CDIO del 2 – 02312-14 indledende programmering og 02313 udviklingsmetoder i IT-systemer

Projekt navn: **Projekt 2**

Gruppe nr: **13**

Afleveringsfrist: **Mandag d. 11/11 2013 kl. 5:00**

Denne rapport er afleveret via Campusnet (Der skrives ikke under)

Denne rapport indeholder 25 sider inkl. denne side.

Studie nr, Efternavn, Fornavne underskrift

s133988, Malte Magnussen _____

Kontakt person (Projektleder)

s134219, Leonardo Samari _____

s133974, Nicolai Hansen

s124278, Aleksandar Vlacic

s133980, Jiahua Liang

s132091, Ole Kalsbøll

Timeregnskab

CDIO 2							
Time-regnskab	Ver. 2013-09-07						
Dato	Deltager	Kode	Rapport	Test	Diagram	Andet	Ialt
2013-10-29	Aleksandar		2				2
2013-10-29	Ole				2		2
2013-10-29	Leonardo		2				2
2013-10-29	Nicolai				2		2
2013-11-01	Malte		1		1		2
2013-11-01	Aleksandar		2				2
2013-11-01	Ole	1			1		2
2013-11-01	Jiahua	1			1		2
2013-11-01	Nicolai	1			1		2
2013-11-01	Leonardo				2		2
2013-11-05	Malte	1	1				2
2013-11-05	Aleksandar	1			1		2
2013-11-05	Ole	2					2
2013-11-05	Jiahua	2					2
2013-11-05	Nicolai	2					2
2013-11-07	Jiahua	2					2
2013-11-05	Leonardo	1			1		2
2013-11-08	Malte	2	1				3
2013-11-08	Aleksandar		2				2
2013-11-08	Ole	3					3
2013-11-08	Jiahua	2					2
2013-11-08	Nicolai			2			2
2013-11-08	Leonardo		2				2
2013-11-09	Ole				2		
2013-11-09	Malte		1		1		2
2013-11-10	Leonardo				2		2
2013-11-10	Malte		1		1		2
2013-11-10	Nicolai		2				2
	Sum	21	17	2	18	0	56

Indholdsfortegnelse

Introduktion

Vores kunde har givet os en opgave, som går på at udvikle et nyt spil til dem. Dette nye spil er et brætspil, hvori terningerne fra det forrige spil også bliver brugt. Spillet er mellem 2 spillere, og der spilles med 2 terninger. Man kan lande på felter mellem 2-12 afhængigt af det antal øjne man slår med terningerne, og at lande på et felt kan enten medføre positiv, neutral eller negativ virkning på spillerens beholdning. Den første spiller som opnår en beholdning på 3000 har vundet.

Vi vil i denne rapport gøre vores bedste for at opfylde kundens krav og udvikle et godt spil. Måden vi tænker at gøre dette på, er først og fremmest ved at lave nogle diagrammer for at danne os et overblik, før vi skal kode. Vi kommer også til at bruge kode fra vores tidligere spil, for at kunne danne grundlaget for vores kodning i dette nye spil.

Kravspecificering

Vi har en række funktionelle og non-funktionelle krav til vores nye spil.

De funktionelle krav lyder som følgende:

- Spillet skal kun kunne spilles mellem 2 personer.
- Når spilleren slår, skal man lande på et felt med numrene fra 2-12.
- Når man lander på et felt, skal en tekst udskrives omhandlende det aktuelle felt.
- Hvert et felt har en specifik effekt på spillerens beholdning og spillets gang.
- Hver spiller skal starte med en beholdning på 1000.
- Et vundet spil skal opnås, ved at en spiller opnår en beholdning på 3000.

Og de non-funktionelle krav lyder:

- Spillet skal kunne spilles på DTU's databarer uden bemærkelsesværdige forsinkelser.
- Spillet skal være sprogligt simpelt, så det let kan oversættes til andre sprog.
- Det skal være let at implementere nye terninger ind i spillet.
- Man skal kunne bruge spillerne og deres beholdninger i andre spil.

Udvikling af programmet

Før vi starter med at kode, gennemgår vi de krav vores kunde har givet os, og så opstiller vi diagrammer udfra vores navneordsanalyse. Vi har opstillet et case diagram for vores spiller og udvikler - vi har udvikler med da vi vil lave en test af vores spil senere.

Use-case diagram

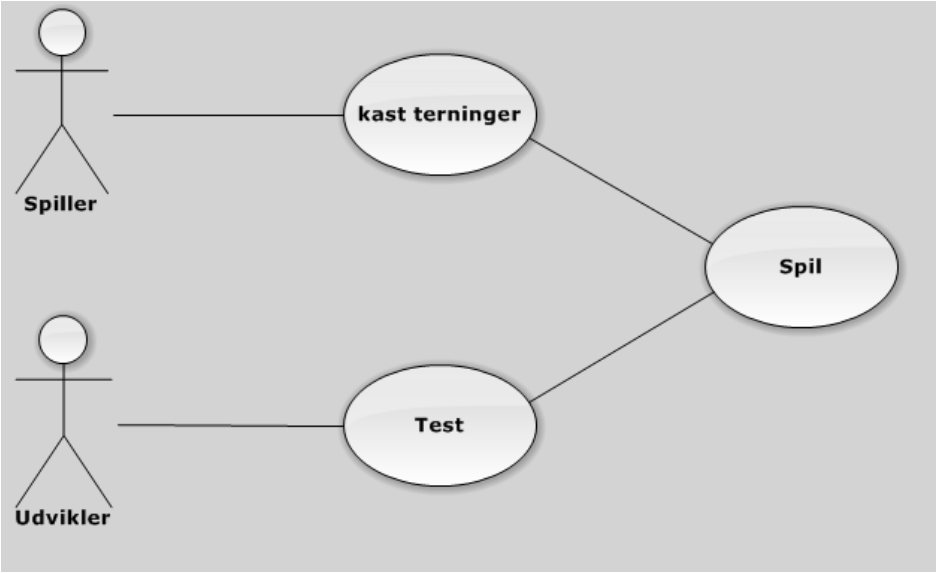

Diagram 1 Use Case, lavet med Software Ideas Modeler

Udfra dette use case diagram har vi opstillet vores use case beskrivelser.

Spil

1. Spiller indtaster spiller navn 1 og spiller navn 2
2. Spiller vælger at spille
3. Systemet "kaster" 2 terninger
4. Systemet rykker spiller til feldt svarende til terning kast
5. Systemet tildeler spiller point

Punkt 2-5 gentages indtil en spiller har opnået 3000 point.

udvikler

1. Udvikler indtaster spiller 1 og spiller 2
2. Udvikler lander på the wearwall og tjekker om spilleren får et ekstra slag
3. Udvikler tester om vundet spil kan opnåes
4. Udvikler tester om man kan kommer under 0 i point

Punkt 2 testes ved at sætte terningerne til at slå 10.

Punkt 3 testes ved at gennemfører spillet.

Navneordsanalyse

Ord	Modelkomponent	Ordklasse
Spil	Klasse	Substantiv
Terning	Klasse	Substantiv
Spiller	Klasse	Substantiv
Raflebæger (DiceCup)	Klasse	Substantiv
Konto	Klasse	Substantiv
Indsæt (navne)	Operatør	Verbum
Kast med terninger (rollDice)	Operatør	Verbum
Sum/return Sum	Operatør	Verbum
Udskriv tekst (println)	Operatør	Verbum
Effekt (positiv/negativ)		
Feltliste	Array	Substantiv

Som man kan se udfra vores navneordsanalyse har vi valgt navne på vores klasser, operatører og vores array. Vi har dog valgt at give dem engelske navne i programmet, idet vi mener alle og enhver skal have muligheden for at rette i det, også selvom persononen ikke kan dansk.
At vi har valgt at give vores ting engelske navne, giver også en international forståelse for koden.

Domænemodel

Vi har nu vores navne til vores klasser og kan derfor opstille en domænemodel så vi ved hvad de forskellige klasser skal kunne.

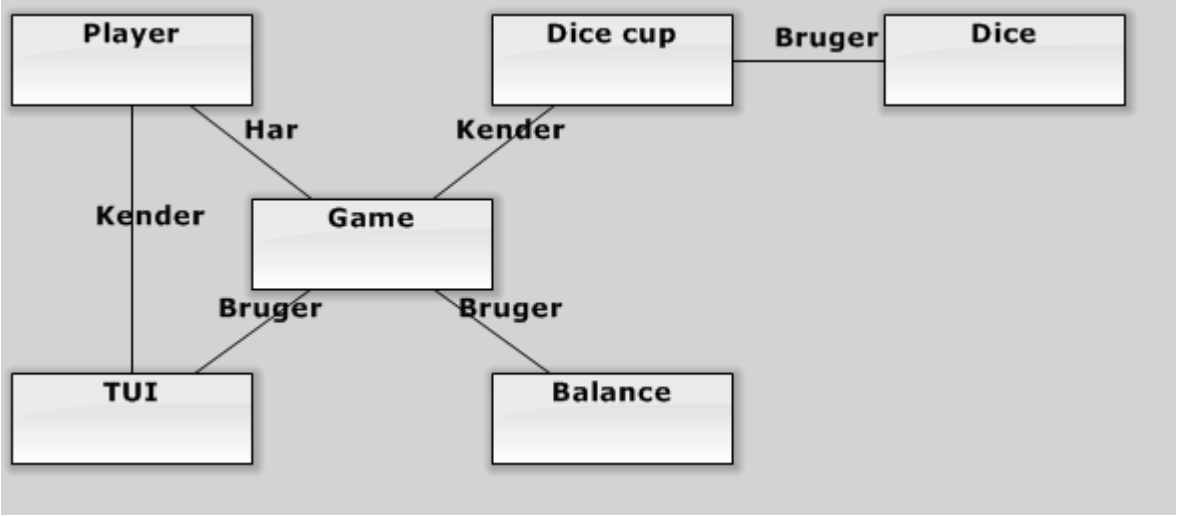

Diagram 2 Domænemodel, lavet med Software Ideas Modeler

Vi har opstillet vores domænemodel, sådan så game har associationer til alle vores klasser med undtagelse til Dice, da den får terning kastende igennem Dice cup. Grunden til at Player og TUI kender hinanden, er at vi gemmer navne direkte ind i player når TUI spørg efter dem.

BCE-model

For at danne et overblik over vores klasser, med andet end kun vores domænemodel, har vi også lavet en BCE-model.

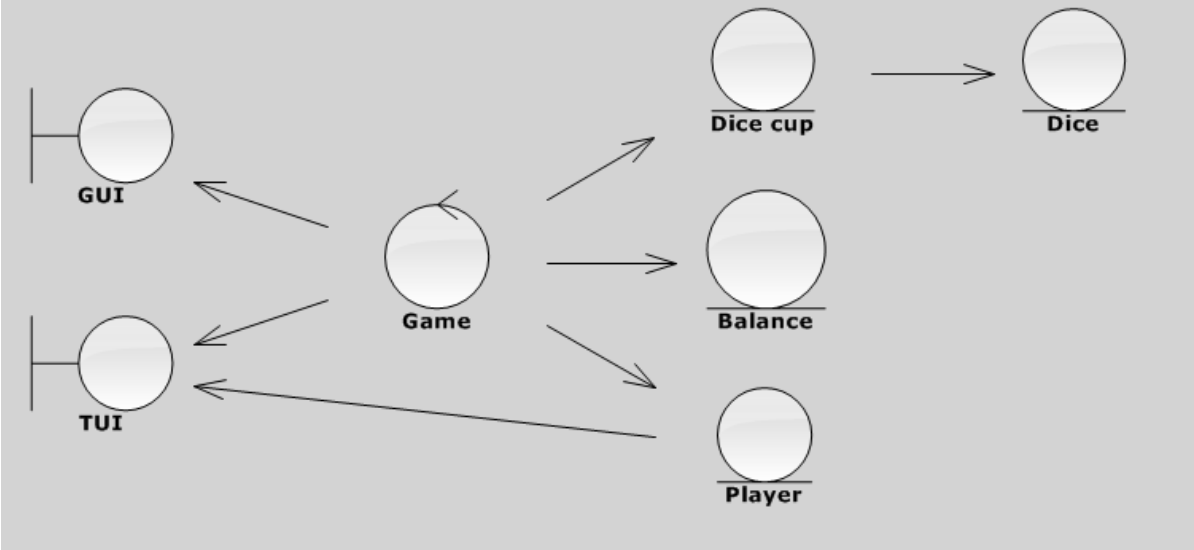

Diagram 3 BCE-model, lavet med Software Ideas Modeler

Vi har vores to boundries, GUI og TUI, de bliver brugt hhv. til dette grafiske output til brugen, og det tekst basseret output til bruger. Vi har kun en controller, Game, da vi mener at de andre klasser har så små funktioner at vi hellere vil benytte dem som entities.

Sekvensdiagram

Vi har også benyttet os at et sekvensdiagram, for at se hvordan spillerene kommer igennem spillet.

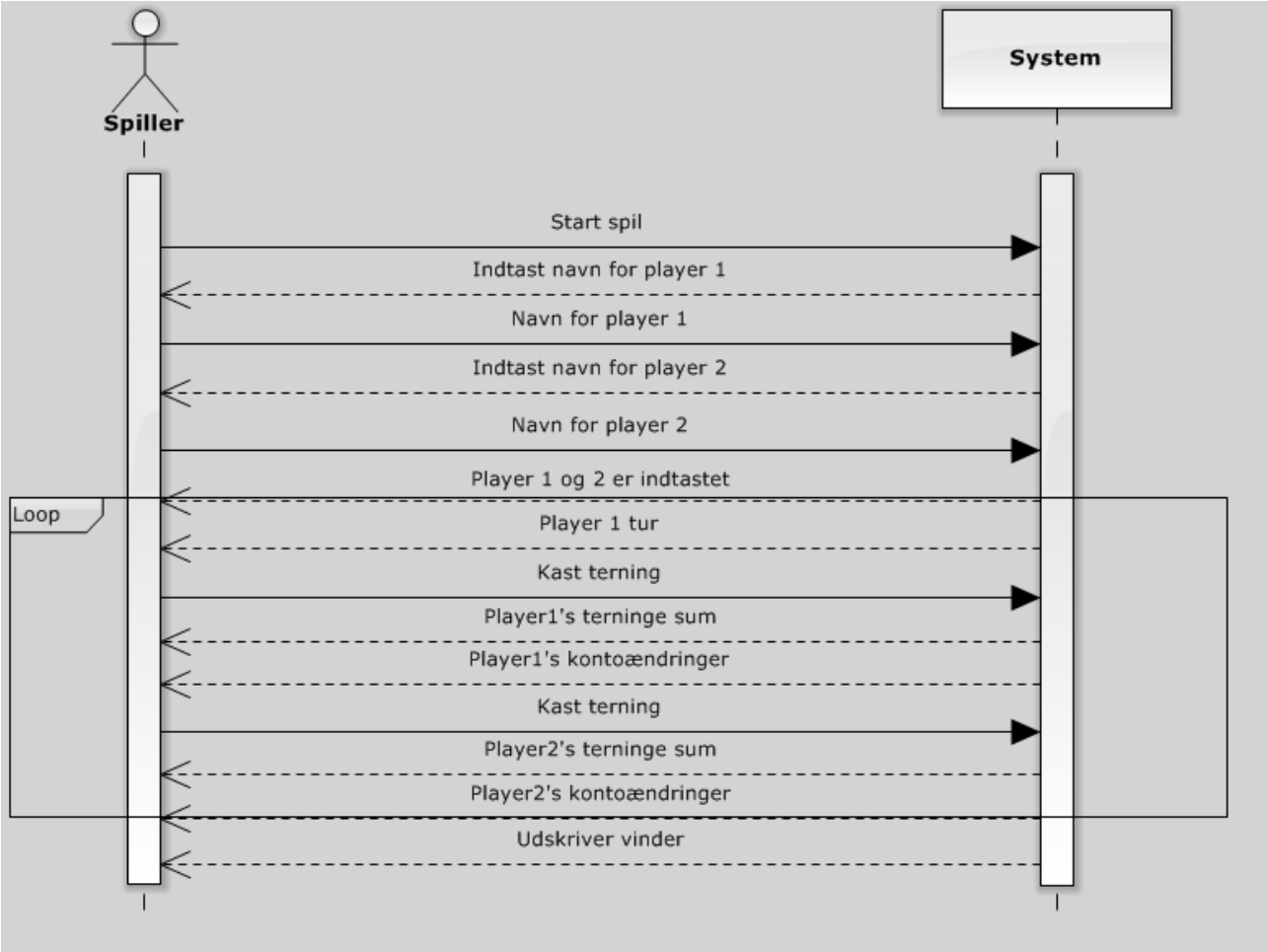

Diagram 4 Sekvensdiagram, lavet med Software Ideas Modeler

Benyttelse af GRASP

Vi har benyttet os af GRASP, som vi kan se på vores klasser. Vi har 6 forskellige klasser, hvoraf det kun er game klassen, der kender til de andre klasser(undtagen dice klassen, der kun kender til dicecup). Det betyder altså, at det hele bliver samlet i game klassen, og det er let at ændre i koden, da man derfor ikke skal ind og ændre i 5 forskellige klasser, men kun 1 eller evt. 2 hvis noget skal laves om, og det er derfor også let at genbruge evt. player klassen eller dice klassen til andre spil. Eksempler herunder:

Ekspert:

Vores klasser består af de nødvendige informationer der skal til for at udføre hvert enkelte handling de er programmeret til. F.eks. er vores dice-klasse kodet til at oprette en enkelt terning, hvor vores dicecup-klasse bruger terningen to gange (dvs. får to terninger) og ryster derefter begge terninger.

Skaber:

Her samles alle vores klasser, der tilsammen udgører vores spil. Vi har en dicecup klasse, player klasse, balance klasse og en TUI klasse, hvor alle klasser bliver ”samlet” i game klassen, og man kan derfor spille spillet.

Lav kobling:

Her har vi sørget for at de forskellige klasser kun kender til game klassen. Dette gør, at det er let at udskifte f.eks. player klassen eller terning klassen, eller benytter dem i et helt andet spil.

Desgin domænemodel

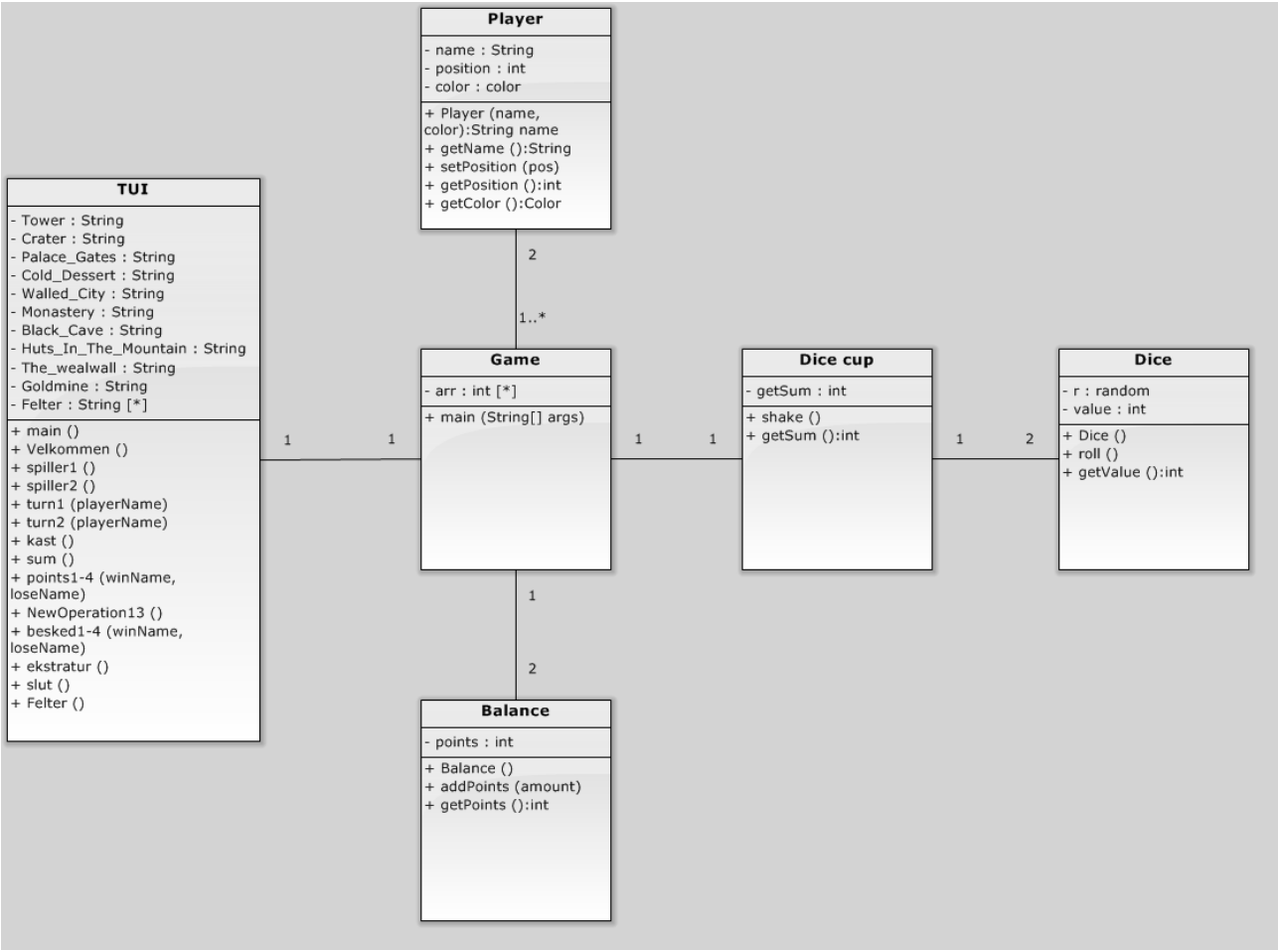

Diagram 5 Desing domænemodel, lavet med Software Ideas Modeler

Som man kan se udfra vores design domænemodel, har vi valgt at player og TUI ikke kender hinaden alligvel. Alt går igennem Game klassen, og derfor har vi valgt at player 1s og player 2's navne også bliver gemt igennem Game og så videre til player.

Desgin sekvensdiagram

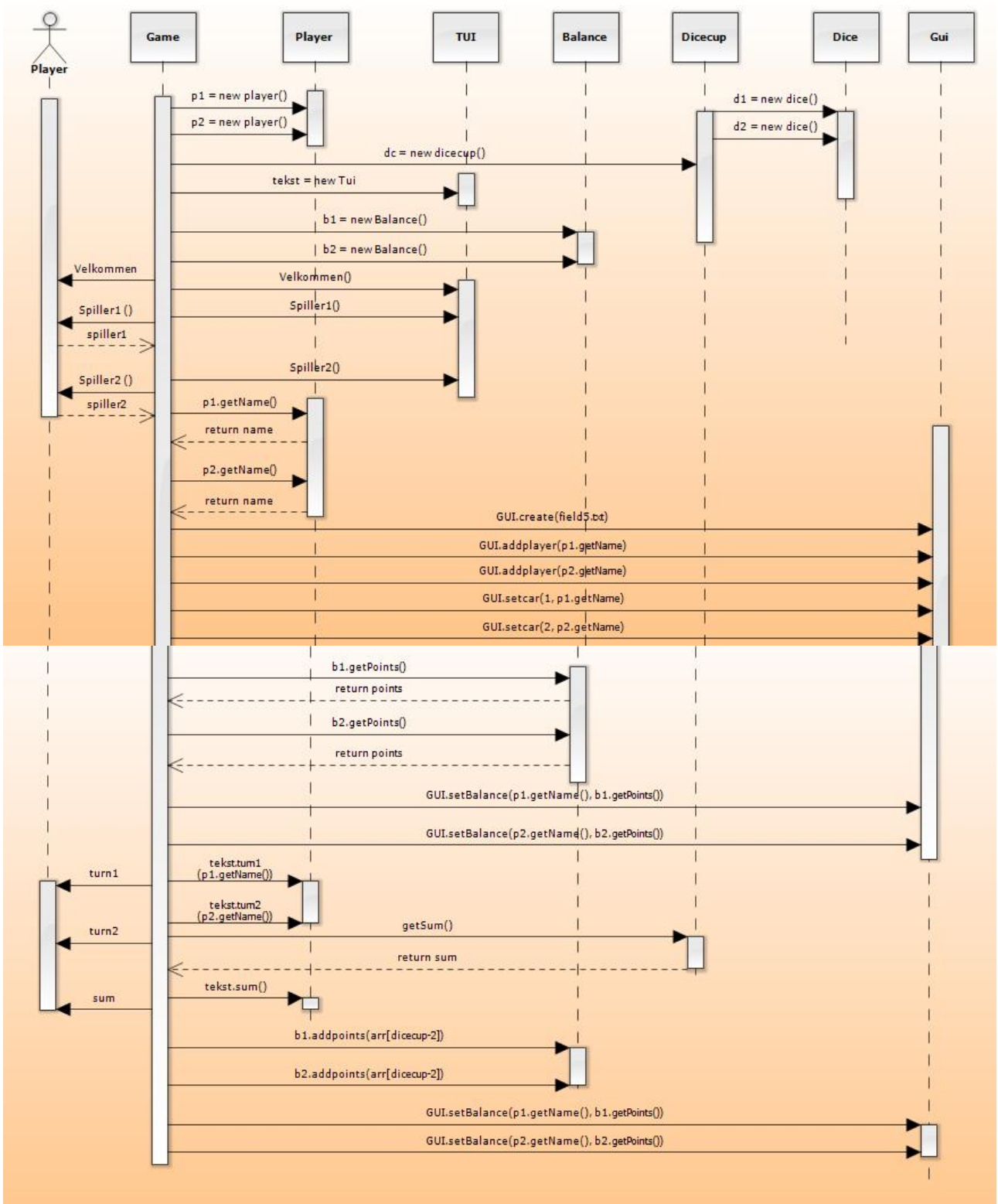

Diagram 6 Design sekvensystem diagram, lavet med Software Ideas Modeler, billed taget med Windows Klippeværktøj

Test

Vi har testet vores spil, ved at teste om de to spilleres kontoer kan komme i minus. Vi har testet det ved at lave en ny class der hedder test, hvor vi har en spiller der slår med to terninger. Testen er simpel og går egentlig bare ud på at vise spillet gang, på en måde der er nem at overskue. Spilleren slår med terningerne indtil der er opnået en beholdning på 0 eller derunder og afslutter derefter spillet. Testen gør det nemt at følge med i hvert enkelt slag og virkningen heraf (dette kan ses i bilag 2 – kode for test). Kører man testen flere gange opdager man hurtigt at spillet altid afsluttes så snart beholdningen når 0 eller derunder. Billedet nedenfor viser testen når den er blevet kørt igennem:

```
Start score: 1000
Terning sum: 7  Point: 0        Samlet score: 1000
Terning sum: 2  Point: 250      Samlet score: 1250
Terning sum: 7  Point: 0        Samlet score: 1250
Terning sum: 10 Point: -80      Samlet score: 1170
Terning sum: 11 Point: -90      Samlet score: 1080
Terning sum: 4  Point: -100     Samlet score: 980
Terning sum: 6  Point: 180      Samlet score: 1160
Terning sum: 8  Point: -70      Samlet score: 1090
Terning sum: 3  Point: -200     Samlet score: 890
Terning sum: 8  Point: -70      Samlet score: 820
Terning sum: 4  Point: -100     Samlet score: 720
Terning sum: 5  Point: -20      Samlet score: 700
Terning sum: 11 Point: -90      Samlet score: 610
Terning sum: 8  Point: -70      Samlet score: 540
Terning sum: 9  Point: -60      Samlet score: 480
Terning sum: 10 Point: -80      Samlet score: 400
Terning sum: 9  Point: -60      Samlet score: 340
Terning sum: 10 Point: -80      Samlet score: 260
Terning sum: 5  Point: -20      Samlet score: 240
Terning sum: 11 Point: -90      Samlet score: 150
Terning sum: 8  Point: -70      Samlet score: 80
Terning sum: 7  Point: 0        Samlet score: 80
Terning sum: 8  Point: -70      Samlet score: 10
Terning sum: 10 Point: -80      Samlet score: -70
Spillet er tabt
```

Billed 1 test, billed taget med Windows Klippeværktøj

Her kan man se at spillet er tabt når samlet score går i minus.

Bilag 1 – kode af spil

Game class

Vi blev nød til at tage billeder af klassen Game, da denne ikke blev køn da vi bare kopirede den ind direkte fra kode, til word. Alle billederne er taget med Windows Klippeværktøj.

```
package Spil;

import java.awt.Color;

public class Game {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        Player p1, p2; // de to spillere
        int turn = 0; // holder styr på hvis tur det er
        Dicecup dc = new Dicecup();
        Tui tekst = new Tui();
        String command;
        Balance b1 = new Balance();
        Balance b2 = new Balance();


        int[] arr = new int[] { 250, -200, -100, -20, 180, 0, -70, -60, -80, -90, 650 };

        // velkommens tekst
        tekst.velkommen();
        tekst.spiller1();
        p1 = new Player(scan.next(), Color.PINK); // opretter en ny spiller med det indtastede navn og farve 1
        tekst.spiller2();
        p2 = new Player(scan.next(), Color.YELLOW); // opretter endnu en ny spiller med det indtastede navn og farve 2

        GUI.create("fields5.txt");
        GUI.addPlayer(p1.getName(), 0, p1.getColor()); // tilføjer spiller 1 til pladen
        GUI.addPlayer(p2.getName(), 0, p2.getColor()); // tilføjer spiller 2 til pladen
        GUI.setCar(1, p1.getName()); // sætter spiller 1 position til 1
        GUI.setCar(1, p2.getName()); // og spiller 2 til 1

        GUI.setBalance(p1.getName(), b1.getPoints());
        GUI.setBalance(p2.getName(), b2.getPoints());
```

```
boolean gameRunning = true;

while (gameRunning) {

    boolean ekstratur = true;

    while (ekstratur) {
        System.out.println();

        // Printer hvis tur det er.
        if (turn == 0)
            tekst.turn1(p1.getName());
        else if (turn == 1)
            tekst.turn2(p2.getName());

        command = "";

        // Bruger indtaster "k" for kaste terningen
        while (!command.equals("k")) {
            tekst.kast();
            command = scan.next();
        }

        dc.shake();

        // Printer terningerne i konsollen
        tekst.sum();

        System.out.println();

  // Tilføj point til den der kastede terninger
  if (turn == 0) {
      b1.addPoints(arr[Dicecup.getSum - 2]);
      p1.setPosition(Dicecup.getSum);
      tekst.Felter();
  } else {
      b2.addPoints(arr[Dicecup.getSum - 2]);
      p2.setPosition(Dicecup.getSum);
      tekst.Felter();
  }

  // Opdater point på spillepladen
  GUI.setBalance(p1.getName(), b1.getPoints());
  GUI.setBalance(p2.getName(), b2.getPoints());
```

```
    // Hvis en af spillerne har vundet eller tabt, stop spillet og print besked
    if (b1.getPoints() >= 3000) {
        tekst.points1(p1.getName(), p2.getName());
        GUI.setCar(12, p1.getName());
        tekst.besked1(p1.getName(), p2.getName());
        gameRunning = false;
    } else if (b2.getPoints() >= 3000) {
        tekst.points2(p2.getName(), p1.getName());
        GUI.setCar(12, p2.getName());
        tekst.besked2(p2.getName(), p1.getName());
        gameRunning = false;
    } else if (b1.getPoints() <= 0) {
        tekst.points3(p2.getName(), p1.getName());
        GUI.setCar(1, p1.getName());
        tekst.besked3(p2.getName(), p1.getName());
        gameRunning = false;
    } else if (b2.getPoints() <= 0) {
        tekst.points4(p1.getName(), p2.getName());
        GUI.setCar(1, p2.getName());
        tekst.besked4(p1.getName(), p2.getName());
        gameRunning = false;
    }
```

```
                // Opdater spillerne position på spillepladen
                if (turn == 0) {
                    GUI.removeAllCars(p1.getName());
                    GUI.setCar(p1.getPosition(), p1.getName());
                } else if (turn == 1) {
                    GUI.removeAllCars(p2.getName());
                    GUI.setCar(p2.getPosition(), p2.getName());
                }

                if (Dicecup.getSum != 10)
                    ekstratur = false;
                else
                    tekst.ekstratur();
            }


            // Skift spilleren tur
            if (turn == 0) {
                turn = 1;
            } else if (turn == 1) {
                turn = 0;
            }
        }

        // Print slutresultatet i konsollen
        tekst.slut();

        scan.close();
        System.exit(0);
    }

}
```

Balance class

```
package Spil;

public class Balance {
		private int points;

		public Balance() {
				points = 1000;
		}

		// tilføjer point til spilleren
		public void addPoints(int amount)

		{
				points += amount;
		}

		// returnerer hvor mange point spilleren har
		public int getPoints() {
				return points;
		}
}
```

Dice class

```
package Spil;

import java.util.Random;

public class Dice {
		Random r;
		int value;

		public Dice() {
				r = new Random();
		}

		// "kaster" terningen ved at gemme en tilfældig værdi mellem 1 og 6 i value
		public void roll() {
				value = r.nextInt(6) + 1;
		}

		// returnerer terningens værdi
		public int getValue() {
				return value;
		}
}
```

Dicecup class

```
package Spil;

import boundaryToMatador.GUI;

public class Dicecup {

		static Dice d1 = new Dice(); // de 2 terninger
		static Dice d2 = new Dice();
		static int getSum;

		public void shake() {
				d1.roll();
				d2.roll();
				getSum = d1.getValue() + d2.getValue();

				System.out.println("Første terning viser:\t" + d1.getValue());
				System.out.println("Anden terning viser:\t" + d2.getValue());

				// viser terningerne på spillepladen
				GUI.setDice(d1.getValue(), d2.getValue());
		}

		// returnerer terningens værdi
		public int getSum() {

				return getSum;

		}
}
```

Player class

```
package Spil;

import java.awt.Color;

public class Player
{
	private String name;
	private int position;
	private Color color;

	//Når et spiller object initialiseres gives et navn og en farve(mellem 1 og
6)
	public Player(String name, Color color)
	{
		this.name = name;
		this.color = color;
	}

	//returnerer spillerens navn
	public String getName()
	{
		return name;
	}

	public void setPosition(int pos)
	{
		position = pos;
	}

	public int getPosition()
	{
		return position;
	}

	//returnerer spillerens farve
	public Color getColor()
	{
		return color;
	}

}
```

TUI class

```
package Spil;

import boundaryToMatador.GUI;

public class Tui {

            public void velkommen() {
                        System.out.println("Velkommen til Terningspil");
                        System.out.println("");
            }

            public void spiller1(){
                        System.out.print("Spiller 1 skriv dit navn: ");
            }

            public void spiller2(){
                        System.out.print("Spiller 2 skriv dit navn: ");
            }

            public void turn1(String playerName) {
                        System.out.println(playerName + "'s tur.");
            }

            public void turn2(String playerName) {
                        System.out.println(playerName + "'s tur.");
            }

            public void kast(){
                        System.out.println("Indtast \"k\" for kast");
            }

            public void sum(){
            System.out.println("Terningerne viser samlet:\t " + Dicecup.getSum);
            }

            public void points1(String winName, String loseName){
            System.out.println(winName + " har vundet!\n" + loseName + " har tabt!");
                        System.out.println("");
                        System.out.println("Tryk på Ok for at afslutte spillet");
            }

            public void points2(String winName, String loseName){
            System.out.println(winName + " har vundet!\n" + loseName + " har tabt!");
                        System.out.println("");
                        System.out.println("Tryk på Ok for at afslutte spillet");
            }

            public void points3(String winName, String loseName){
            System.out.println(winName + " har vundet!\n" + loseName + " har tabt!");
                        System.out.println();
                        System.out.println("Tryk på Ok for at afslutte spillet");
            }
```

```
public void points4(String winName, String loseName){
System.out.println(winName + " har vundet!\n" + loseName + " har tabt!");
            System.out.println();
            System.out.println("Tryk på Ok for at afslutte spillet");
}

public void besked1(String winName, String loseName){
GUI.showMessage(winName + " har vundet!\n" + loseName + " har tabt!");
}

public void besked2(String winName, String loseName){
GUI.showMessage(winName + " har vundet!\n" + loseName + " har tabt!");
}

public void besked3(String winName, String loseName){
GUI.showMessage(winName + " har vundet!\n" + loseName + " har tabt!");
}

public void besked4(String winName, String loseName){
GUI.showMessage(winName + " har vundet!\n" + loseName + " har tabt!");
}

public void ekstratur(){
System.out.println("Terningerne viser tilsammen 10. Ekstra tur!");
}

public void slut(){
            System.out.println("");
            System.out.println("Spillet er slut! Slutresultat:");
}
```

```
public void Felter(){
                        String Tower = "Du finder en prinsesse der er låst inde i et
tårn, du befrier hende og sælger hende for 250 poin, +250 pointt";
                        String Crater = "Du falder ned i et krater, og betaler 200
point for at få hjælp, -200 point";
                        String Palace_Gates ="Du betaler 100 point i indgang til
Palacet, -100 point";
                        String Cold_Dessert = "Du får koldbrand i foden, og betaler en
fyr 20 point for at save den af, -20 point";
                        String Walled_City = "En fyr udfordrer dig til at klarte over
muren, du vinder 180 point, +180 point";
                        String Monastery = "Du begår indbrud i et forladt kloster, du
finder intet, 0 point";
                        String Black_Cave = "Du betaler 70 point for et glas med
ildfluer, -70 point";
                        String Huts_In_The_Mountain = "Du betaler 60 point for en
overnatning og en tvivlsom prostitueret, -60 point";
                        String The_Wearwall = "Du kaster 80 point mod en mur for at
smadre den, du får en ekstra tur, -80 point";
                        String The_Pit = "Du hjælper en person op af et hul, personene
slog dig omkuld og stjæler 90 point, -90 point";
                        String Goldmine = "Du hyrer en masse mennesker til at grave
efter guld for dig, +650 point";

String [] Felter = new String[] { Tower , Crater , Palace_Gates, Cold_Dessert ,
Walled_City , Monastery , Black_Cave , Huts_In_The_Mountain , The_Wearwall , The_Pit,
Goldmine};
            System.out.println(Felter[Dicecup.getSum - 2]);
            }

}
```

Bilag 2 – kode af test

```
package Spil;


public class Test {

            public static void main(String[] args) {

Dice d1 = new Dice();
Dice d2 = new Dice();
int[] arr = new int[] { 250, -200, -100, -20, 180, 0, -70, -60, -80, -90, 650 };
boolean gameRunning = true;

            d1.roll();
            d2.roll();
            int sum = d1.getValue() + d2.getValue();
                        Balance b1 = new Balance();

                        System.out.println("Start score: 1000");
                        b1.addPoints(arr[sum - 2]);
                        System.out.println("Terning sum: " + sum + "    Point: " +
arr[sum -2] + "         Samlet score: " + b1.getPoints());

                        while (gameRunning) {
             if (b1.getPoints() >= 3000){
                         System.out.println("Spillet er vundet");
                         gameRunning = false;
             } else if (b1.getPoints() >= 0){
                        d1.roll();
                        d2.roll();
                        int sum2 = d1.getValue() + d2.getValue();
                        b1.addPoints(arr[sum2 - 2]);
                        System.out.println("Terning sum: " + sum2 + "   Point: " +
arr[sum2 -2] + "        Samlet score: " + b1.getPoints());
            } else if (b1.getPoints() < 0){
                        System.out.println("Spillet er tabt");
                        gameRunning = false;}
                        }
            }
}
```