1.1.1

Startsemester oriëntatieverslag Leeruitkomst Software

Kaan Gogcay
457632
PD03
Lisette Penterman-Overkamp

Versie: 1.2

Versiebeheer

Versienummer	Datum	Auteur	Veranderingen
1.0	12-10-2020	Kaan Gogcay	Voorpagina ingevuld, versiebeheer ingevuld. Introductie gemaakt.
1.1	23-10-2020	Kaan Gogcay	Methods kopje toegevoegd
1.2	24-10-2020	Kaan Gogcay	Wedstrijd stukje toegevoegd ik ga hem nu inleveren

Inhoudsopgave

Inhoud

Inleiding

Welkom in het oriëntatieverslag van Kaan Gogcay. In dit verslag vertel ik even wat over mezelf. Vervolgens laat ik zien dat ik de basis van coderen beheers, dit doe ik door programma’s te bespreken die ik heb gemaakt deze 8 weken.

Introductie

Ik ben Kaan Gogcay. Met een Havo/VWO advies ben ik van 2 Havo/VWO gezakt naar 3 mavo. Ik wist dat ik duidelijk niet in mavo thuis hoorde en ik wist ook nog niet welke vervolgopleiding ik wilde gaan doen. Daarom besloot ik terug te gaan naar de Havo. Ik heb de Havo afgerond met een N&T profiel met Wiskunde D en Kunst Beeldend. Ik ben een grote fan van competitief gamen. Competitie heeft een speciaal plekje in mijn hart. Ook vind ik het wel leuk om piano te spelen. Ik kan het nog niet zo goed, maar ik probeer het te leren. Verder ben ik ook dol op sport. En dan heb ik het vooral over teamsport zoals: voetbal, volleybal, hockey en softbal. Ik kan erg serieus naar het leven kijken, maar ik kan ook gezellig zijn. Ten slotte wil ik nog kwijt dat ik rechtvaardigheid erg belangrijk vind. Als ik onrechtvaardigheid zie en ik kan er iets aan doen, dan doe ik dat ook.

Ten eerste wilde ik gewoon leren hoe je moest coderen. Coderen in console ging best goed. Maar een paar weken verder moesten we ook gebruik maken van forums en dit vond ik erg lastig. Daarom wilde ik ook leren hoe je moet programmeren in forums. Hierbij heb ik hulp gekregen van andere klasgenoten. Ook heb ik door middel van tutorial video’s wat bijgeleerd.

Reflectie / evaluatie

Terugkijken

Ik ga een voor een de programmeer concepten af en geef daar voorbeelden bij.

Variabelen

Ik heb zo een beetje in elk programma wel variabelen gebruikt, ik zal er proberen een van mijn eerste creaties erbij te pakken.

```
Je gaat nu een paar rekenvragen beantwoorden.
We beginnen met 3 oefenopgaven, deze opgaven hebben geen invloed op je cijfer
ROND KOMMA GETALLEN ALTIJD NAAR BENEDEN AF!
Klik op een toets om door te gaan.

Wat is 9 - 4
5
Correct!

Wat is 3 ÷ 4
0
Correct!

Wat is 9 - 2
7
Correct!
Je hebt bij de oefenopgaven 0 fouten gemaakt.

Nu ga je een toets maken met 20 rekenvragen. Je moet minstens 80% goed hebben, veel succes!

Wat is 4 ÷ 4
1
Correct!

Wat is 4 ÷ 3
1
Correct!

Wat is 3 + 4
```

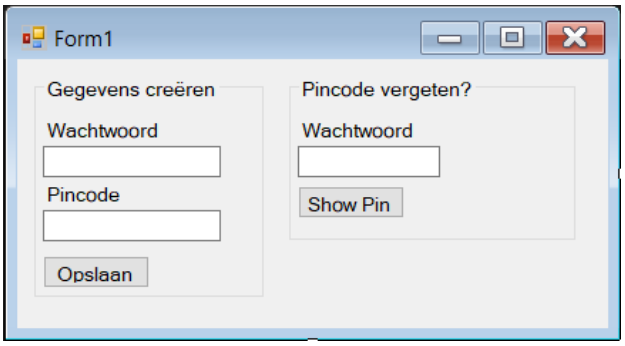
In dit programma heb ik een rekentoets gemaakt. Dit heb ik gedaan door random sommen te laten verschijnen.

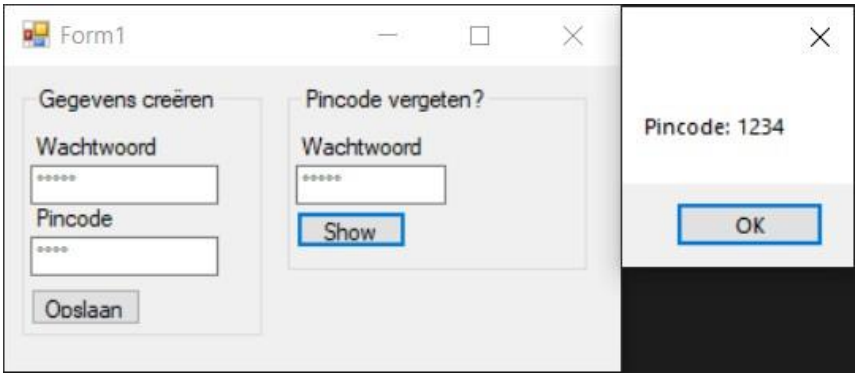
```
using Microsoft.VisualBasic;
using System;

namespace Rekenmachine
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Je gaat nu een paar rekenvragen beantwoorden.");
            Console.WriteLine("We beginnen met 3 oefenopgaven, deze opgaven hebben geen invloed op je cijfer");
            Console.WriteLine("ROND KOMMA GETALLEN ALTIJD NAAR BENEDEN AF!");
            Console.WriteLine("Klik op een toets om door te gaan.");
            Console.ReadKey();

            int ans = 82; // ans is geen 0 omdat het antwoord op de volgende vraag vaak 0 kan zijn, zo heb je minder oefenopgaven

            int fouten = 0;
            for (int i = 0; i < 3; i++)
            {
                Console.WriteLine();

                Random rnd = new Random();
                int getal1 = rnd.Next(1, 10);
                int getal2 = rnd.Next(1, 10);
                int plusMinKeerDelen = rnd.Next(1, 5);

                if (plusMinKeerDelen == 1) // +
                {
                    while (ans != getal1 + getal2)
                    {
                        Console.WriteLine("Wat is " + getal1 + " + " + getal2);
                        ans = Convert.ToInt32(Console.ReadLine());
```

Hier zie je dat ik **int getal1** en **int getal2** heb aangemaakt, dit zijn de twee getallen waarmee er telkens gerekend wordt. De reden dat ik deze getallen in een variabel heb gezet. is zodat ik ze na elke som een random waarde kan geven. Zo krijg je niet telkens dezelfde sommen.

String

Hier zie je mijn programma genaamd **PinCodeReminder**. In dit programma kun je een wachtwoord en een pincode creëren. Dat doe je door bij **Gegevens creëren** alles in te typen en op opslaan te drukken. Nadat je op opslaan hebt geklikt kun je bij **Pincode Vergeten?** je wachtwoord opnieuw invoeren en op **Show Pin** klikken. Vervolgens krijg je in een **MessageBox** je pincode te zien.





Dit heb ik allemaal voor elkaar gekregen door het variabel **string** te gebruiken. Hier linksonder kun je zien dat ik een string aanmaak genaamd **wachtwoord**. Het variabel wachtwoord kun je zelf een waarde geven. Laten we kijken naar het stukje code bij **button1_Click**. Je ziet dat de string wachtwoord gelijk staat aan **TB_wachtwoord**. Dit stukje code werkt pas als je op de button met opslaan klikt. Dus als je “Hallo” hebt staan in de textbox en je klikt op opslaan, dan is je wachtwoord “Hallo”.

```
public partial class PinCodeReminder : Form
{
    string wachtwoord;
    int pincode;

    1 reference
    public PinCodeReminder()
    {
        InitializeComponent();
    }


    1 reference
    private void button1_Click(object sender, EventArgs e)
    {

        wachtwoord = TB_wachtwoord.Text;
        pincode = Convert.ToInt32(TB_pin.Text);
    }
```

```
1 reference
private void button1_Click_1(object sender, EventArgs e)
{
    // haal wachtwoord en pincode terug

    string wachtwoordHerhalen;
    wachtwoordHerhalen = TB_pinVergeten.Text;

    if (wachtwoord == wachtwoordHerhalen)
    {

        MessageBox.Show("Pincode: " + pincode);
    }
}
```

In het rechter stukje zie je dat ik nog een **string** aanmaak genaamd **wachtwoordHerhalen**. wachtwoordHerhalen staat gelijk aan de wachtwoord-textbox in de groupbox **Pincode Vergeten?**. Je ziet dat ik hier een if-statement heb staan waarin staat **(wachtwoord == wachtwoordHerhalen)**. Dit houdt in dat je de wachtwoord die je net hebt opgeslagen in in de groupbox **Gegevens creëren** opnieuw moet invoeren in de groupbox **Pincode Vergeten**. Als de twee wachtwoorden inderdaad matchen, dan krijg je een **MessageBox** met je pincode. Als de twee wachtwoorden niet matchen dan gebeurt er helemaal niks.

Conditional Statements

```
// Player keuze
int playerKeuze = 0;

if (RB_steen.Checked) // Steen
{
    playerKeuze = 1;
    PB_player_steen.Visible = true;
}

else if (RB_papier.Checked) // Papier
{
    playerKeuze = 2;
    PB_player_papier.Visible = true;
}

else if (RB_schaar.Checked) // Schaar
{
    playerKeuze = 3;
    PB_player_schaar.Visible = true;
}

else
{
    MessageBox.Show("Maak eerst een keuze"); // Dit gebeurt niet
}
```

Dit is een stukje uit mijn **Steen Papier Schaar** programma. Hier zie dat ik **int playerKeuze** heb aangemaakt. Dat heb ik gedaan omdat ik wil dat de waarde van playerKeuze 1,2 of 3 moet worden. Maar het belangrijke voor nu zijn de **if-statements**. In het programma staan 3 **radio-buttons**. Een voor een wordt er gecontroleerd of de radio-buttons gecheckt zijn of niet. De radio-button die gecheckt is beslist of je steen, papier of schaar hebt gekozen. Als je geen radio-buttons hebt gecheckt, dan zal er een **MessageBox** verschijnen op je scherm waarin staat dat je eerst een keuze moet maken. Dit gebeurt omdat alle if-statements er boven niet kloppen. Daarom wordt er gedaan wat er in de **else** staat.

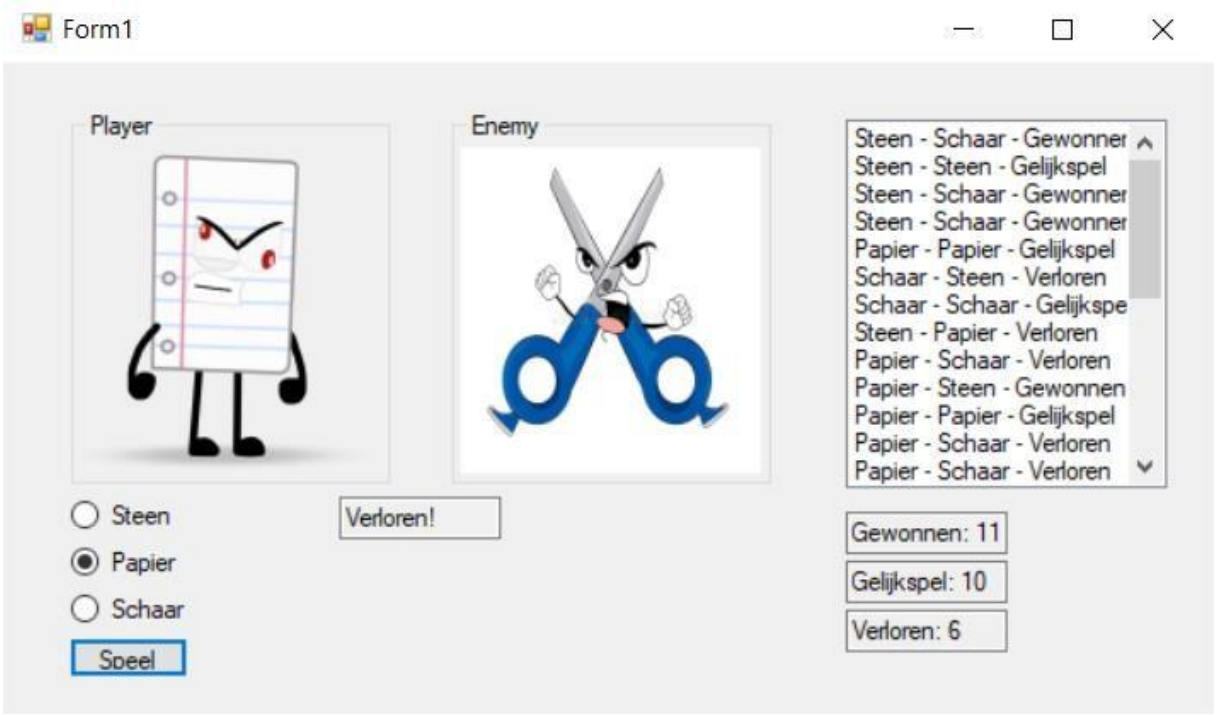
Loops

```
using System;

namespace Hoger_of_Lager
{
    class Program
    {
        static void Main(string[] args)
        {
            Random rnd = new Random();
            int number = rnd.Next(1, 100);

            Console.WriteLine("Raad het getal, het getal zit tussen 0 en 100.");

            int gok = Convert.ToInt32(Console.ReadLine());

            while (gok != number)
            {
                if (gok > number)
                {
                    Console.WriteLine("Lager");
                }

                if (gok < number)
                {
                    Console.WriteLine("Hoger");
                }
                gok = Convert.ToInt32(Console.ReadLine());
            }

            if (gok == number)
            {
                Console.WriteLine("Je hebt het getal geraden!");
            }

        }
    }
}
```

Hier zie je een mijn gehele **Hoger of Lager** programma. Ik heb een **while** loop gebruikt waarin staat **gok != number**. Dit betekent dat de stuk code in de while loop door blijft loopen totdat gok gelijk is aan number. Hier in dit programma is gok het nummer dat de speler intypt, number is het random getal die de speler probeert te raden. Dus als de speler het getal raadt is het voorbij.

List

Dit is het laatste voorbeeld uit mijn Steen Papier Schaar programma dat ik ga geven.



Zoals je hierboven kunt zien heb je rechts in het programma een **ListBox**. In deze ListBox staan de uitkomsten van de games die er zijn gespeeld. Het schema werkt zo *Player – Enemy – Result*. Telkens als er linksonder op de knop **Speel** wordt geklikt dan wordt er één item toegevoegd aan de ListBox.

```
// Results

// Gelijkspel
if (playerKeuze == 1 & enemyKeuze == 1) // Gelijkspel
{
    TB_result.Text = "Gelijkspel!";
    LB_results.Items.Add("Steen - Steen - Gelijkspel");
}
else if (playerKeuze == 2 & enemyKeuze == 2) // Gelijkspel
{
    TB_result.Text = "Gelijkspel!";
    LB_results.Items.Add("Papier - Papier - Gelijkspel");
}
else if (playerKeuze == 3 & enemyKeuze == 3) // Gelijkspel
{
    TB_result.Text = "Gelijkspel!";
    LB_results.Items.Add("Schaar - Schaar - Gelijkspel");
}
```

Hier zie je de mogelijke uitkomsten voor gelijkspel. De reden dat ik niet **(playerKeuze == enemyKeuze)** heb gedaan is omdat ik dan niet meer wist hoe ik de uitkomst in een list kon toevoegen. Dus heb ik besloten alles apart in te voeren. Maar zoals je kunt zien staat er bij elke if/else if **LB_results.Items.Add**. Hiermee voeg ik iets toe aan mijn ListBox. Wat daar langs in quotes staat is wat er wordt toegevoegd in de ListBox.

Method

Hier een voorbeeld uit mijn programma TerminalMethods.

```
    Console.WriteLine("Dit is mijn programma voor methods");
    keuzeMenu();
    keuze = Convert.ToInt32(Console.ReadLine());
    Line();
}

while (keuze == 1) // Instructies
{
    Console.WriteLine("<INSTRUCTIES>");
    Console.WriteLine("In dit programma heb je 3 gebruiker slots.");
    Console.WriteLine("in elke slot kun je een gebruiker opslaan");
    Console.WriteLine("ga naar 'Gebruiker aanmaken' om een gebruiker aan te maken");
    keuzeMenu();
    keuze = Convert.ToInt32(Console.ReadLine());
    Line();
}

while (keuze == 2) // Gebruikers
{
    Console.WriteLine("<GEBRUIKERS>");
    Console.WriteLine(gebruiker1);
    Console.WriteLine(gebruiker2);
    Console.WriteLine(gebruiker3);
    Console.WriteLine();
    keuzeMenu();
    keuze = Convert.ToInt32(Console.ReadLine());
    Line();
```

Je ziet hier dat de methods **keuzeMenu();** en **Line();** meerdere keren voorkomen. Ik heb in dit programma een menu gemaakt, ook kun je in dit programma gebruikers aanmaken. Op de volgende pagina laat ik eerst even zien hoe het eruit zien als je het programma gebruikt.

Je begint dus bij het Menu, vanuit het menu kun je naar andere plekken. Je ziet het zelfde menu steeds maar weer terug komen. Dat is de method keuzeMenu(); ook zie je de lijn vaker terugkomen, dat is de method Line();

```
// Functions
static void keuzeMenu()
{
    Console.WriteLine("0) Menu");
    Console.WriteLine("1) Instructies");
    Console.WriteLine("2) Gebruikers");
    Console.WriteLine("3) Gebruiker aanmaken");
}

static void Line()
{
    Console.WriteLine("-----------------");
}
```

```
<MENU>
In dit programma kun je gebruikers aanmaken
Dit is mijn programma voor methods
0) Menu
1) Instructies
2) Gebruikers
3) Gebruiker aanmaken
1
-----------------
<INSTRUCTIES>
In dit programma heb je 3 gebruiker slots.
in elke slot kun je een gebruiker opslaan
ga naar 'Gebruiker aanmaken' om een gebruiker aan te maken
0) Menu
1) Instructies
2) Gebruikers
3) Gebruiker aanmaken
2
-----------------
<GEBRUIKERS>
Empty
Empty
Empty

0) Menu
1) Instructies
2) Gebruikers
3) Gebruiker aanmaken
3
-----------------
<GEBRUIKER AANMAKEN>
Kies een gebruiker slot
1) Empty
2) Empty
3) Empty
1
Voornaam: Kaan
Achternaam: Gogcay
Gebruiker toegevoegd!

0) Menu
1) Instructies
2) Gebruikers
3) Gebruiker aanmaken
2
-----------------
<GEBRUIKERS>
Kaan Gogcay
Empty
Empty
```

Ook heb ik in het programma een derde method aangemaakt genaamd **VerkeerdeWaarde();** deze method wordt gebruikt als je een getal invult die je niet hoort in te vullen. zie het volgende voorbeeld. hieronder

```
static void VerkeerdeWaarde()
{
    Console.WriteLine("Voer een geldige waarde in");
}
```

```
<MENU>
In dit programma kun je gebruikers aanmaken
Dit is mijn programma voor methods
0) Menu
1) Instructies
2) Gebruikers
3) Gebruiker aanmaken
5
----------------
Voer een geldige waarde in
3

<GEBRUIKER AANMAKEN>
Kies een gebruiker slot
1) Empty
2) Empty
3) Empty
5
Voer een geldige waarde in
3
Voornaam:
```

Enum

```
1 reference
public enum Weapon { Sword, Hammer, Spear, Axe, Scythe }
 1 reference
 public enum Element { Water, Grass, Fire, Rock, Electric}

 0 references
 static void Main(string[] args)
 {
     int WeaponChoice = -1;
     int ElementChoice = -1;

     Console.WriteLine("Get ready for the fight and pick a weapon!");
     Console.WriteLine("Sword=0, Hammer=1, Spear=2, Axe=3, Scythe=4");

     while (WeaponChoice >= 5 | WeaponChoice <= -1)
     {
         WeaponChoice = Convert.ToInt32(Console.ReadLine());
         if (WeaponChoice >= 5 | WeaponChoice <= -1)
         {
             Console.WriteLine("Invalid Answer. Please try again");
         }
     }
     var weapon = (Weapon)WeaponChoice;
     Console.WriteLine();
```

Hier zie je dat ik twee **enums** heb aangemaakt. Maar omdat ze vrijwel identiek zijn ga ik er eentje bespreken. In dit programma kun je een *elemental weapon* creëren. Als het programma start dan krijg je de vraag welk wapen je wilt. Dit kun je kiezen door een getal in te voeren. Je komt tegelijkertijd ook in een **while** loop terrecht. Deze while loop controleert of je een geldig cijfer invoert. Als je geen geldig cijfer invoert moet je opnieuw een cijfer invoeren totdat je een geldig cijfer hebt ingevoerd. Het komt er dus op neer dat je je uitrusting kunt kiezen. Valt verder niet veel over te vertellen.

```
    Console.WriteLine("Choose an element");
    Console.WriteLine("Water=0, Grass=1, Fire=2, Rock=3, Electric=4");
    while (ElementChoice >= 5 | ElementChoice <= -1)
    {
        ElementChoice = Convert.ToInt32(Console.ReadLine());
        if (ElementChoice >= 5 | ElementChoice <= -1)
        {
            Console.WriteLine("Invalid Answer. Please try again");
        }
    }
    var element = (Element)ElementChoice;
    Console.WriteLine();

    Console.WriteLine("You created a " + element + " " + weapon + ". Good luck in the fight!");

}
```

(rest van de code)

Leesbaarheid

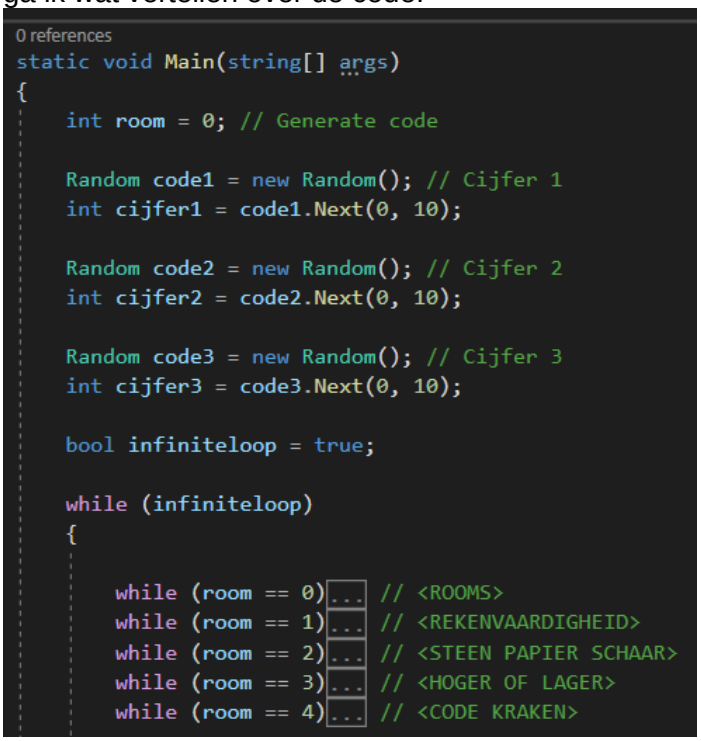
```
{
    // Reset images
    PB_player_schaar.Visible = false;
    PB_player_papier.Visible = false;
    PB_player_steen.Visible = false;

    PB_enemy_schaar.Visible = false;
    PB_enemy_papier.Visible = false;
    PB_enemy_steen.Visible = false;


    // Player keuze
    int playerKeuze = 0;

    if (RB_steen.Checked) // Steen
    {
        playerKeuze = 1;
        PB_player_steen.Visible = true;
    }

    else if (RB_papier.Checked) // Papier
    {
        playerKeuze = 2;
        PB_player_papier.Visible = true;
    }

    else if (RB_schaar.Checked) // Schaar
    {
        playerKeuze = 3;
        PB_player_schaar.Visible = true;
    }

    else
```

```
// Results

// Gelijkspel
if (playerKeuze == 1 & enemyKeuze == 1) // Gelijkspel
{
    TB_result.Text = "Gelijkspel!";
    LB_results.Items.Add("Steen - Steen - Gelijkspel");
}
else if (playerKeuze == 2 & enemyKeuze == 2) // Gelijkspel
{
    TB_result.Text = "Gelijkspel!";
    LB_results.Items.Add("Papier - Papier - Gelijkspel");
}
else if (playerKeuze == 3 & enemyKeuze == 3) // Gelijkspel
{
    TB_result.Text = "Gelijkspel!";
    LB_results.Items.Add("Schaar - Schaar - Gelijkspel");
}

// Win/Loss
else if (playerKeuze == 1 & enemyKeuze == 2) // Steen, Papier
{
    TB_result.Text = "Verloren!";
    LB_results.Items.Add("Steen - Papier - Verloren");
}
else if (playerKeuze == 1 & enemyKeuze == 3) // Steen, Schaar
```

Dit zijn 2 afbeeldingen van mijn **Steen Papier Schaar** programma. Zoals je kunt zien heb in mijn variabelen passende namen gegeven. Kijk bijvoorbeeld naar **playerKeuze** en **enemyKeuze**. Ik hoef niet een te zeggen wat ze doen. Want als je weet dat het over steen papier schaar gaat, dan begrijp je al dat playerKeuze de keuze van de speler is. En dat enemyKeuze de keuze van de tegenstander is. Verder probeer ik boven elk stukje code te zetten waar het over gaat door middel van **comments**. Kijk bijvoorbeeld naar ***// Player keuze***, ***// Reset images***, ***// Results***. Het beschrijft wat de code doet met zo min mogelijk woorden. Ook staat er langs de code af en toe een comment, zo kun je erg makkelijk volgen wat er gebeurt als je iets doet en zo weet je waar je je bevind in de code. Ten slotte wil ik het nog hebben over de **forms** die ik heb gebruikt. Telkens als ik een form gebruik geef ik het een passende naam. Eerst kijk ik naar wat ik gebruik, is het een **picturebox**? Dan zet ik er **PB_** voor. Is het een **radio-button**? Dan zet ik er **RB_** voor. Is het een **textbox**? Dan **TB_**. Door de forms te benamen op deze manier is het erg simpel om te weten wat alles is, en wat het doet.

Wedstrijd

Voor mijn wedstrijd programma had ik een Code breaker in gedachten. Dat is dus een programma waarin je cijfers verzamelt en uiteindelijk de cijfers in de juiste volgorde zet om de code te raden. Ik heb besloten dit in console te gaan maken. De reden hiervoor is omdat ik in console werken altijd makkelijker vond. Nu ga ik wat vertellen over de code.



```
0 references
static void Main(string[] args)
{
    int room = 0; // Generate code

    Random code1 = new Random(); // Cijfer 1
    int cijfer1 = code1.Next(0, 10);

    Random code2 = new Random(); // Cijfer 2
    int cijfer2 = code2.Next(0, 10);

    Random code3 = new Random(); // Cijfer 3
    int cijfer3 = code3.Next(0, 10);

    bool infiniteloop = true;

    while (infiniteloop)
    {

        while (room == 0)... // <ROOMS>
        while (room == 1)... // <REKENVAARDIGHEID>
        while (room == 2)... // <STEEN PAPIER SCHAAR>
        while (room == 3)... // <HOGER OF LAGER>
        while (room == 4)... // <CODE KRAKEN>
```

Concept

Op het begin van de code zie je dat er 3 random cijfers tussen -1 en 10 worden aangemaakt. Als je de code gaat raden. Hieronder zie je een if-statement waarin staat **(Slot1 == cijfer1 & Slot2 == cijfer2 & Slot3 == cijfer3)**. De cijfers zijn de getallen die aan het begin gerandomised zijn. De SlotX zijn de getallen die je invoert als je de code gaat raden. Hier wordt er dus als het ware gecheckt of je de code goed invoert. Stel dat je 2/3 cijfers weet en de derde wilt gaan raden. Dat gaat hem niet worden, want je kunt maar 1 keer raden. Als je een poging hebt gewaagd stopt het programma. De code raden gebeurt allemaal in **(room == 4)** oftewel **<CODE KRAKEN>**. Daarnaast zie je nog de 4 andere rooms.

```
while (room == 4) // <CODE KRAKEN>
{
    Console.WriteLine("<CODE KRAKEN>");
    Console.WriteLine("Je hebt 1 poging om de code te kraken");
    Console.Write("Cijfer 1: ");
    int Slot1 = Convert.ToInt32(Console.ReadLine());
    Console.Write("Cijfer 2: ");
    int Slot2 = Convert.ToInt32(Console.ReadLine());
    Console.Write("Cijfer 3: ");
    int Slot3 = Convert.ToInt32(Console.ReadLine());

    if (Slot1 == cijfer1 & Slot2 == cijfer2 & Slot3 == cijfer3)
    {
        Console.WriteLine("Code geaccepeteerd!");
        line();
        infiniteloop = false;
        room = 1415926534;
    }

    else
    {
        Console.WriteLine("Fout! Je hebt gefaald");
        room = 1415926535;
```

Rooms

In **(room == 0)** oftewel **<ROOMS>** kun je alle rooms zien.

```
<ROOMS>
1) Rekenvaardigheid
2) Steen Papier Schaar
3) Hoger of Lager
4) Code Kraken
```

Dat ziet er zo uit. Als je het getal 1 invult ga je naar Rekenvaardigheid en als je 2 invult ga je naar Steen Papier Schaar enz. De code in deze room is niet al te groot.

```
while (room == 0) // <ROOMS>
{
    rooms();
    room = Convert.ToInt32(Console.ReadLine()); // Naar andere room gaan
    line();
} // <ROOMS>
```

Je ziet hier 2 methods, **rooms()** en **line()**. rooms() zou je overbodig kunnen noemen omdat het nergens anders voorkomt dan in de room **<ROOMS>**, maar ik vond het wel mooi en overzichtelijk als er zo weinig stond. De method **line()** wordt wel in elke room gebruikt. Hier zijn de 2 methods.

```
static void rooms()
{
    Console.WriteLine("<ROOMS>");
    Console.WriteLine("1) Rekenvaardigheid");
    Console.WriteLine("2) Steen Papier Schaar");
    Console.WriteLine("3) Hoger of Lager");
    Console.WriteLine("4) Code Kraken");
} // Rooms
```

```
static void line()
{
    Console.WriteLine("-----------------------------------------");
} // Line
```

Verder zie je nog het stukje code **room = Convert.ToInt32(Console.ReadLine());**. De comment erlangs zegt het al, je gaat naar een andere room.

Rekenvaardigheid

In room 1 oftewel **<REKENVAARDIGHEID>** kun je je eerste cijfer krijgen.

```
while (room == 1) // <REKENVAARDIGHEID>
{
    Console.WriteLine("<REKENVAARDIGHEID>");
    Console.WriteLine("Rond deelsommen altijf af naar beneden");
    if (Rekentoets() == 0)
    {
        Console.WriteLine("Code: " + cijfer1 + " _ _");
    }

    returnToMenu();
    line();
    room = 0;
} // <REKENVAARDIGHEID>
```

Voor deze room heb ik een programma gebruikt die ik ooit eerder heb gemaakt. Dat programma heette als het goed is 'Rekentoets'. Deze programma heb ik in de method **Rekentoets()** gezet. wel heb ik een aantal dingen verandert. Het enige wat het programma nu eigenlijk nog doet is 10 random sommen creëren met getallen tussen 0 en 21. Het aantal fouten dat je maakt in dit programma wordt bijgehouden in de method

Ook zag je nog de method **returnToMenu()** deze method laat je op een toets klikken en je krijgt er een kleine tekst bij. Staat ook in elke room dus is wel fijn.

```
static void returnToMenu()
{
    Console.WriteLine("Klik een toets om terug te gaan naar <ROOMS>");
    Console.ReadKey();
} // Terug naar menu
```

Rekentoets().

```
static int Rekentoets()
{
    int fouten = 0;

    for (int i = 0; i < 10; i++) // Begin toets
    {
        int ans = 82;

        Random rnd = new Random();
        int getal1 = rnd.Next(1, 21);
        int getal2 = rnd.Next(1, 21);
        int plusMinKeerDelen = rnd.Next(1, 5);

        Console.WriteLine();

        if (plusMinKeerDelen == 1)...

        if (plusMinKeerDelen == 2)...

        if (plusMinKeerDelen == 3)...

        if (plusMinKeerDelen == 4)...
    }
    return fouten;
```

Zoals je kan zien wordt uiteindelijk de waarde **fouten** gereturnd. Als deze waarde gelijk is aan 0 (dus 0 fouten) dan krijg je het eerste cijfer van de code te zien. Als je 1 of meer fouten had dan krijg je geen stukje van de code. Vervolgens wordt je weer teruggestuurd naar het menu zodat je het opnieuw kan proberen als je teveel fouten had, of je kan naar een andere room.

Steen Papier Schaar

Dit is in room 2. Hier speel je steen papier schaar tegen een bot. Voor dit programma heb ik een enum gebruikt. Ook heb ik het een en ander gekopieerd van mijn forms versie van steen papier schaar. Het doel hier is om gelijk te spelen tegen de bot. Eerst had ik in gedachten om boter kaas en eieren te maken en dat je perse moest verliezen. Maar uiteindelijk had ik dat toch niet gedaan omdat ik in console ging werken en ik snapte eigenlijk ook niet hoe ik moest beginnen en ik dacht steen papier schaar staat toch al klaar dus, vandaar.

```
while (room == 0)... // <ROOMS>
while (room == 1)... // <REKENVAARDIGHEID>
while (room == 2) // <STEEN PAPIER SCHAAR>
{
    Console.WriteLine("<STEEN PAPIER SCHAAR>");
    if (SteenPapierSchaar() == 0)
    {
        Console.WriteLine("Code: _ " + cijfer2 + " _");
    }

    returnToMenu();
    line();
    room = 0;
} // <STEEN PAPIER SCHAAR>
```

Je ziet hier dat als je return waarde van **SteenPapierSchaar()** gelijk is aan 0 dat je dan het 2de cijfer krijgt van de code. Dit betekent eigenlijk, als je gelijk speelt dat krijg je een cijfer. De kans dat je gelijk speel in steen papier schaar is 1/3. Deze room is niet zo spannend vergeleken met de andere rooms.

Hoger of Lager

Dit is room 3. Het is basically gewoon mijn hoger of lager programma copy paste naar een method.

```
while (room == 3) // <HOGER OF LAGER>
{
    Console.WriteLine("<HOGER OF LAGER>");

    if (HogerOfLager() <= 6)
    {
        Console.WriteLine("Code: _ _ " + cijfer3);
    }

    returnToMenu();
    line();
    room = 0;
} // <HOGER OF LAGER>
```

Als **(HogerOfLager() <= 6)** dan krijg je de derde cijfer. Maar wat betekent dat eigenlijk. De method houdt bij hoe veel pogingen je doet om het getal te raden. Je aantal pogingen is dus ook de return waarde. Als je vaker dat 6 keer hebt gegokt dan krijg je uiteindelijk niet het 3de cijfer.

Code Kraken

In deze room kun je de code een keer raden en dan stopt het programma

```
while (room == 4) // <CODE KRAKEN>
{
    Console.WriteLine("<CODE KRAKEN>");
    Console.WriteLine("Je hebt 1 poging om de code te kraken");
    Console.Write("Cijfer 1: ");
    int Slot1 = Convert.ToInt32(Console.ReadLine());
    Console.Write("Cijfer 2: ");
    int Slot2 = Convert.ToInt32(Console.ReadLine());
    Console.Write("Cijfer 3: ");
    int Slot3 = Convert.ToInt32(Console.ReadLine());

    if (Slot1 == cijfer1 & Slot2 == cijfer2 & Slot3 == cijfer3)
    {
        Console.WriteLine("Code geaccepeteerd!");
        line();
        infiniteloop = false;
        room = 1415926534;
    }

    else
    {
        Console.WriteLine("Fout! Je hebt gefaald");
        room = 1415926535;
        infiniteloop = false;
        line();
    }
} // <CODE KRAKEN>
```

Dit is het stukje code dat erbij hoort. De getallen die je gokt komen in de **Slots**. De **Slots** worden vergeleken met de **Cijfers**. Als deze allemaal gelijk zijn dan krijg je “Code geaccepteerd” klopt het niet dan krijg je “Fout! Je hebt gefaald. Verder zie je ook in beide rooms **infiniteloop = false**. De reden dat ik dit doe is zodat de code niet meer door blijft loopen. Want de reden dat je heel de tijd van room naar room kan is omdat er een while loop aan de gang is.

```
bool infiniteloop = true;

while (infiniteloop)
{

    while (room == 0)... // <ROOMS>
    while (room == 1)... // <REKENVAARDIGHEID>
    while (room == 2)... // <STEEN PAPIER SCHAAR>
    while (room == 3)... // <HOGER OF LAGER>
    while (room == 4) // <CODE KRAKEN>
```

De infiniteloop in deze while loop is altijd true totdat je de code probeert te raden. Dan wordt infinite loop false en kun je uit het programma.

Ten slotte zag je ook nog dat je naar een room gestuurd werd bij beide situaties.

```
if (room == 1415926535 & infiniteloop == false) // Foute code
{
    Console.WriteLine("Loser");
}

else if (room == 1415926534 & infiniteloop == false) // Goede code
{
    Console.WriteLine("Kanjer");
}
```

Deze rooms kun je alleen in als room gelijkstaat aan dat getal én als infinite loop false is. De reden dat deze in aparte plekken staan is omdat de code anders door blijft loopen, heel raar. Dus stel dat ik deze if's hierboven weg haal en de inhoud onder **infiniteloop = false** (plaatje vorige blz) zet dan stopt de code niet.

Nawoord

Ik heb in dit semester zo een beetje alles geleerd wat je hier ziet. Ik kon nog niet coderen. Ik kan er wel een lang verhaal van maken maar alles wat ik tot nu toe kan heb ik in dit semester geleerd. Dingen die ik vaker heb gebruikt maar niet heb aangetoond is **random**.

Ik begrijp methods nog niet helemaal heb ik het gevoel. Dan bedoel ik met return waardes. Maar als ik software niet ga kiezen dan wil ik me niet verder ontwikkelen.

Ik ben best wel trots op de Steen Papier Schaar programma. Dat komt omdat ik daar zo een beetje alles wat ik erin wilde zetten erin heb gezet. daarmee bedoel ik dat er plaatjes inzitten, er zit ook een list in. Het is gewoon een compleet programma dat er ook verzorgd uitziet. Ook ben ik erg trots op de Rekentoets die ik de eerste paar dagen had gemaakt. Hier ben ik zo trots op omdat ik het zo vroeg in de opleiding had gemaakt. Ook ben ik trots op mijn wedstrijd opdracht het is geworden zoals ik het wilde hebben. Vind het ook wel leuk om te spelen.

Meer methods gebruiken. Ook is het belangrijk de basis zo snel mogelijk te begrijpen. Maar als ik eerlijk, het meeste in software lukte gewoon dus ik vind het wel prima hoe het gegaan is

Vooruitkijken

In het volgende semester weet ik nog niet waar ik me op wil focussen. Ik weet alleen dat ik sowiesoe geen business wil kiezen. Business spreekt mij echt niet aan. Software en technology vind ik wel leuk omdat het niet al te lastig is. Media heb ik nog niet zo een heel goed beeld bij, maar misschien kies ik het wel. Het was wel leuk een game maken en een vlog maken. Infra is het lastigst van allemaal, maar ik vind het wel het interessantst. Dus ik moet even nog goed nadenken.