

Projektnavn: CDIO Final M2

Gruppenummer: 18

Afleveringsfrist: *Mandag den 11/01 2017 Kl. 09:00*

Fag og retning: Diplom Softwareteknologi

Denne rapport er afleveret via Campusnet (der skrives ikke under).

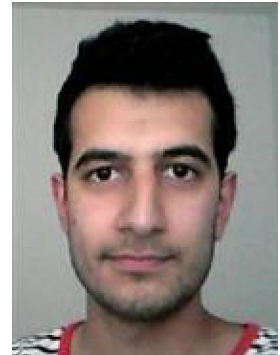
Denne rapport indeholder **XX** sider inklusiv denne.



Kasper Leiszner
s165218



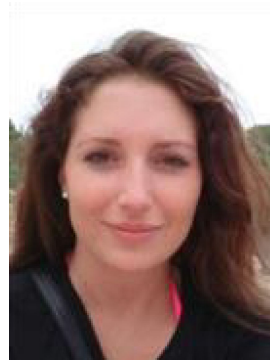
Frederik Von
Scholten s145005



Bijan Negari
s144261



Troels Lund
s161791



Helene Zgaya
s104745

Timeregnskab

Dato*	Deltager	Design & Analyse	Impl.	Test	Dok.	Andet	Ialt
02-01-2017	Troels					3	3
02-01-2017	Kasper	1				2	3
02-01-2017	Helene					3	3
02-01-2017	Bijan		1			2	3
02-01-2017	Frederik					3	3
03-01-2017	Troels	3				2	5
03-01-2017	Kasper	1			1	3	5
03-01-2017	Frederik	3				2	5
03-01-2017	Bijan	3				2	5
03-01-2017	Helene	3				2	5
03-01-2017	Troels	2	2	1			5
04-01-2017	Frederik					2	2
05-01-2017	Helene		5				5
06-01-2017	Helene	4	3				7
05-01-2017	Troels		4				4
05-01-2017	Frederik		5				5
06-01-2017	Troels	1	3			1	5
04-01-2017	Kasper		2			1	3
05-01-2017	Kasper		3				3
06-01-2017	Kasper		2			1	3
06-01-2017	Frederik		5				5
08-01-2017	Kasper		1				1
09-01-2017	Kasper	1	2			1	4
09-01-2017	Frederik		5				5
10-01-2017	Kasper			4		1	5

Punktforms oplæg

Opsummering

Vi har indtil videre opnået at lave et matador spil, som fungerer, med meget få fejl. Dog er det uden alle features og regler. Det vil sige at vi fremover vi indføre nye features, og derefter sikre at det ikke skaber nye fejl. Således vi altid vil have et spil med få fejl.

Features som er implementert:

- Spil bræt med 40 fuld funktionelle felter
- 2 til 6 spillere
- Ekstra slag, når to ens slås
- Prototype af chance kort
- Fuldt funktionelt fængsels felt
 - Betale sig ud
 - Slå sig ud
 - Bruge chance kort til at komme ud
- Et rafflebære med to terninger
- Ejer en spiller en felt gruppe i samme farve, får han ekstra afgift
- GUI
- At vinde spillet
- At gå bankerot
- Et banksystem til at give og trække penge fra spillere

Features som vi regner med at indføre de sidste par dage:

- Huse og Hoteller
- At sælge grunde
- Pantsætning af grunde
- Udbyggelse på chancekort
- Debugging

Det vil sige at vi i den sidste tid, gerne vil nå at lave så meget så muligt, så vi får et spil der ligner det originale matador spil mest muligt. Dog vægter vi højest at have en velfungerende udgave af spillet, om det så vil sige at vi skal hive nogle funktioner ud igen. I grov træk vil det sige at vi vil nå så langt som muligt, men uden at det vil påvirke mængden af spillets fejl.

```

classDiagram
    package controller {
        class GameController {
            +player: Player
            +cup: Dicecup
            +out: Out
            +endGame: boolean
            +moveController: MoveController
            +main() String! : void
            +GameController() : GameController
            +GameController() Out : GameController
            +setup() : void
            +rollGame() : void
            +turn() Player : void
            +printAccon() Player : void
            +addPlayer() Player : void
            +resetOwnedFields() Player : void
            +getBankrupt() Player : void
            +winner() : void
            +getPlayerArray() Player : void
            +setPlayer() Player : void
        }
        class MoveController {
            +movePlayer() int, Player : void
        }
    }

    package model {
        class BankAccount {
            +balance: int
            +BankAccount() : BankAccount
            +addSum() int : boolean
            +withdraw() int : boolean
        }
        class Player {
            +name: String
            +account: BankAccount
            +bankingStatus: boolean
            +playerPos: int
            +breweryCount: int
            +fieldCount: int
            +jailCards: ArrayList<FreeJailCard>
            +Player() String : void
            +getJailCards() ArrayList<FreeJailCard> : void
            +setJailCards() ArrayList<FreeJailCard> : void
        }
        class Die {
            +value: int
            +roll() : void
        }
        class Dicecup {
            +die1: Die
            +die2: Die
            +roll() : void
        }
    }

    package model.fields {
        class Fieldlist {
            +fields: Field[]
            +FieldNames() String[] : Fieldlist
            +FieldDescription() String[] : Fieldlist
            +FieldSet() int : Fieldlist
        }
        class Field {
            +name: String
            +description: String
            +out: Out
            +Field() String, String, Out : void
            +landOn() Player, Out : void
            +rollString() String : void
            +getValue() int : void
        }
        class Ownable {
            +price: int
            +owner: Player
            +wantToBuy: boolean
            +Ownable() String, String, int, Out : void
        }
        class Chance {
            +Chance() String, String, Out : void
        }
        class GoToPrison {
            +GoToPrison() String, String, Out : void
        }
        class Tax {
            +taxAmount: int
            +taxRate: int
            +Tax() String, String, int, Out : void
        }
        class Parking {
            +bonus : void
        }
        class Plot {
            +groupNumber: int
            +roll: int
            +Plot() String, String, int, int, Out : void
        }
        class Brewery {
            +baseRent: int
            +Brewery() String, String, int, int : void
        }
        class Fleet {
            +baseRent: int
            +Fleet() String, String, int, int, Out : void
        }
    }

    package model.cards {
        class Card {
            +description: String
            +roll: Out
            +Card() String, String, Out : void
            +doCard() Player, Out : void
        }
        class BalanceCard {
            +BalanceCard() String, int, Out : void
        }
        class MoveFleetCard {
            +MoveFleetCard() String, Out : void
        }
        class MoveFreeCard {
            +MoveFreeCard() String, Out : void
        }
        class FreeJailCard {
            +FreeJailCard() String, Out : void
        }
        class FamilyCard {
            +FamilyCard() String, int, Out : void
        }
        class Gift {
            +Gift() String, int, Out : void
        }
        class Deck {
            +cardsCount: int
            +cards: Card[]
            +roll: Out
            +des: String[]
            +Deck() Out : void
            +shuffleArray() void : void
            +rollCards() void : void
        }
    }

    package view {
        class Out {
            +Output : void
            +FakeOutput : void
        }
        class Language : void
    }

    GameController --> MoveController
    GameController --> BankAccount
    GameController --> Player
    GameController --> Die
    GameController --> Dicecup
    GameController --> Fieldlist
    GameController --> Field
    GameController --> Ownable
    GameController --> Chance
    GameController --> GoToPrison
    GameController --> Tax
    GameController --> Parking
    GameController --> Plot
    GameController --> Brewery
    GameController --> Fleet
    GameController --> Card
    GameController --> BalanceCard
    GameController --> MoveFleetCard
    GameController --> MoveFreeCard
    GameController --> FreeJailCard
    GameController --> FamilyCard
    GameController --> Gift
    GameController --> Deck
    Out --> Output
    Out --> FakeOutput
    Language --> Out
  
```