-Test schreiben -> Implementation naher.
D1.	Beschlossen wurde das klassische Vorgehen, den Klassendiagrammentwurf(O2), mit dem Gui-Layout Entwurf zu kombinieren. Das "test driven development" (O4) wurde jedem als Personliche Entscheidung überlassen.	

Klassenaufteilung und Klassenbeschreibung:

GameLogic	*Enthält Spiellogk.*
isWinner()	True, falls es gibt ein Gewinner.
nextPlayer()	Gibt den Nächsten aktiven Spieler zurück

GameBoard	*Verwaltet Felder*
GameField[][] board	Das Feld Array.
getFieldOwner()	Gibt den Spieler zurück, der das Feld besitzt.

GameField	*Ein Spielfeld. Abstrakte Superklasse oder Interface*

WorkField	*Ein Spielfeld. Subklasse von GameField.*
int money	das Geld, dass das Feld zurück gibt, wenn man work() betätigt wird.
int turnsToWait	Die Zahl der Züge, die das Spieler warten muss, falls "work()" betätigt wird.
work() :: int	Spieler wird für "turnsToWait" Runde inaktiv, bekommt aber "money" viel Geld zurück.

ModuleField	*Ein Spielfeld. Subklasse von GameField.*
int credits	Anzahl credits die das Feld zurückgibt
int money	
enroll()::int	Feld "kaufen", gibt Credits zurück
upgrade()	Erhöht den Gewinn, dass das Feld zurück gibt, wenn man enroll() betätigt
takeFee()	Es werden "money"-viel CHF werden dem Spieler abgezogen.

Player	*Enthält Spielerspezifische Statistiken eines Spielers.*
int credits	Anzahl Credits, dass der Spieler besitzt.
int money	Das Geld, dass der Spieler besitzt
boolean isHuman	PC oder Mensch am Spielen?
isBroke()	True, falls money == 0
isWorking()	True, falls Spieler ist am arbeiten(inactive) in dieser Runde.

GameController	*Verwaltet den Spielverlauf.*
GameLogic gameLogic	
setPlayerNummer()	
setPlayers()	

Config	*Support-Klasse. Enthält Konstanten und statische Hilfsfunktionen.*
final int MAX_PLAYERS	

Einige Fragen zum Klassendiagramm:

Q1.	Wo wird die Information über die Spieler gespeichert?
D1.	<offen>
Q2.	Was soll GameBoard.getFieldOwner() zurückgeben?
D2.	<offen>

GUI Statistics: Credits/Geld

Q1.	Wie wollen wir die Spiel-Statistiken: Credits, Money, anzeigen?	
	O1	Über einem Buttonklick "Show statistics" wird sie sichtbar.
		<abgelehnt> Information ist Teil des Spieles, somit muss immer sichtbar sein
	O2.	Für jeden Spieler separat gezeichnet.
		<abgelehnt>
	O3.	Gemeinsame Statistik auf einem Bar.
		<akzeptiert>

		Wie im klassischen Spiel hat man steht Zugriff zu den Spiel-Informationen.
D1.	Beschlossen wurde eine globale Statistik über aller Spielern auf einem Bar zu platzieren.	

Q2.	Wo wollen wir "Fundbox" positionieren?	
	O1.	Im unteren Fenster Teil.
D2.	<offen>	

Q3	Wie soll man Statistik für 2 Spieler im Gegensatz zu 4 Spielern darstellen?	
	O1	Es wird die 1. und 3. Spalte benutzt.
D3	<offen>	

Interaktion mit GUI:

Q1.	Welche Buttons sind notwendig für die Interaktion mit dem Spiel?
A1.	Spieler interagiert folgendermassen mit Modulen: • enroll () • work () • ask for help() - Stipendium beeintragen • upgrade ()
Q2.	Wo werden die Interaktionsknöpfe angezeigt?
D2.	Auf einem separaten Event-Field.

Fragen zu den Gui-Feldern:

Q1.	# Feldern, die es geben soll?
A1.	Spielfeld wir als ein Raster mit 11x11 Feldern implementiert. Dabei werden nur die äusseren Felder benutzt.
Q2.	Welche Felder-Layout wollen wir haben?
D2.	<offen>

Klassenaufteilung:

Oliver Corrodi	Game-UI;
Tobias Ritscher	GameController;
Nikita Smailov	GameField;GameBoard;
Erman Zankov	GameLogik; Player;

Todo's:

1. Uns auf den definitiven Spiel-Verlauf eignen.
2. Spiel durchspielen.
3. Game-Board-Layout entwerfen.

Abkürzungen:

Q#.	Frage Nummer #
A#.	Antwort auf die Frage #

D#.	Entscheidung
O#.	Option Nummer #
<abgelehnt>	Wurde abgelehnt
<akzeptiert>	Wurden angenommen
<offen>	Ist noch offen.