UNIVERSITEIT
GENT

# PROJECT MONOPOLY

Softwareontwikkeling

benjamin fraeyman

# Table of Contents

# Uitwerking Console

## Deel 1: structuur

### Overzicht Klassen

- Player
- Dice
- Properties
- Card
- Commands

### Klasse Player

Eigenschappen:

- String Name (naam speler)
- Int Money (geld die speler bezit)
- Int X (locatie speler)
- List<Properties> properties (velden in spelers bezit)
- (List<ChanceCard> ChanceCards / List<CommunityCard> CommunityCards)

Methodes:

- Boughtloc:
  Hiermee ken je een huis toe aan de speler, je trekt ook de kost van het huis van de speler af.
- Addmoney:
  Wordt gebruikt om geld toe te voegen of af te trekken van de speler.
- Move:
  Verplaatst de speler.
- Add(kaarttype):
  Nog te implementeren, het is de bedoeling dat de speler kaarten kan bijhouden zoals “niet naar gevangenis”.

### Klasse Dice

Eigenschappen:

- Int Zijden

Methodes:

- ThrowDice:
  Werpt de dobbelsteen.

# Klasse Properties

## *Basisklasse – Properties*

Eigenschappen:

- String Name
- String Street

Methodes:

- Display

## *Afgeleide klasse – Start*

Met deze klasse af te leiden kan ik makkelijk checken of een speler langs start komt.

## *Afgeleide klasse – Utility*

Deze klasse is niet volledig uitgewerkt door tijdsgebrek. Functies die deze klasse bevat is het aantal keer werpen om de huur te bepalen en de check of de speler beide velden in zijn bezit heeft.

## *Afgeleide klasse – RailWay*

Deze klasse is niet volledig uitgewerkt door tijdsgebrek. Functies die deze klasse bevat is de check hoeveel de speler van deze velden in zijn bezit heeft.

## *Afgeleide klasse – House*

Eigenschappen:

- Int Value (kost van het veld)
- Int BaseRent (basisrent van het veld)
- Int Houses (aantal huizen op het veld)
- String Owner (eigenaar van het veld)

Methodes:

- AddHouse:
  Een extra huisje bouwen op het veld.
- SetOwner:
  Een eigenaar toekennen aan dit veld.

## Klasse Card

### Opmerking:

Deze klasse werkt niet helemaal daar ik er te veel wou insteken.
Het was mijn bedoeling om de speler:

- Willekeurig te kunnen zetten
- Naar de gevangenis te laten gaan
- Geld te laten ontvangen of betalen
- …

Maar het probleem was een onderscheid te kunnen maken tussen deze dingen zonder dat de klasse te ingewikkeld werd. Nu dat ik de oplossingen van het labo heb gezien, heb ik door dat ik misschien beter op een andere manier deze klasse had aangepakt.

De eigenschappen en methoden die volgen zijn deze die ik er zou ingestoken hebben.

### Eigenschappen:

- Bool ToJail?
- Int Money (kan + of – zijn)
- Int PlaceAt (ga naar die locatie)
- Bool FreeJail?
- …

### Methodes:

- GoJail:
  Verplaats de speler naar jail zonder langs start te gaan.
- GiveMoney:
  Geef de speler het bedrag (uit de bank of van andere spelers).
- Place:
  Verplaats de speler naar x, laat hem huur betalen of laat het hem kopen of laat hem bouwen.
- …

## Interface ICommands

Bevat enkel de methode Run().

Deze commandos bevatten de werking van het spel. Hieronder volgt een overzicht van alle commandos.

Afgeleide commandos:

- AddHouseCommand
  Met dit command voeg je huizen toe aan het veld
- BuyPropertyCommand
  Hiermee koop je een veld
- LoadGameCommand
  Inladen van het spel
- NewGameCommand
  Opstarten nieuw spel
- PayRentCommand
  Huur betalen
- SaveGameCommand
  Spel opslaan

# Deel 2: Werking

## Opstarten:

### *Nieuw spel:*

Door NewGameCommand te laten lopen doe je onderstaande stappen:

1) Dobbelstenen aanmaken met custom zijdes
2) Namen aan spelers toekennen
3) Velden inladen
4) Dit opslaan in een savefile
   (Door SaveGameCommand te laten lopen)

### *Continue:*

Door LoadGameCommand te laten lopen laad je alles in van de savefile.

Volgende acties worden doorlopen:

0) Dobbelsteen wordt geworpen
1) Speler wordt verplaatst
2) Er wordt gecheckt of het veld koopbaar is
    a. Ja: keuze tussen kopen of niet kopen
    b. Nee: volgende check
3) Er wordt gecheckt of het veld al een eigenaar heeft
    a. Ja: check of de persoon die op het veld eigenaar is
        i. Ja: keuze tussen een huis bouwen of niet.
        ii. Nee: huur betalen aan de eigenaar
    b. Nee: beurt aan de volgende speler

### *Quit:*

Spel wordt opgeslagen en afgesloten.

# Uitwerking GUI

## Deel 1: Structuur

### Overzicht Klassen

- Zelfde klassen als bij console

## Deel 2: Werking

Bijna hetzelfde als in console.

Het grote verschil is dat er in plaats van input naar console verwezen wordt er hier een nieuwe form opent die je om input vraagt.

Open form: LaunchWindow, hier heb je volgende keuzes:

*Nieuw spel:*

Open form: NewGame:

1) Aanpassen van dobbelsteen
2) Namen aan spelers toekennen
3) Automatische opslag van gegevens

*Continue:*

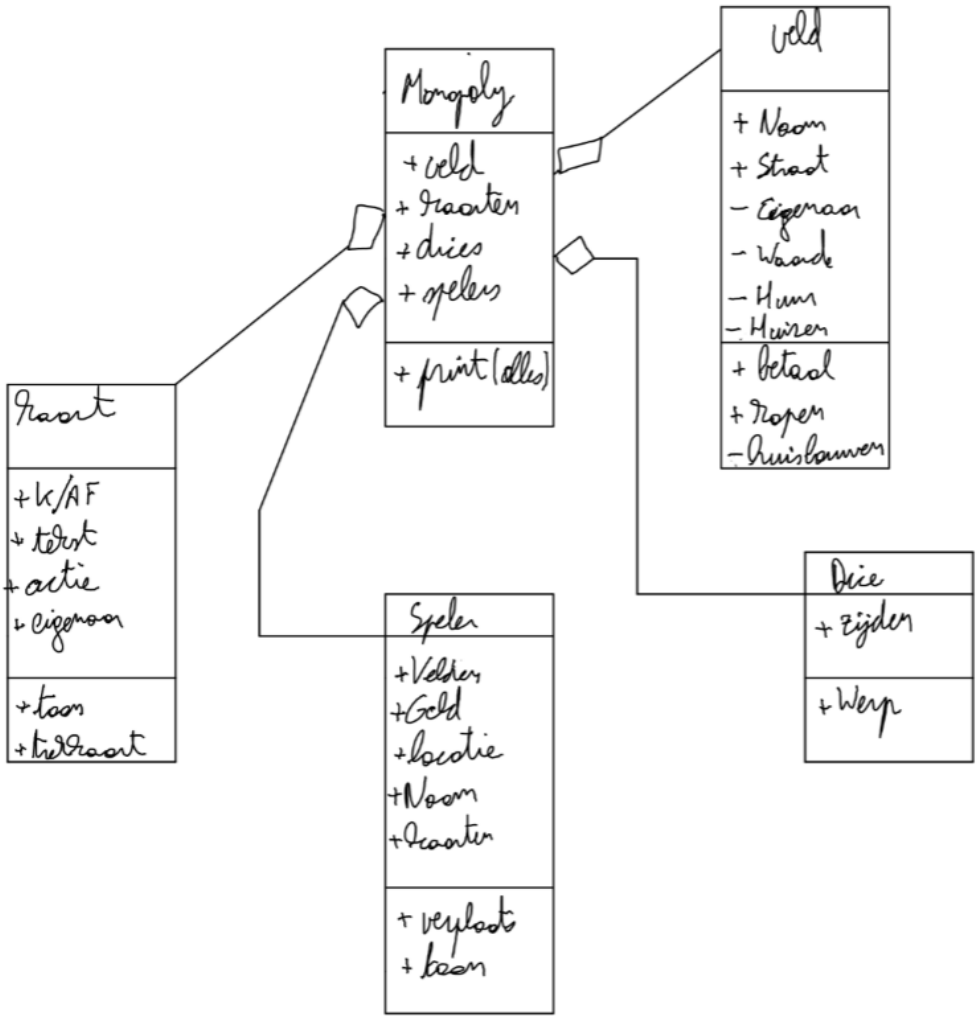
Open forms: Bord

1) Data van de huidige speler wordt getoont
2) Knop “Werp” laat de speler werpen en verplaatst hem
3) Er wordt gecheckt of het veld koopbaar is
    a. Ja: open form: BuyHouse, waar je kan kiezen om het huis te kopen.
    b. Nee: volgende check
4) Er wordt gecheckt of het veld al een eigenaar heeft
    a. Ja: check of de persoon die op het veld eigenaar is
        i. Ja: open form: AddHouseBuilding, keuze tussen een huis bouwen of niet.
        ii. Nee: huur wordt betaalt aan de eigenaar
    b. Nee: beurt aan de volgende speler

*Quit:*

Spel wordt opgeslagen en afgesloten.

# UML-schema


Veld
+ Naam
+ Straat
- Eigenaar
- Waarde
- Huur
- Huizen
+ betaal
+ kopen
- huisbouwen
Monopoly
+ veld
+ kaarten
+ dices
+ spelers
+ print(alles)
Kaart
+ K/AF
+ tekst
+ actie
+ eigenaar
+ toon
+ trekkaart
Speler
+ Velden
+ Geld
+ locatie
+ Naam
+ kaarten
+ verplaats
+ koop
Dice
+ zijden
+ Werp