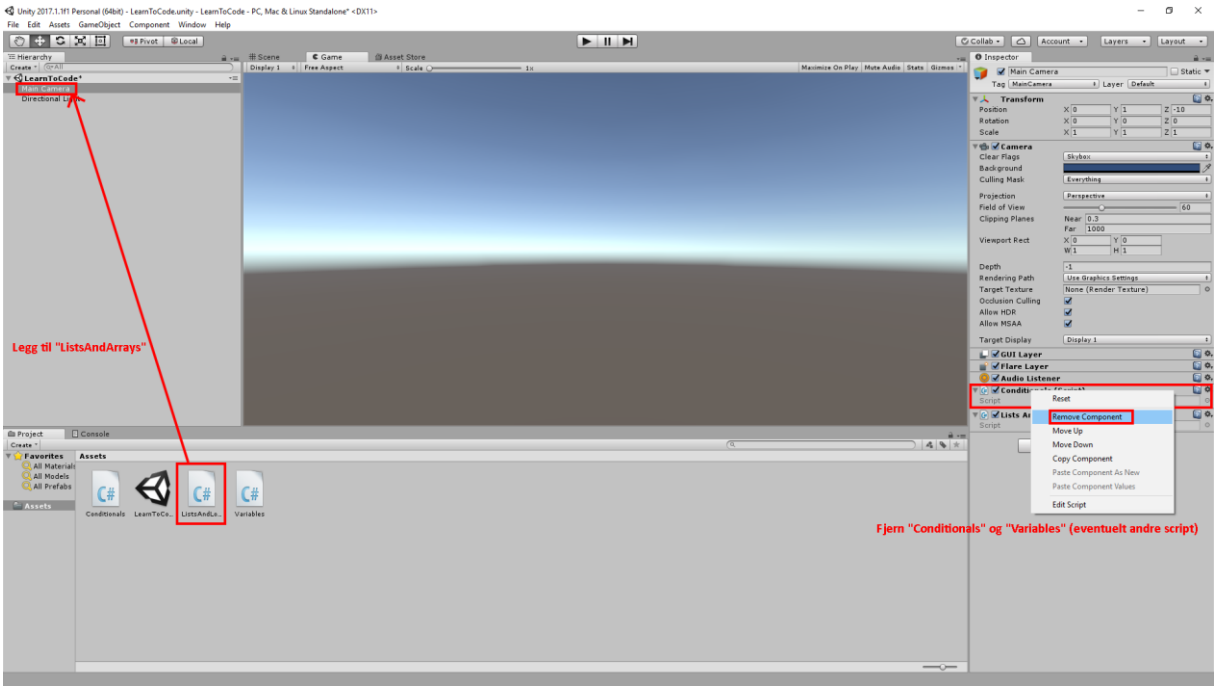
Lær å kode i Unity – del 3: Arrays & loops

I denne delen skal vi lære å jobbe med lister/tabeller (arrays) og løkker (loops).

Lag et nytt C# script og kall det «ListsAndLoops»

Husk å legge til det nye scriptet ditt under Main Camera!

Det kan være lurt å fjerne scriptene du har brukt tidligere hvis du bruker samme prosjekt til alle scriptene. Da slipper du at alt fra tidligere script skrives ut sammen med det nye scriptet. Dette gjøres på følgende måte: Klikk på Main Camera, finn «Conditionals» og «Variables» og eventuelt andre script du har lagt til, høyreklikk og velg Remove Component.



Arrays (tabeller)

En array er en éndimensjonal tabell eller en liste og består av et bestemt antall dataelementer av samme type. Elementene er indeksert fra 0 og oppover.

Eksempler på arrayer:

Array av typen string (tekst) som viser en liste over noen brukernavn:

Element/plass nr.	0	1	2	3
string (brukernavn)	Boogie45	LittleW0nder	DesktopGamer83	DouchebagPotato

Array av typen int (heltall) som viser en liste med aldere:

Element nr.	0	1	2	3	4	5
int (alder)	17	12	36	19	21	28

Det er viktig å merke seg at en array alltid starter på plass 0 i listen!

Så hvorfor trenger vi å kunne lage lister i programmet vårt? Tenk deg for eksempel at du spiller et multiplayer spill. Mye data må lagres i slike spill, blant annet brukernavn. For å kunne få tilgang til disse navnene på en oversiktlig måte kan det være lurt å ha dem lagret i en liste/array.

Vi lager en liste med noen tilfeldige brukernavn:

- Boogie45
- LittleW0nder
- DesktopGamer83
- DouchebagPotato

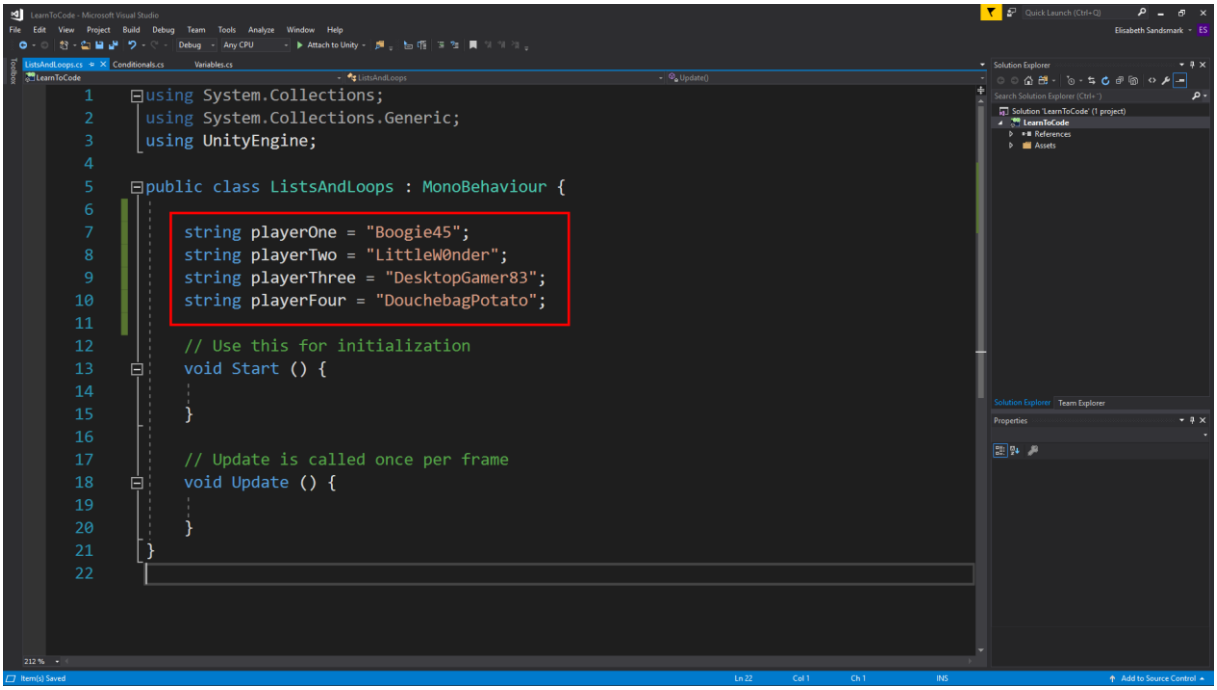
Det er forskjellige måter å få tilgang til disse brukernavnene på i koden vår. La oss begynne med måten dere har vært vant med til nå, med å lage en «string» til hvert av brukernavnene:

string playerOne = «Boogie45»;

string playerTwo = «LittleW0nder»;

string playerThree = «DesktopGamer83»;

string playerFour = «DouchebagPotato»;



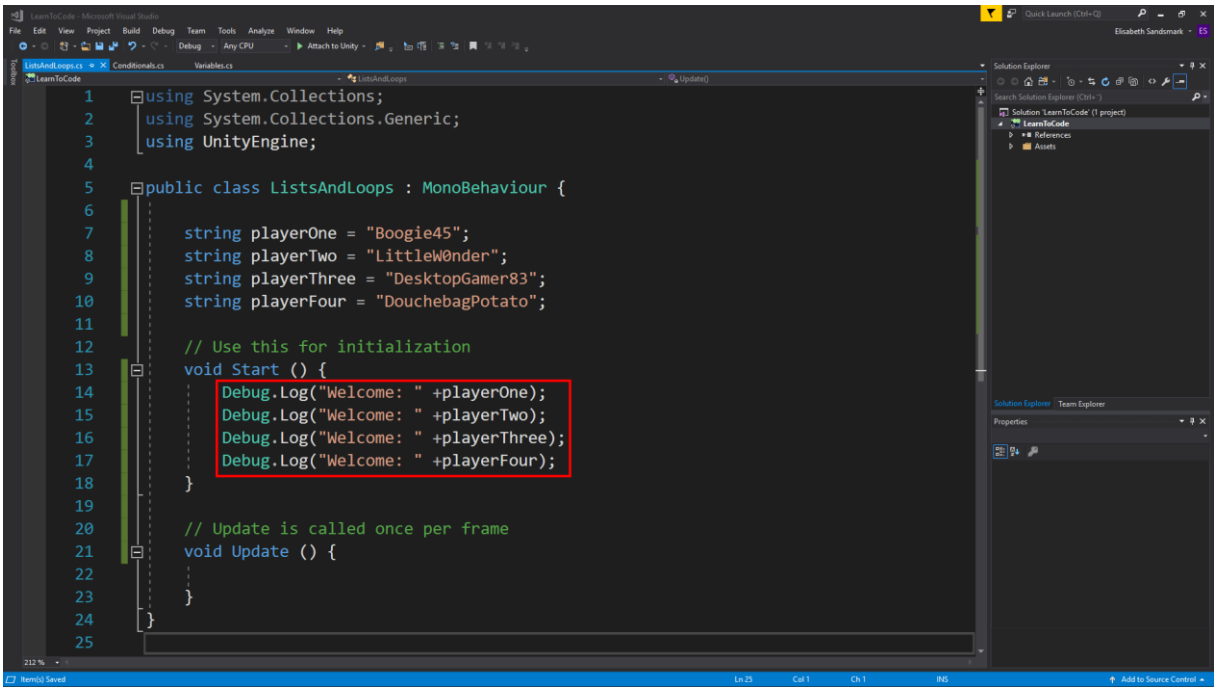
Vi legger også til en velkomstmelding til hver av spillerne på samme måte som vi har gjort tidligere:

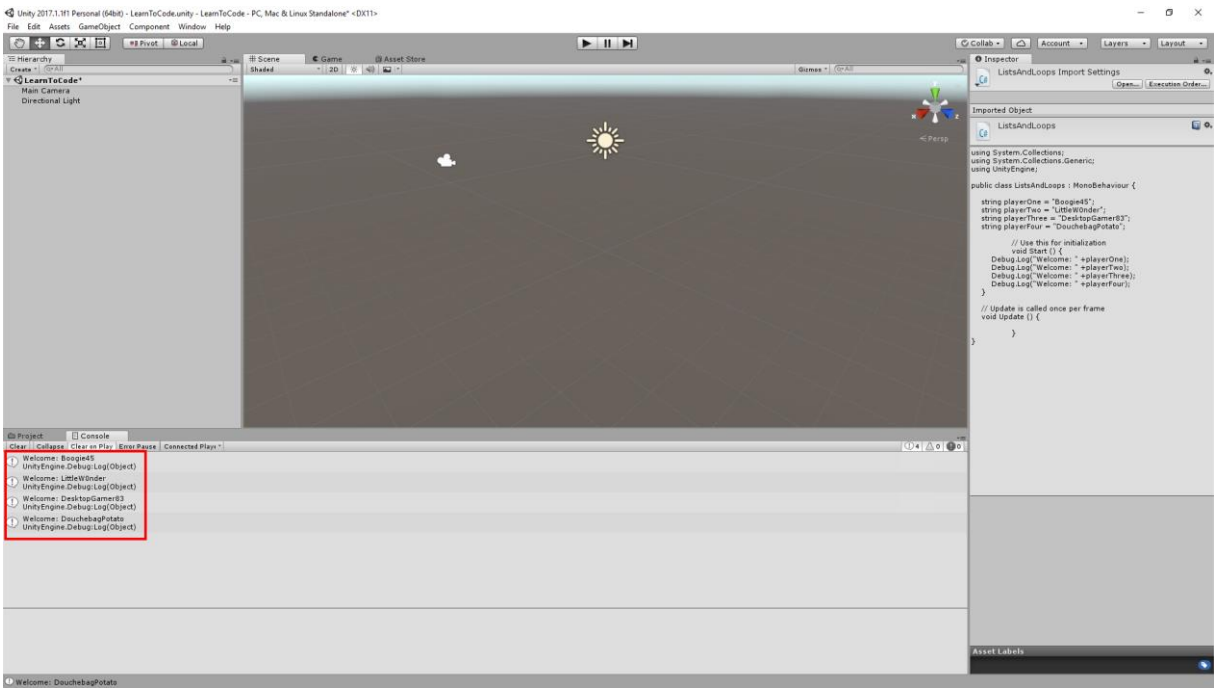
Debug.Log («Welcome: » + playerOne);

Debug.Log («Welcome: » + playerTwo);

Debug.Log («Welcome: » + playerThree);

Debug.Log («Welcome: » + playerFour);





Som du kanskje legger merke til så gjentar du mye av den samme koden om og om igjen her, og det kan være svært upraktisk dersom du har lange lister med navn osv. Tenk deg bare hvor mange brukernavn som ligger lagret i et stort multiplayer spill!

Innenfor koding har vi en huskeregel som vi kaller for «DRY» (don't repeat yourself). Hvis du gjentar mye av det samme i koden din blir koden din fort uoversiktlig og rotete. Det er viktig å bruke de verktøyene vi har til disposisjon, og ett av disse verktøyene er arrays.

Som vi nevnte tidligere har vi ulike typer arrays. De kan være av typen int, eller double, string, bool osv. og det er viktig å bruke riktig type variabel ut fra hva du ønsker å lagre i din array.

Deklarasjon av en array

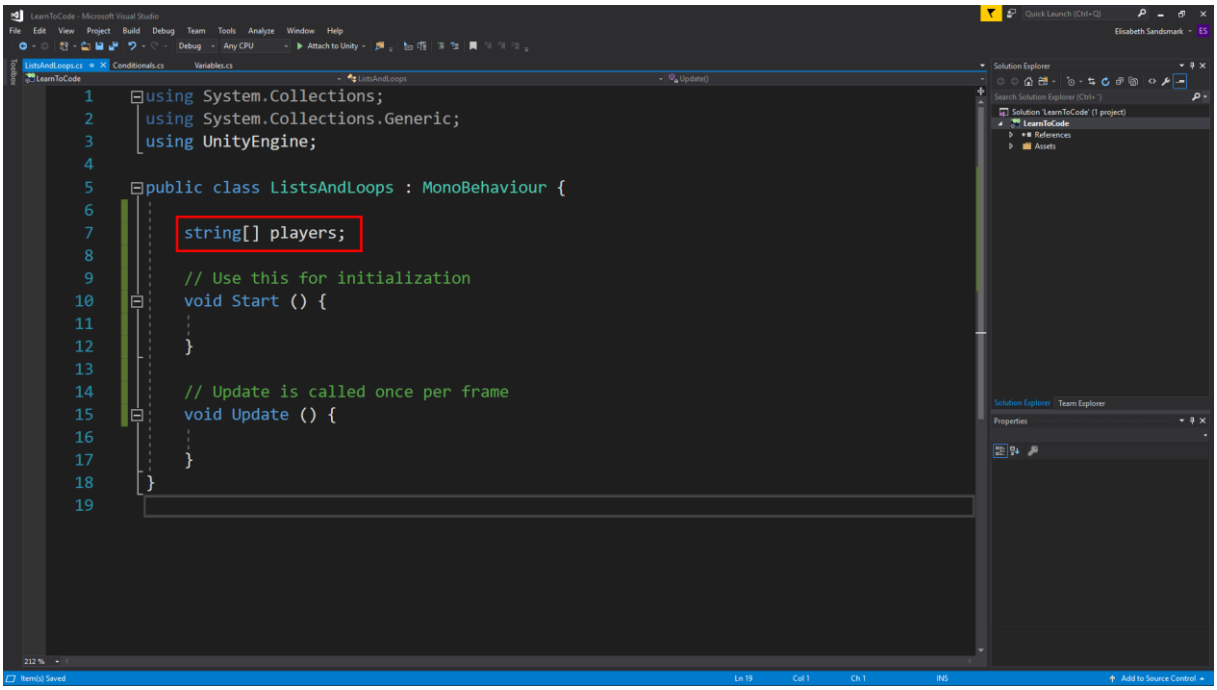
Måten vi lager en array på i C# er å først deklarere typenavn, etterfulgt av tegnene [ ] (disse tegnene indikerer at det er en array). Deretter gir vi arrayen et passende navn ut fra hva den inneholder. En array deklareres på følgende måte:

typenavn[ ] arraynavn;

Vi deklarerer en array for brukernavnene vi laget tidligere i oppgaven:

OBS! Husk å enten slette det du tidligere skrev eller legg det til som en kommentar /* */, vi trenger ikke den koden videre i oppgaven.

string[] players;

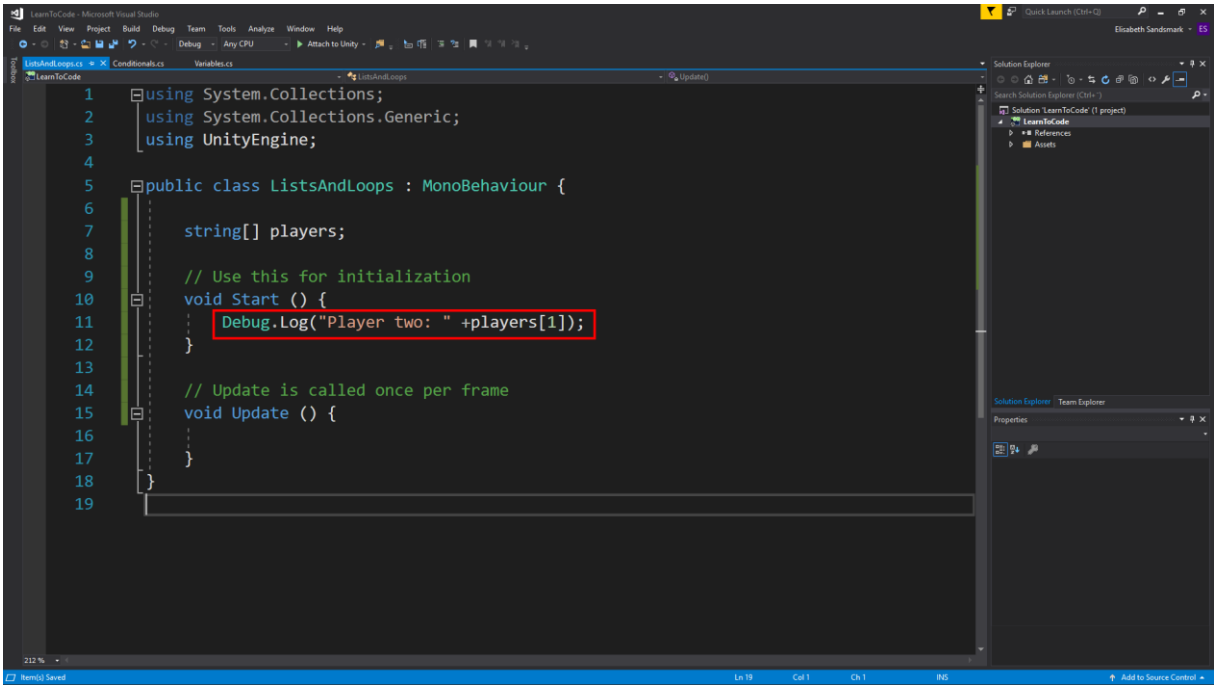


Vi har nå *deklarert* en array, men vi kan ikke bruke den enda fordi den ikke har blitt *opprettet*.

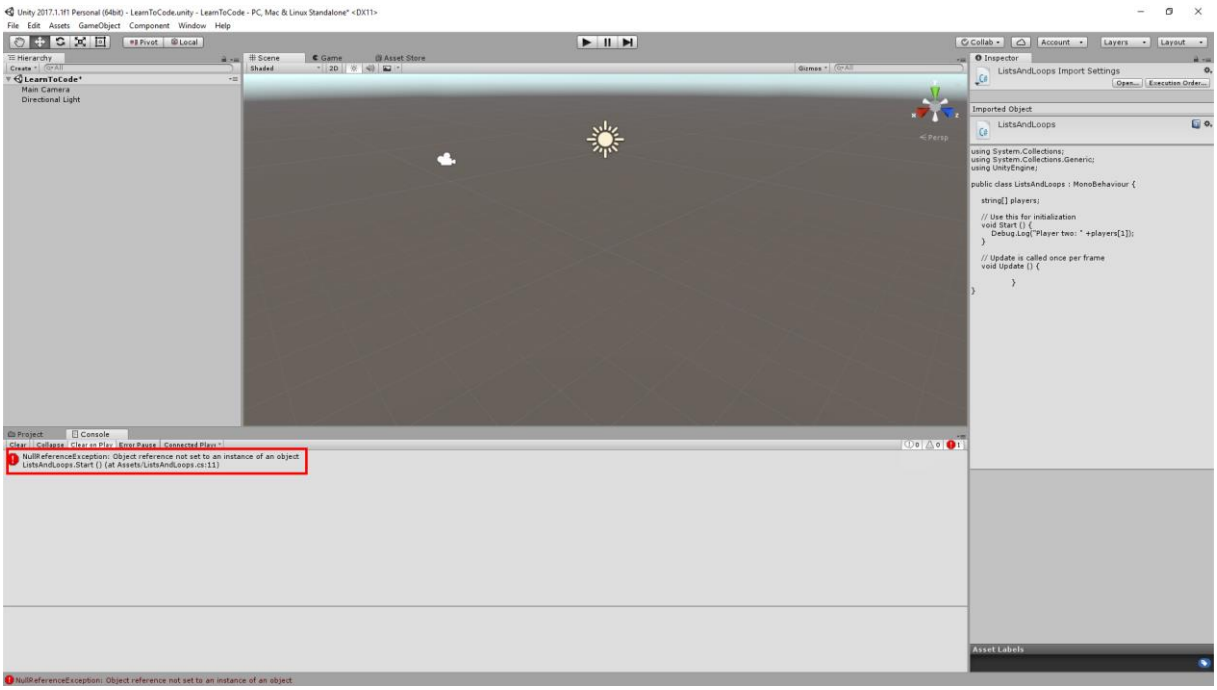
Opprettelse av en array

Lag en tekst som skriver ut navnet til f.eks. spiller to:

Debug.Log («Player two: » + players [1]);



Vår array har foreløpig ikke noe innhold og hvis vi prøver å hente noe ut fra en array som er tom vil du få noe som heter «NullReferenceExeption» og spillet ditt vil krasje (prøv å lagre og teste koden om du ønsker å se hva som skjer). Hvis du noen gang ser denne feilmeldingen «Object reference not set to an instance of an object» betyr det mest sannsynlig at du prøvde å få tilgang til noe som ikke har blitt opprettet enda (f.eks. når du prøver å få tilgang til informasjon i en tom tabell). Vi kan på en måte si at en array ikke eksisterer før den er opprettet.



Så hvordan oppretter vi en array? Det er flere måter å gjøre dette på, blant annet på følgende måte:

typenavn[] arraynavn = new typenavn [antall elementer];

Typenavn står for hvilken variabeltype du ønsker å lagre (eks. string, int, double…).

[] indikerer at det er en array.

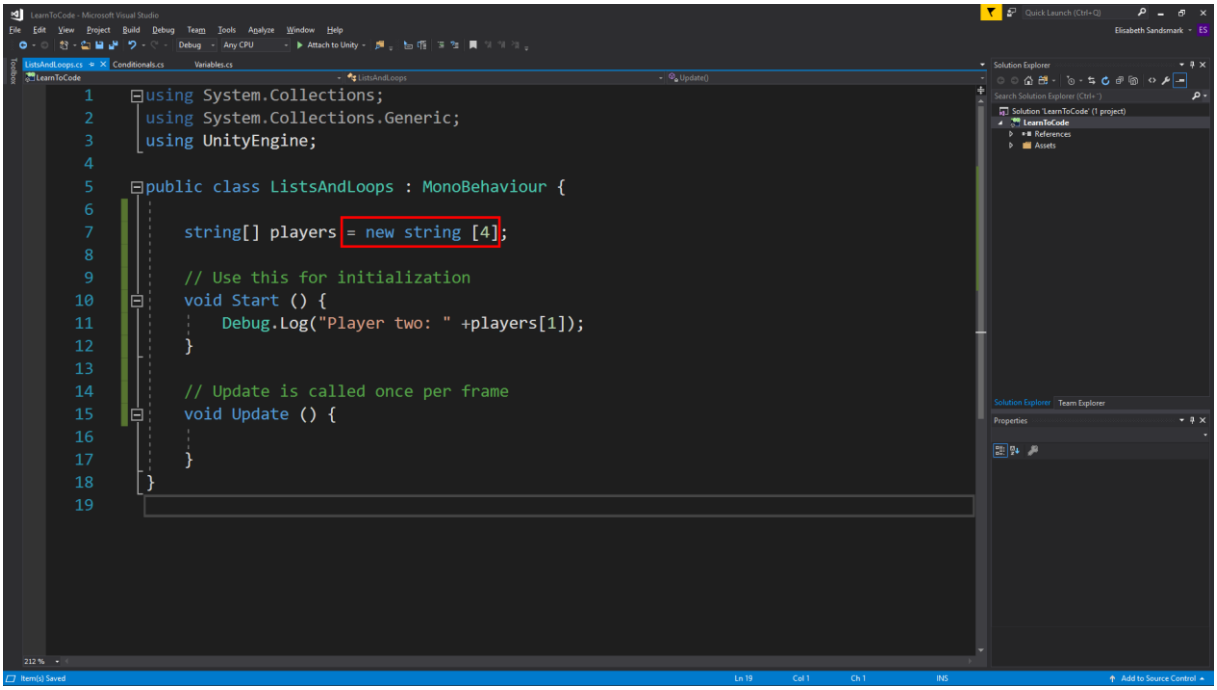
New-operatoren brukes i dette tilfellet for å be om det opprinnelige typenavnet for alle verdiene som skal opprettes. (Tenk deg f.eks. at du har en bilfabrikk og sjefen sier «det er på tide å lage et nytt kjøretøy». Du bruker dermed et «blueprint» på hvordan et kjøretøy skal lages og lager nye kjøretøy av ulike varianter. På samme måte lager vi ulike brukernavn i vårt eksempel.)

[antall elementer] angir hvor mange elementer (av den angitte typen) det skal være plass til i din array f.eks. [4] (gir plass til fire elementer). I C# må en array alltid ha en størrelse. Denne størrelsen kan ikke økes eller minkes underveis (ulikt mange andre kodespråk), men den kan være tom.

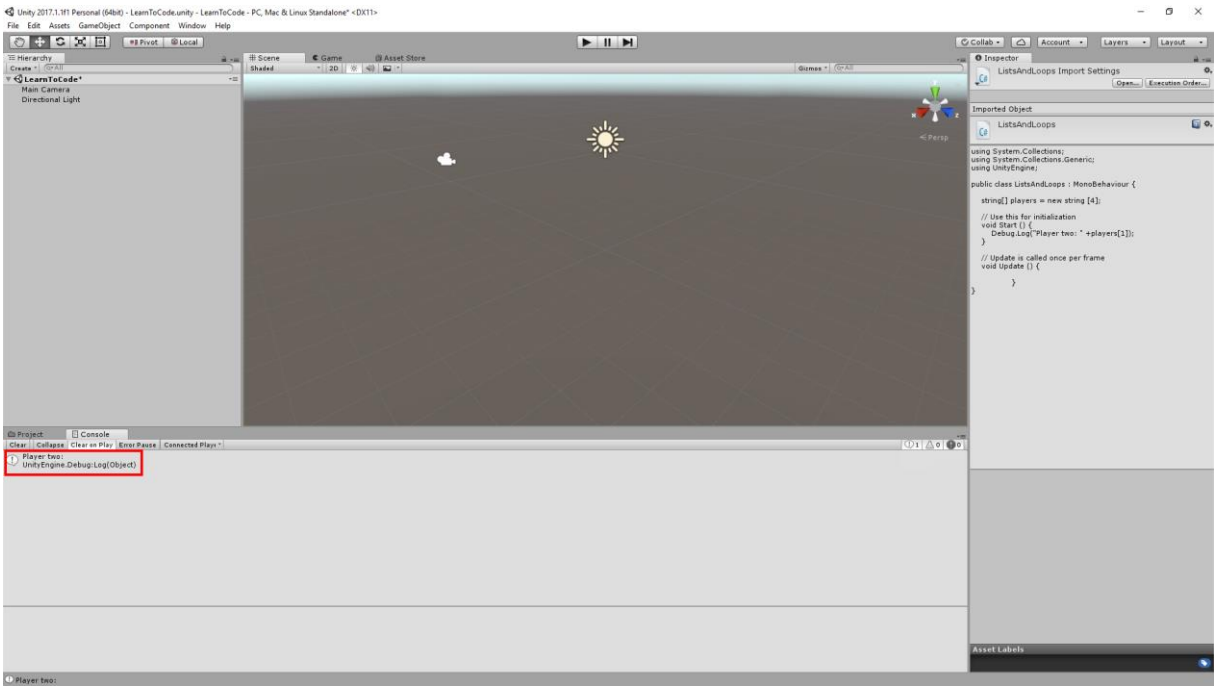
Et lite tips: Aldri bruk et gigantisk nummer i en array hvis du vet hvor mange elementer du skal opprette. Dersom du bruker et stort tall kommer det til å reservere mye plass i minnet og det kan føre til at spillet ditt kjører tregt, spesielt dersom det er et mobilspill! Så vær forsiktig med hvor mange elementer du legger til. Bruk ikke mer enn du trenger!

La oss deklarere vår array. I eksempelet har jeg brukt en array med 4 elementer/plasser, men du kan fritt legge til flere om du øsnker.

string[] players = new string [4];



Hvis du prøver å kjøre koden din nå så vil det fungere, men arrayen er tom og du vil ikke få noe navn i utskriften. Grunnen til at den fungerer nå er at vi har satt av fire plasser (som f.eks. på kino når du reserverer en plass ved å kjøpe billetter til deg og dine venner. Du får da tilgang til disse setene under kinofilmen, men plassene er ikke fylt før personene faktisk er der.).



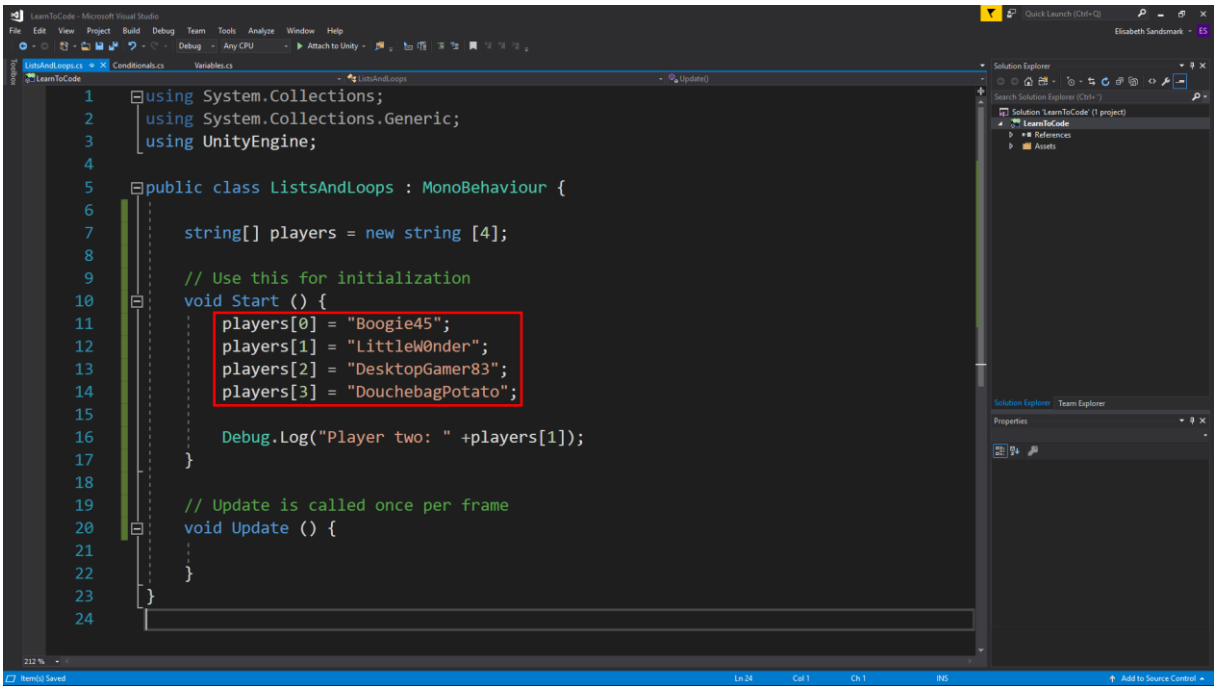
Det vi nå ønsker å gjøre er å gi hver av disse plassene en verdi. En måte å gjøre dette på er slik:

players [0] = «Boogie45»;

players [1] = «LittleW0nder»;

players [2] = «DesktopGamer83»;

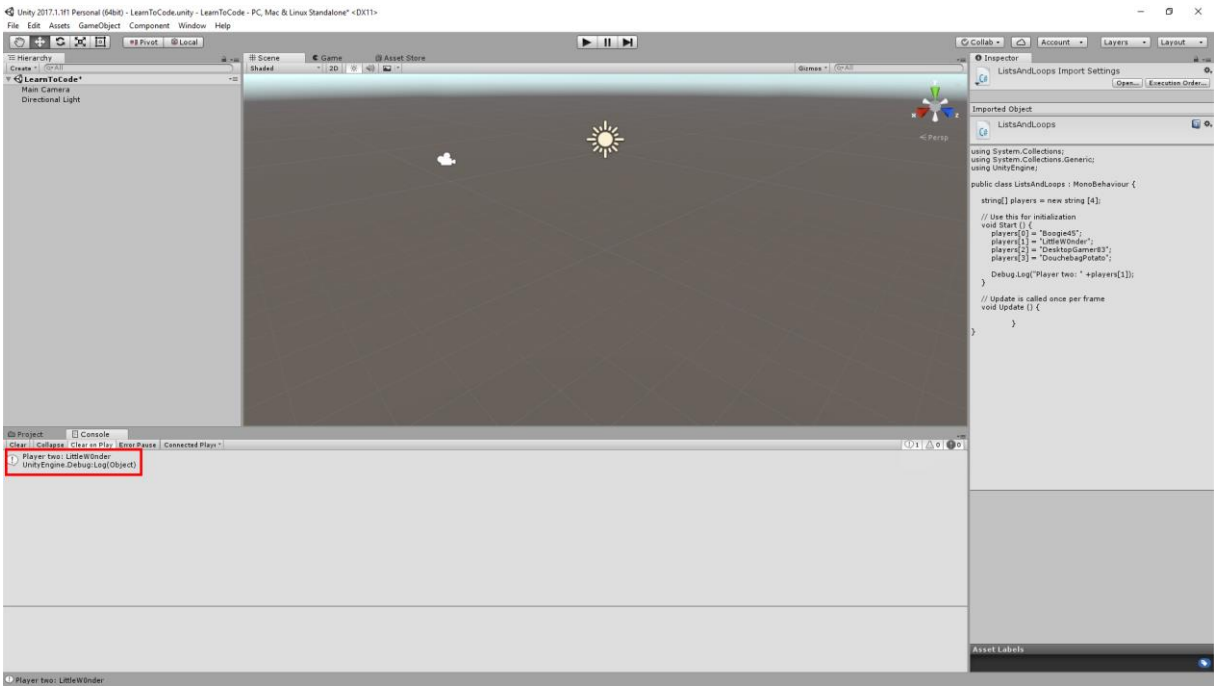
players [3] = «DouchebagPotato»;



Plass 0 i arrayen får verdien «Boogie44», plass 1 får verdien «LittleW0nder» osv.

Husk at arrayer alltid begynner på 0!

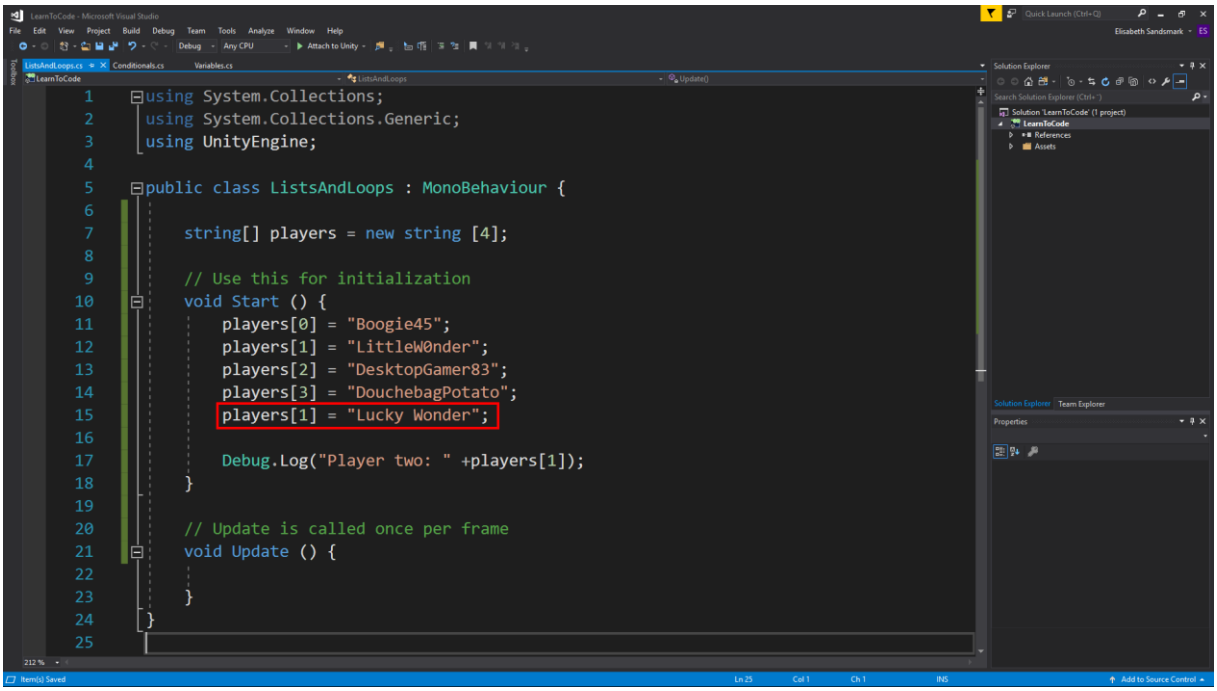
Hvis du kjører koden nå vil det skrives ut navnet på spiller 2 (plass 1 i arrayen).



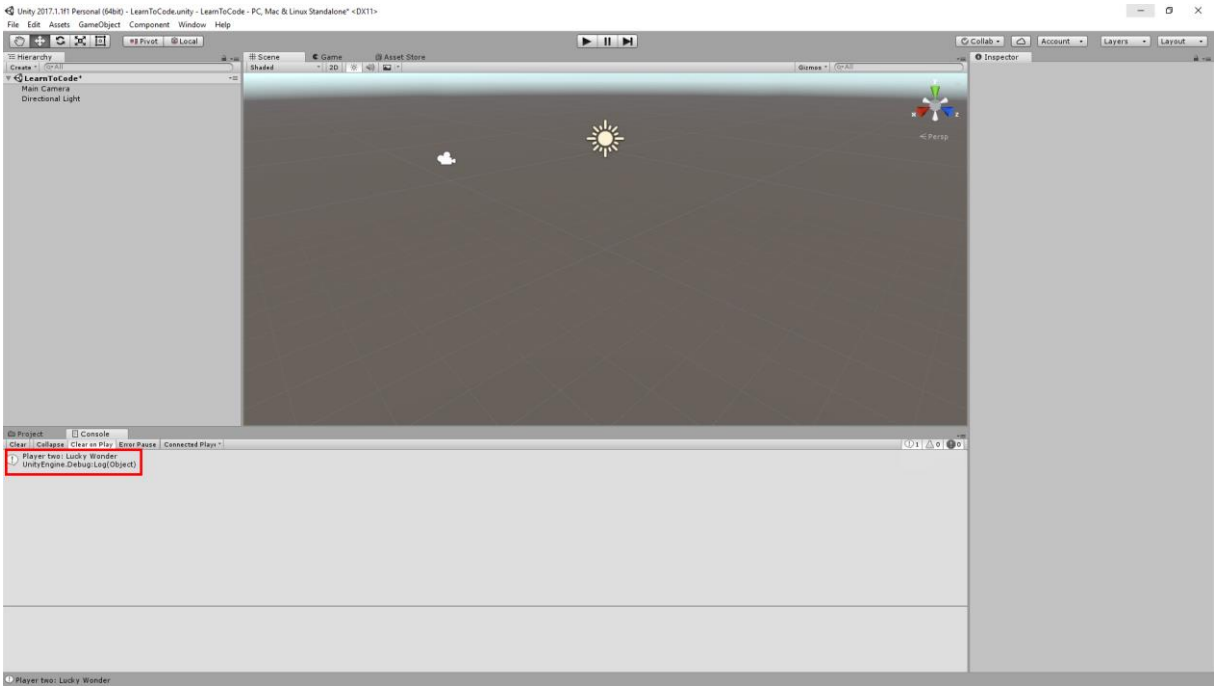
Til nå ser ikke koden noe særlig mindre eller mer oversiktlig ut enn i det første eksempelet vi lagde. Men det vi har gjort er å legge elementene våre i en kolleksjon som vi kan lagre og hente/bruke i f.eks. andre script (mer om dette senere). Dermed slipper vi å gjenta kode om og om igjen.

En annen ting du kan gjøre er å gi verdiene nye navn. F.eks. ønsker kanskje spiller 2 («LittleW0nder») å bytte navn til «Lucky Wonder». Dersom en variabel får tildelt en ny verdi er det denne verdien som gjelder fra da av. Koden leses fra toppen og nedover!

players [1] = «Lucky Wonder»;



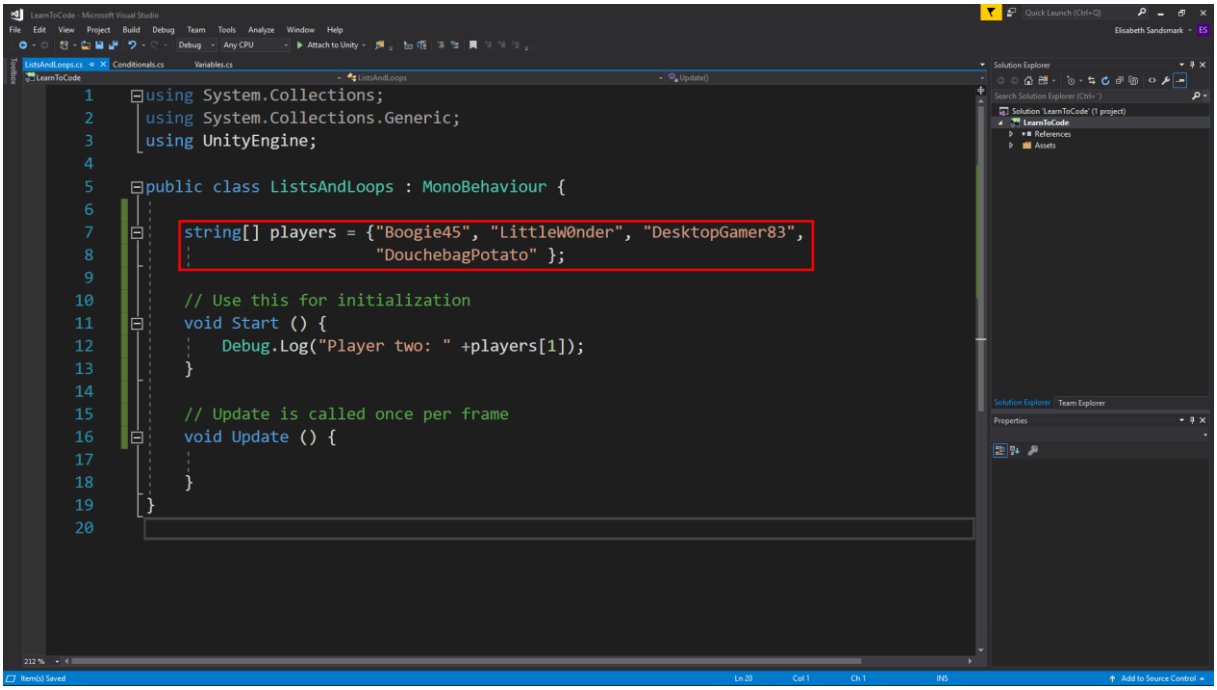
Hvis vi kjører koden nå vil navnet på spiller 2 ha endret seg.



Det finnes også en annen måte å opprette en array på. Denne måten tar mindre plass og ser generelt bedre ut og gjør det samme som i eksempelet tidligere (du oppretter en array med fire elementer og gir hvert av elementene en verdi/navn:

string[] players = { « Boogie45», «LittleW0nder», «DesktopGamer83»,

«DouchebagPotato» };



Legg merke til at det brukes krøllparentes her!

Oppgave

Lag din egen array med minst fem elementer. For eksempel kan du lage en array med navnene på alle i klassen, eller en array med primtallene mellom 1 og 20 osv.

Loops (løkker)

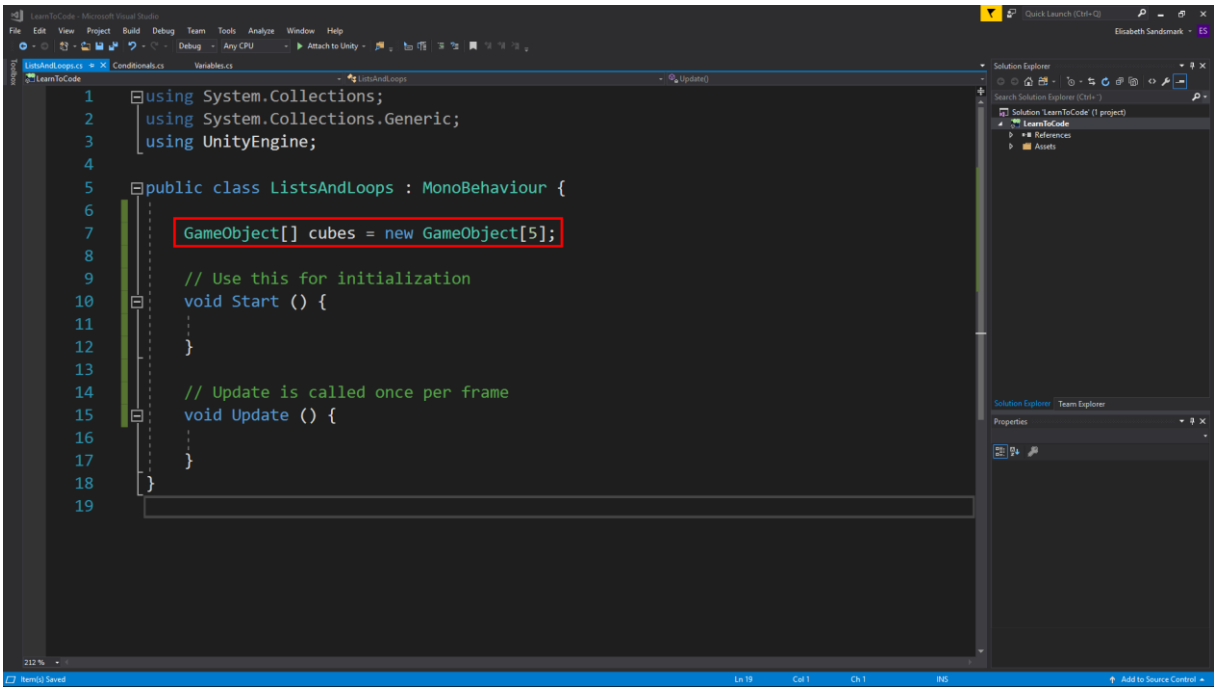
En loop, løkke eller repetisjonskonstruktør består av én eller flere instruksjoner som skal oppfylles gjentatte ganger inntil en eller annen betingelse er oppfylt. Et eksempel på en slik loop er en for-løkke og brukes når vi vet hvor mange ganger vi skal kjøre gjennom løkka. Et annet eksempel er en while-løkke. Vi skal i hovedsak bruke for-løkker og kommer derfor til å fokusere på dette.

Målet vårt i dette eksempelet er å lage 5 kloner av en «cube» som skal vises på skjermen. Disse skal opprettes i en array ved hjelp av en for-løkke. Vi skal også få dem til å flytte x-posisjonen sin ut fra hvilken plass de har i arrayen.

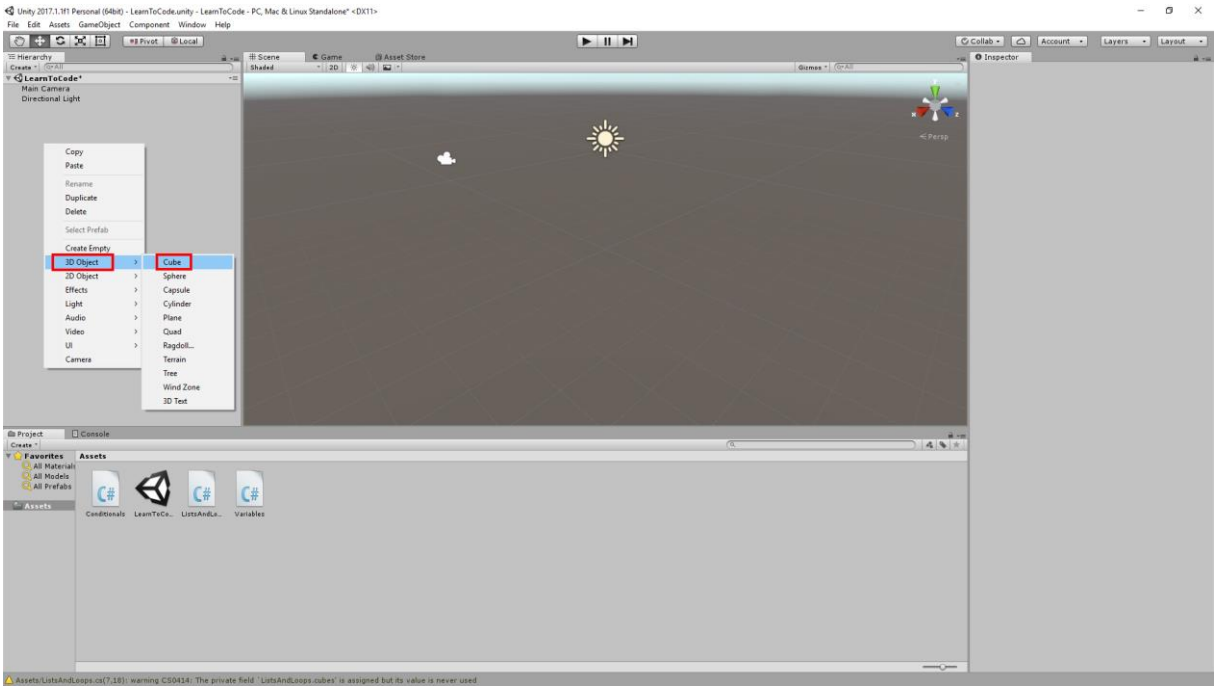
Vi starter med å lage et GameObject og gjør den til en array. Kall den cubes og gi den størrelse 5.

Et GameObject er et objekt som finnes i Unity 3D, et objekt som eksisterer i spillet f.eks. en «cube». (Husk at en array kan være av forskjellige typer, og et GameObject er ikke en int eller en string etc, det er en egen type.)

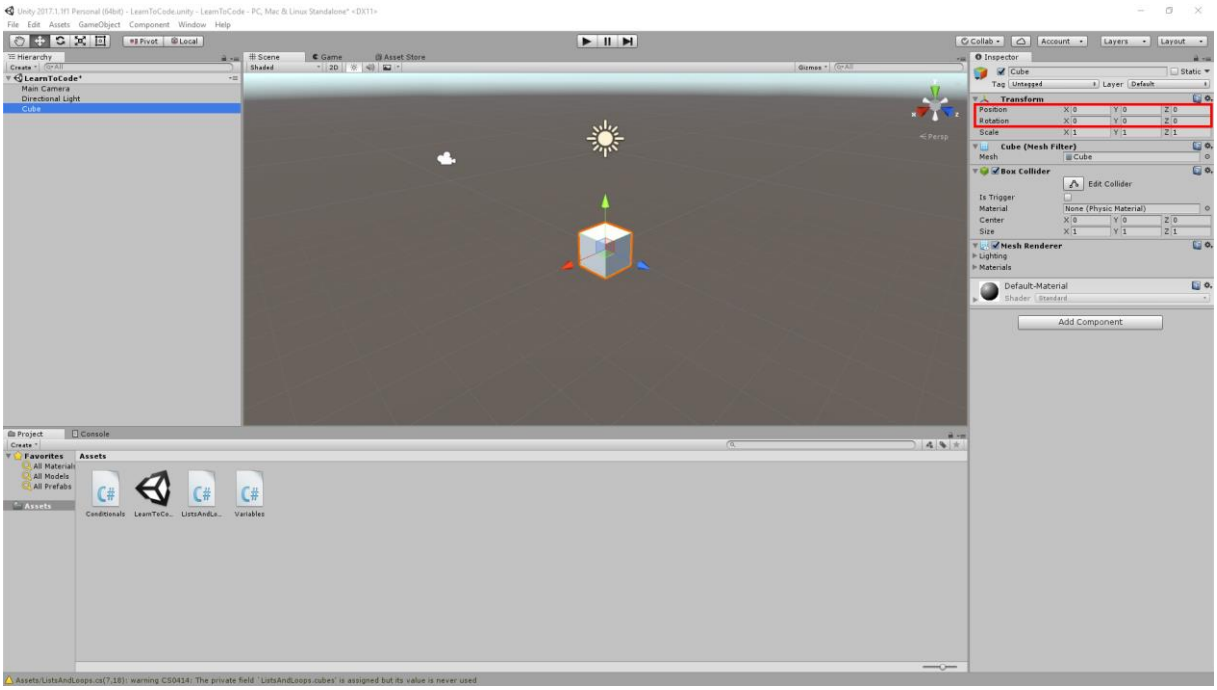
GameObject[] cubes = new GameObject[5];



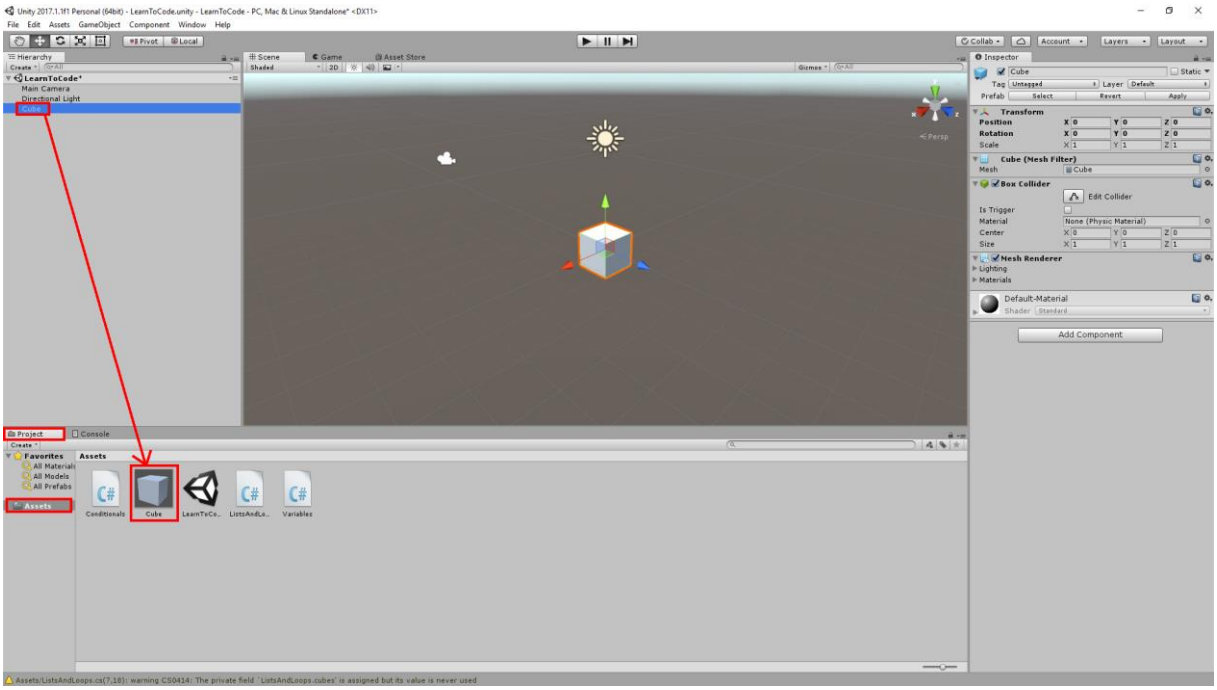
Gå til Unity. Høyreklikk under «Hierarchy» og klikk «3D Object» - «Cube» og lagre.



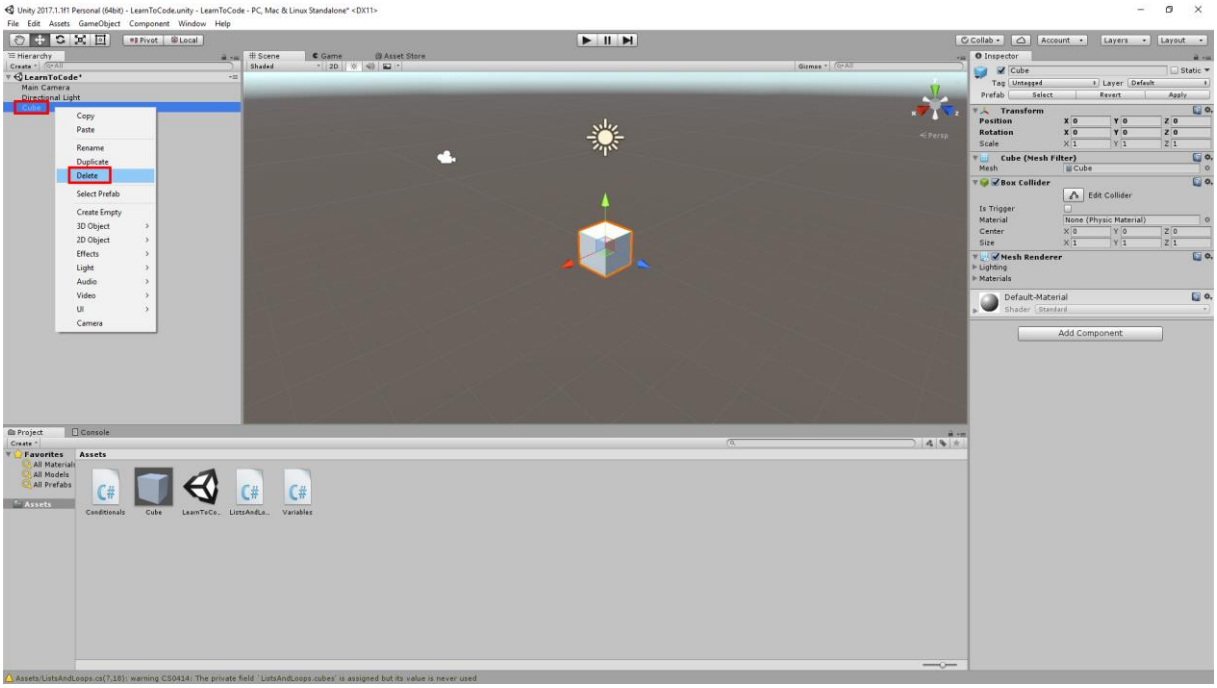
Pass på at alle verdiene til «Position» og «Rotation» under «Inspector» er 0.



Dra «Cube» over til «Assets» mappen din (der scriptene dine ligger). Dette gjør den om til en «Prefab».

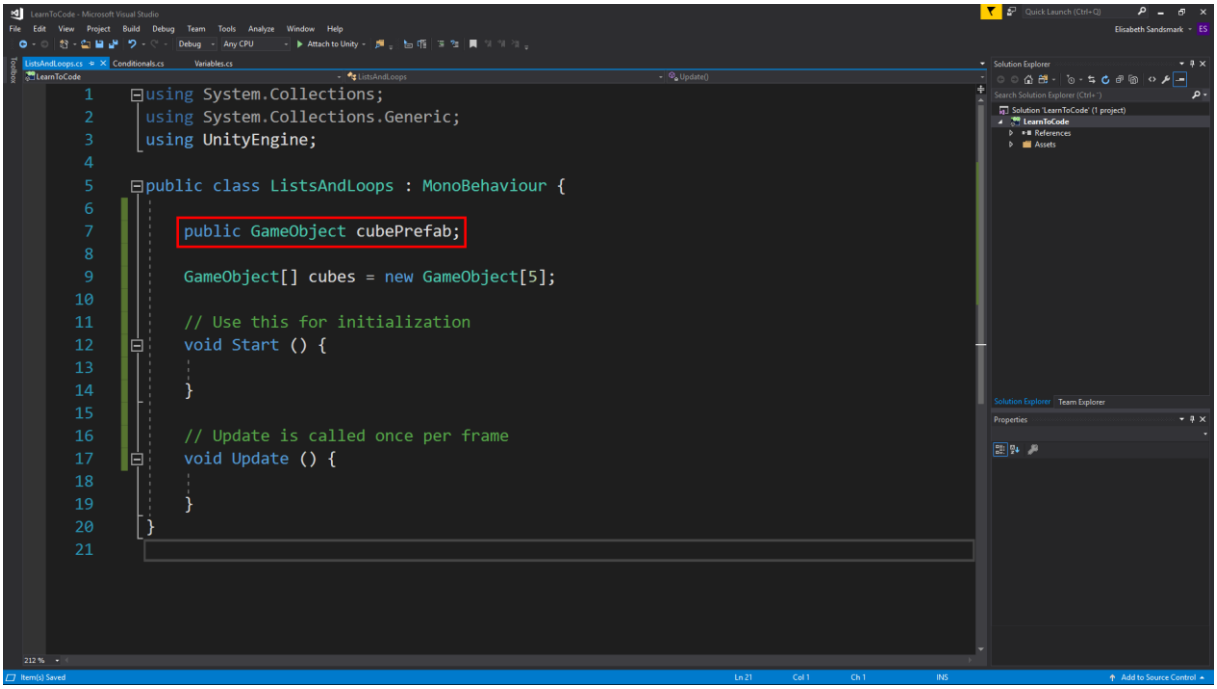


Du kan nå slette selve «Cubes» under «Hierarchy» siden vi har lagret den under «Assets».

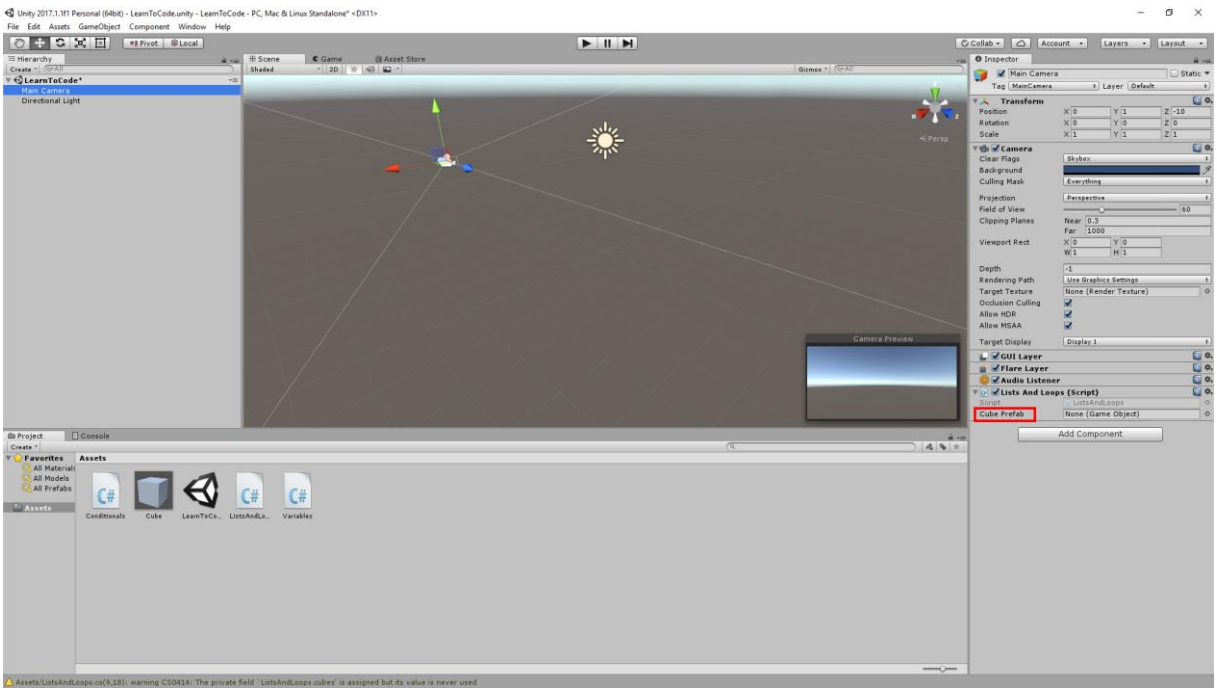


Gå tilbake til koden og lag en public GameObject og kall den «cubePrefab».

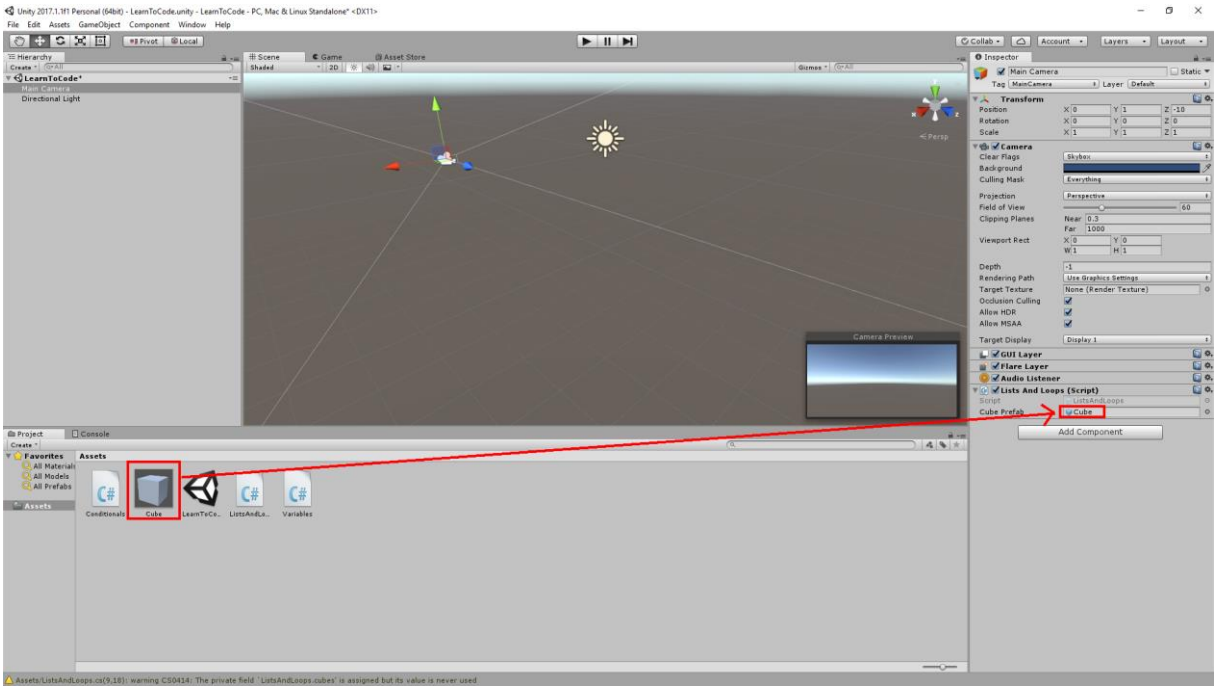
public GameObject cubePrefab;



Grunnen til at vi gjør den public er at vi da kan bruke det i Unity. Hvis vi går tilbake til Unity vil vi nå se at det finnes noe som heter Cube Prefab under Main Camera.



Dra «Cube» fra «Assets» mappen over til «Cube Prefab» som du nettopp opprettet.



Gå tilbake til koden. Det vi ønsker å gjøre er å opprette 5 cubes og lagre dem i en array. Dette kan vi gjøre ved hjelp av en «loop». En loop vil repetere én eller flere instruksjoner gjentatte ganger inntil en eller annen betingelse ikke lenger er oppfylt. Så slipper vi å skrive samme koden om og om igjen! Vi vet at instruksen vår (instruks for å opprette en cube) skal gjentas 5 ganger fordi vi vil ha 5 cubes og vi kan derfor bruke en for-løkke.

En for-løkke skrives på følgende måte:

```
for ( <initialisering>; <test>; <oppdatering> ) {
        <instrukjson(er)>
}
```

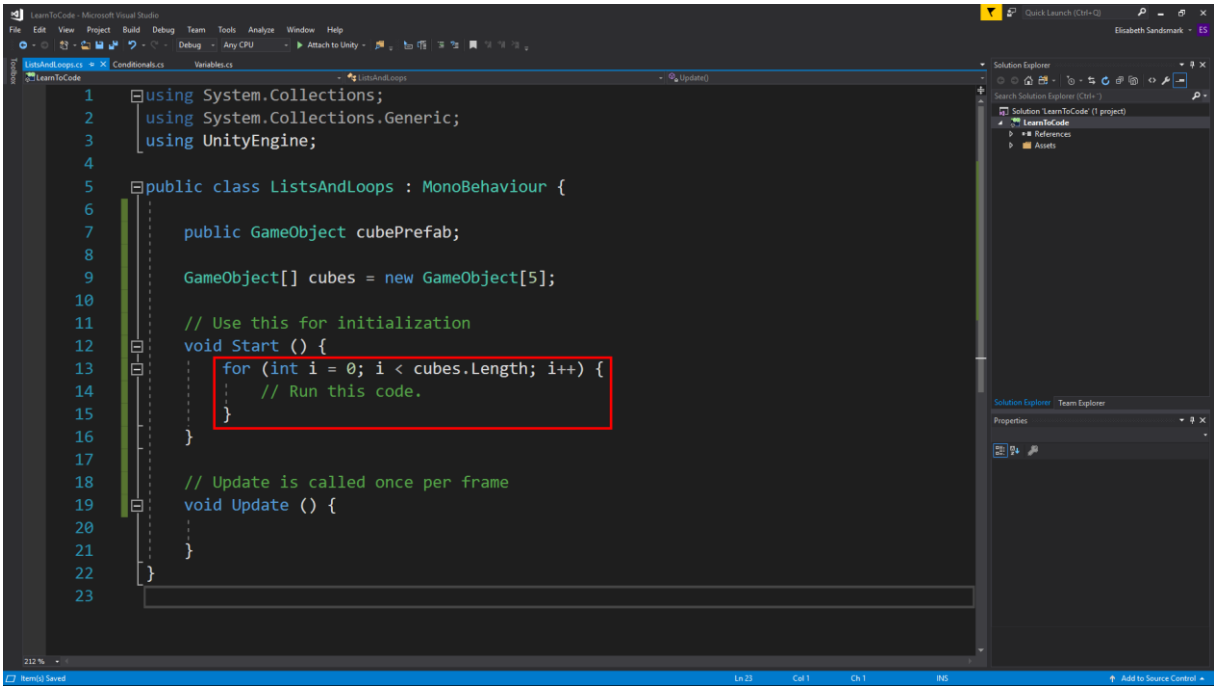
Vi lager en loop som repeterer en instruksjon 5 ganger (for å lage 5 cubes).

```
for (int i = 0; i < cubes.Length; i++) {
        // Run this code.
}
```

int i = 0; : oppretter en variabel av typen int (denne kan vi kalle for hva enn vi ønsker, men det er ofte vanlig å kalle den for «i» eller «j» osv. i små programmer.) og setter startverdien til 0.

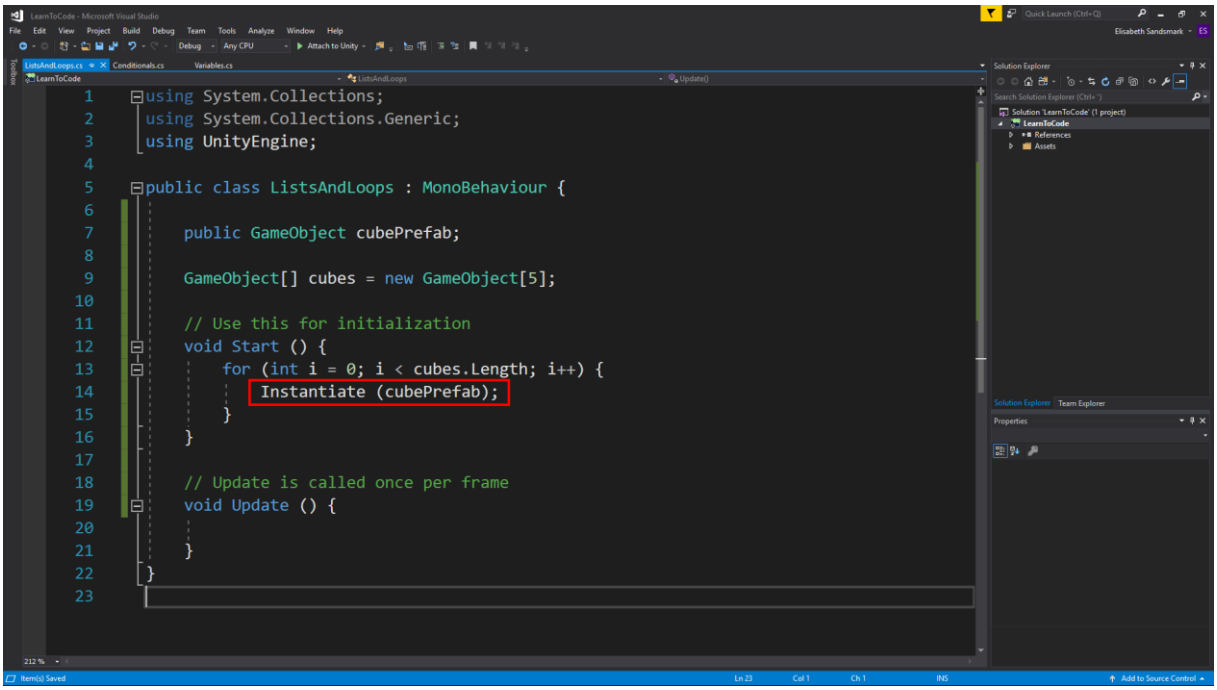
i < cubes.Length; : tester om «i» er mindre enn lengden til vår array for cubes (som vi har satt til 5). Så lenge «i» er mindre enn 5 (altså 4. Husk at en array alltid starter på 0. Vi må derfor bruke mindre enn tegnet«<» og ikke er lik tegnet «=») vil instruksjonene innenfor for-løkken bli utført. Når «i» har nådd 5 vil ikke for-løkken lenger kjøre – den har oppnådd sitt mål.

i++ : øker verdien til «i» med 1 hver gang instruksjonene til for-løkken utføres (hver gang vi legger til en cube i vår array).



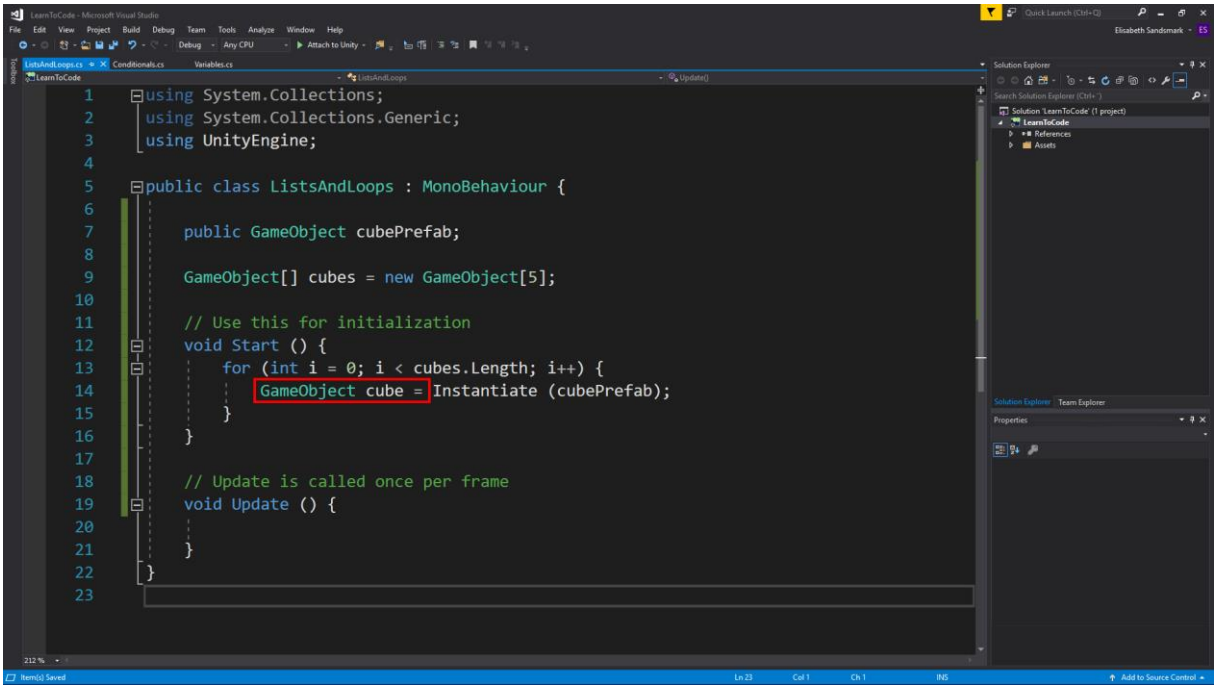
La oss legge til instruksjonene som skal utføres hver gang for-løkka kjøres. Vi begynner med å legge til en Instantiate (mer om dette senere, men kort fortalt gjør det at vi kan klone et objekt og bruke klonene i «Scene» i Unity).

```
for (int i = 0; i < cubes.Length; i++) {
        Instantiate (cubePrefab);
}
```



Vi ønsker å lagre dette i en variabel og gjør dette på følgende måte:

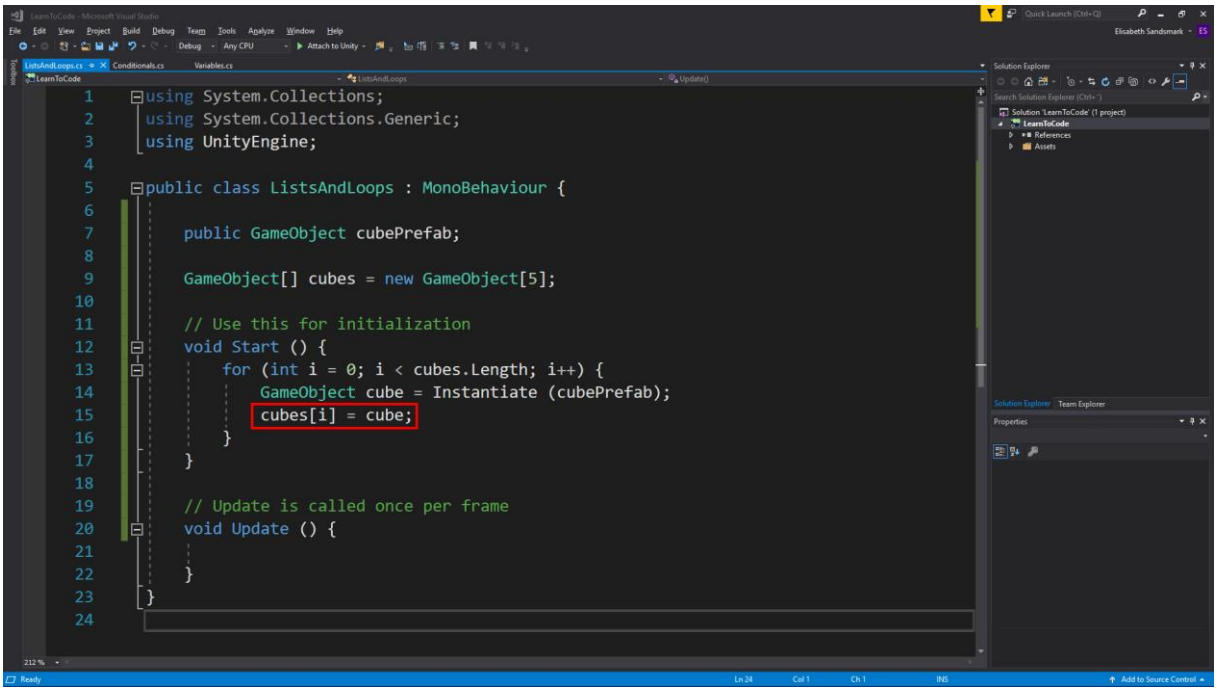
```
for (int i = 0; i < cubes.Length; i++) {
        GameObject cube = Instantiate (cubePrefab);
}
```



Og lagrer så alt i vår array «cubes[]».

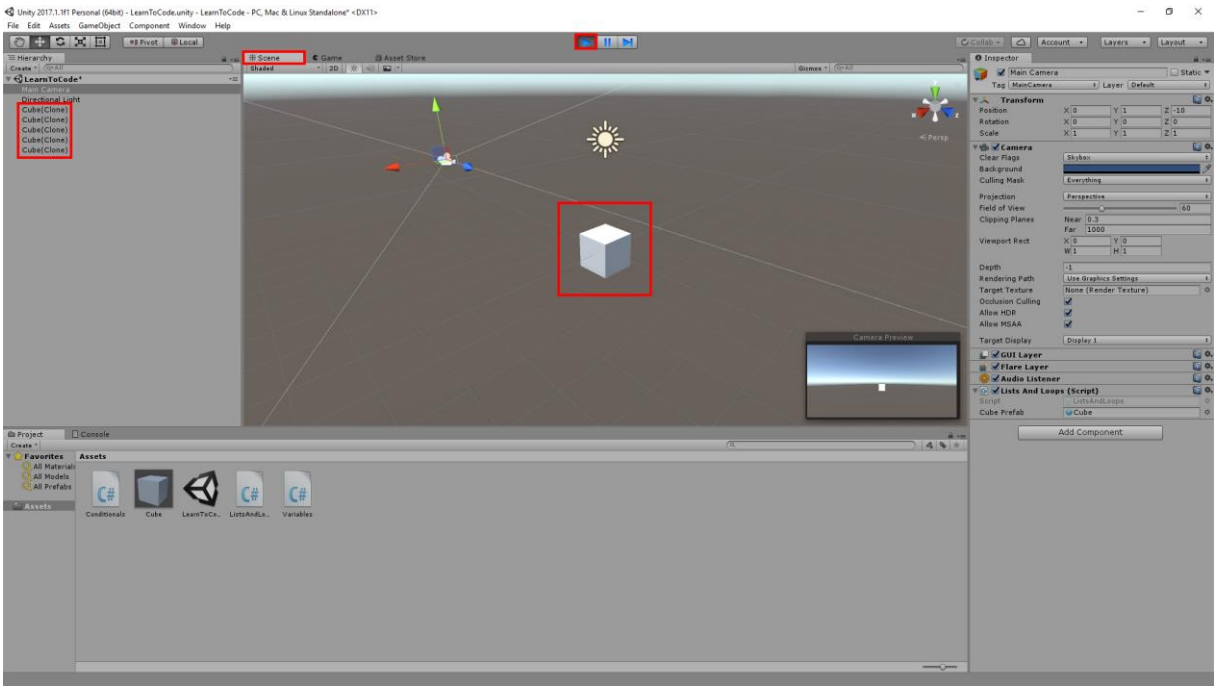
(Vi skriver om verdien til hver plass i arrayen vår (hver gang vi kjører gjennom for-løkka legges en ny «cube» til i arrayen og «i» økes med 1, og vi går til neste plass i vår array – vi starter på 0 og ender på 4 og ender totalt opp med 5 cubes).)

```
for (int i = 0; i < cubes.Length; i++) {
        GameObject cube = Instantiate (cubePrefab);
        cubes [i] = cube;
}
```

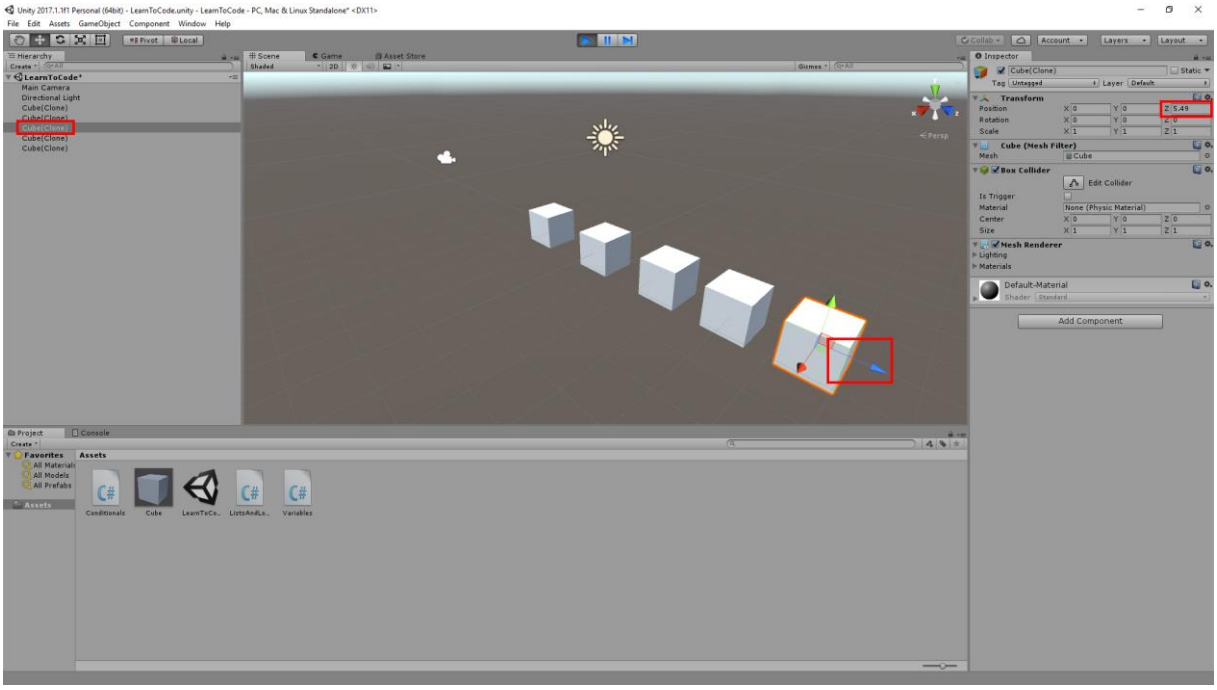


Lagre og kjør koden og se hva som skjer.

Det opprettes 5 kloner av «cube», men de er for øyeblikket helt identiske og befinner seg på samme punkt under «Scene».



Du kan f.eks. trykke på hver enkelt av klonene og flytte på dem (blå pil for å endre posisjon z, rød pil for å endre posisjon x) for å se om du har 5 kloner.



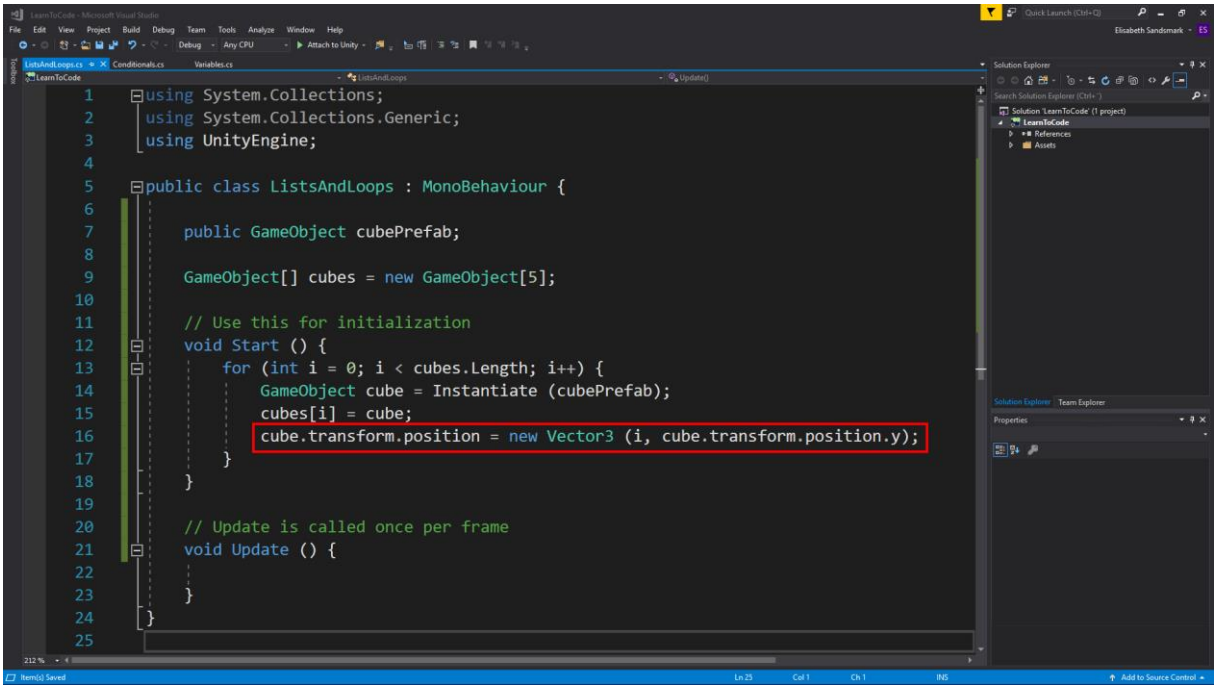
La oss endre posisjonen til hver enkelt cube i koden vår. Vi bruker .transform.position og lager en new Vector3 for å gjøre dette. En Vector3 lagrer x, y og z kordinater til et objekt og brukes for å flytte rundt på objekter.

Vi ønsker å flytte hver cube langs x-aksen og å basere posisjonen til x på antall cubes i arrayen. Så sett første verdi (x-posisjonen) til «i» (husk at «i» økes med 1 hver gang vi kjører gjennom loopen, så x-posisjonen til første cube vil være 0, x-posisjonen til andre cube vil være 1 osv.).

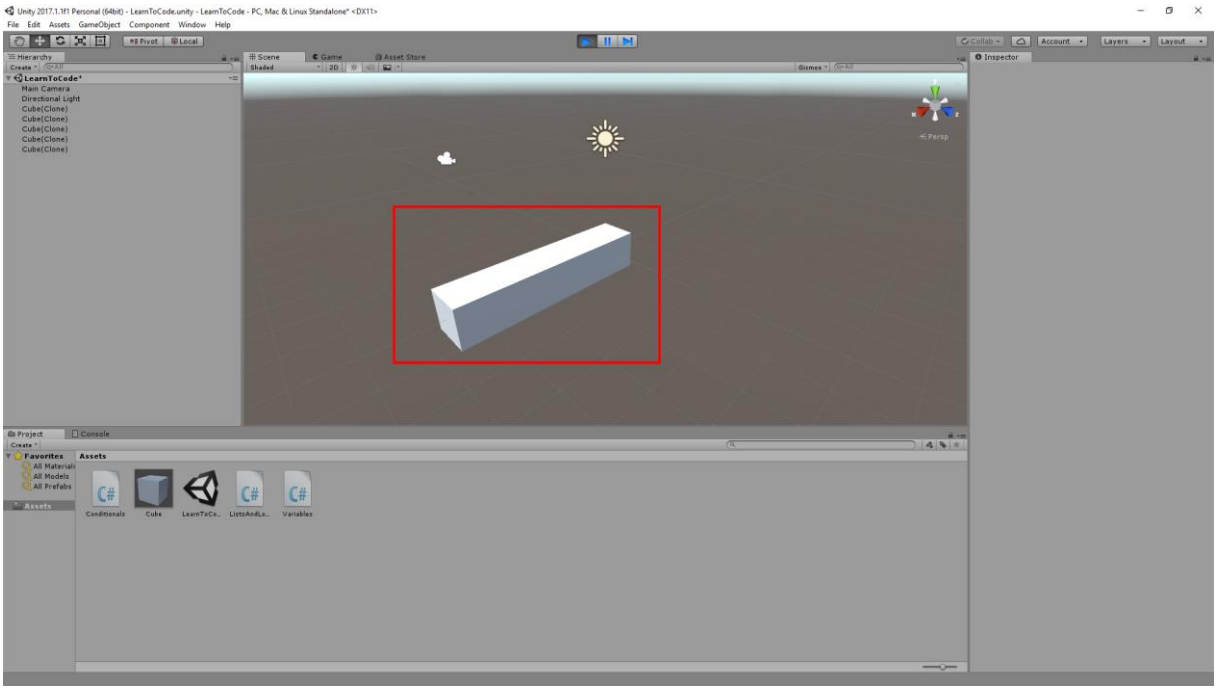
Vi ønsker at y-posisjonen skal være lik for alle cubes, og bruker.transform.position.y for å gjøre dette.

Vi setter ikke noen verdi for z. Dette gjør at z blir samme for alle cubes (slik som y er lik for alle).

```
for (int i = 0; i < cubes.Length; i++) {
        GameObject cube = Instantiate (cubePrefab);
        cubes [i] = cube;
        cube.transform.position = new Vector3 (i, cube.transform.position.y);
}
```

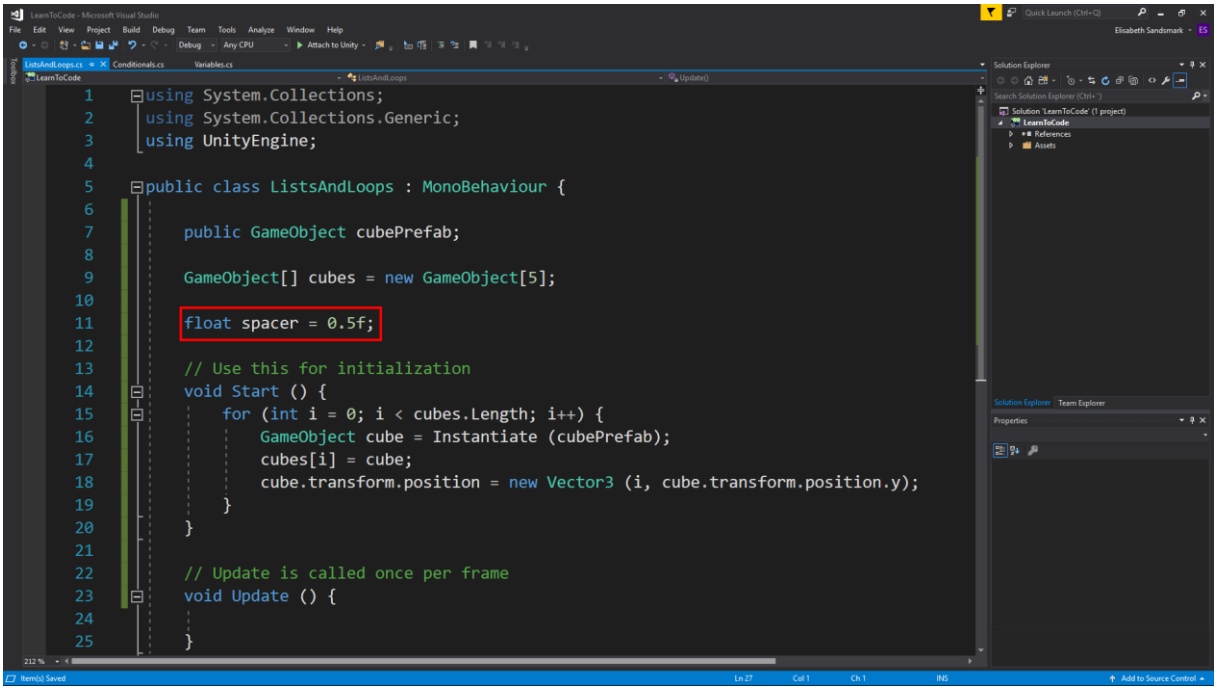


Vi får følgende resultat:



Kanskje ønsker vi litt mer mellomrom mellom våre cubes. Vi kan gjøre dette ved å plusse på en mengde på «i». Lag en variabel av typen float og kall den f.eks. «spacer» og gi den verdien 0.5f

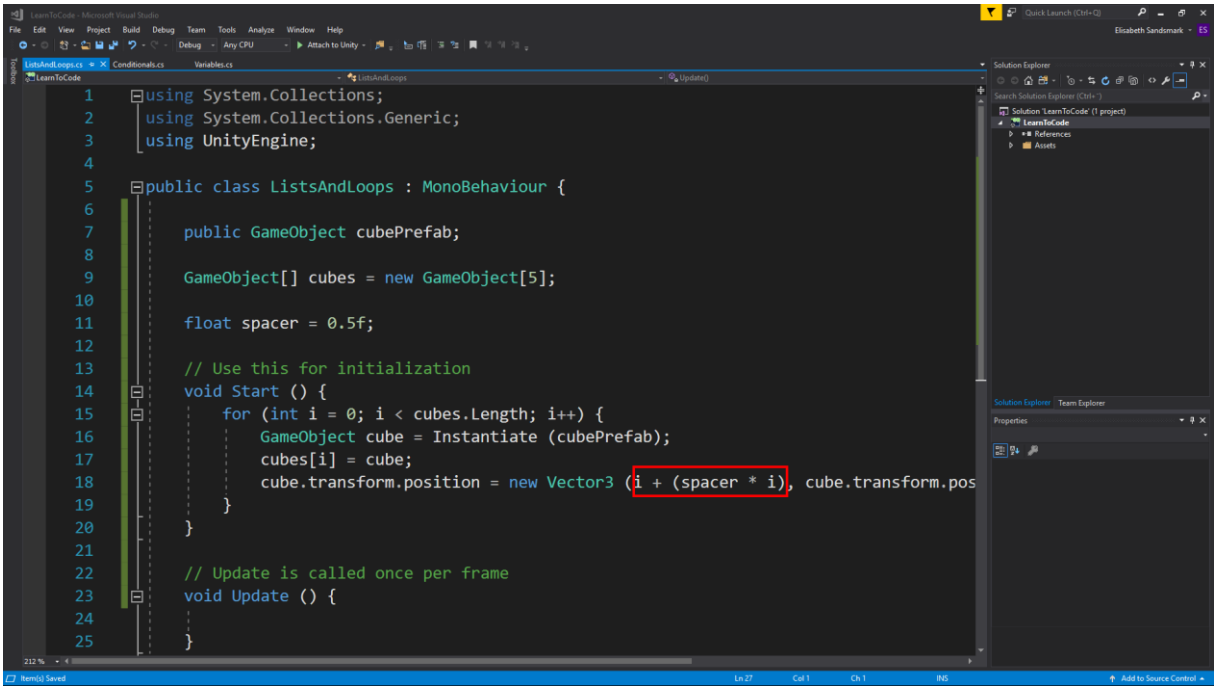
float spacer = 0.5f;



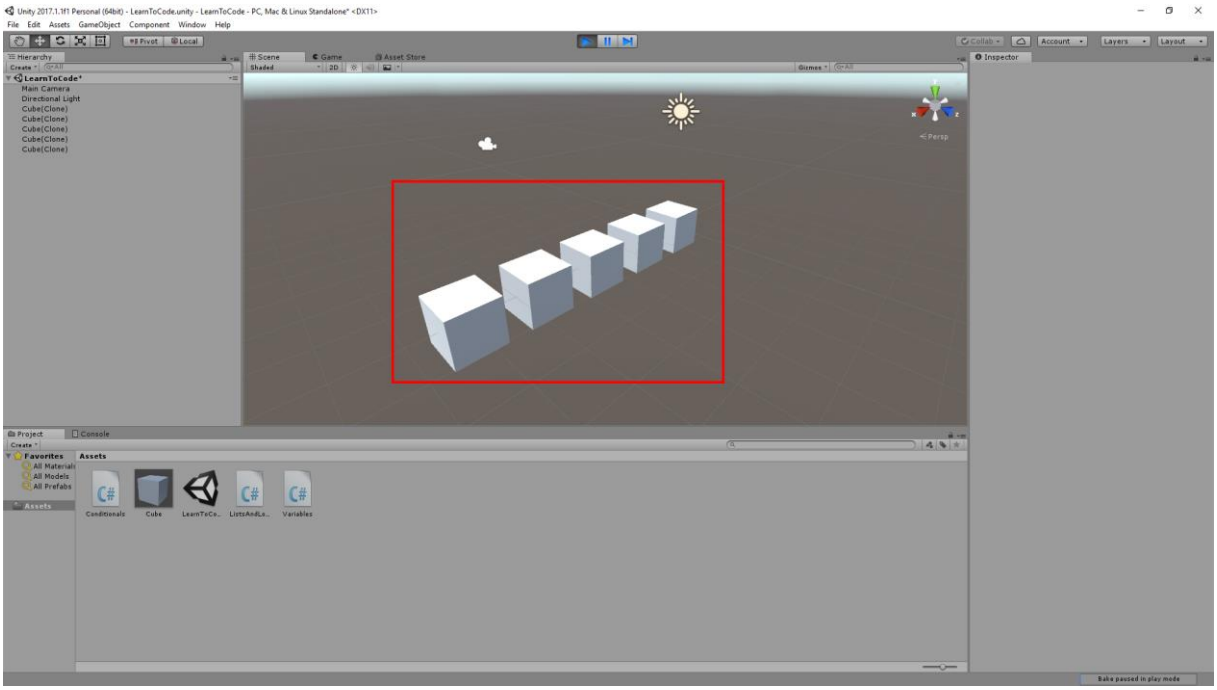
Pluss sammen variabelen «spacer» med «i».

Ved å legge til «spacer» i parentes og gange med «i» forhindrer vi at «spacer» blir tatt i bruk dersom posisjonen til x allerede er 0. (i + (spacer * i)

```
for (int i = 0; i < cubes.Length; i++) {
        GameObject cube = Instantiate (cubePrefab);
        cubes [i] = cube;
        cube.transform.position = new Vector3 (i + (spacer * i),
                                               cube.transform.position.y);
}
```



Vi får følgende resultat:



Ikke fortvil om du ikke forstår alt som skjer. Det tar **tid og øvelse** å forstå både matten og logikken bak koding. Men prøv å lese gjennom alt i dette dokumentet til du forstår hva en array og hva en loop er.

Oppgave

Lag et program som beregner summen for alle tallene fra 1 til 99. (1 + 2 + 3 + 4 + … + 99).

Løsning:

PRØV Å LØSE OPPGAVEN FØR DU SER PÅ FASITEN!

// Lager en variabel som inneholder summen og setter startverdien til 0.

```
int sum = 0;
```

// Lager en for-loop som gaar gjennom hvert tall fra 1 til 99 og legger dem til totalsummen.

```
for (int i = 1; i < 100; i++){
    sum += i;
}
```

// Skriver ut totalsummen.

```
Debug.Log(sum);
```

Følgende skritt utføres in den rekkefølgen som er angitt:

1. Initialiseringen utføres først og kun én gang. Variabler som deklareres ved initialiseringen (tellevariablen «i» i eksempelet), vil bli lokale variabler til for-løkka (de vil ikke eksister utenfor den).
2. Løkke-intsruksjonen(e) utføres så sant testen (i eksempelet i < 100) gir true/sann verdi.
3. Oppdatering av tellevariablen (i eksempelet i++) utføres til slutt.
4. Så utføres testen på nytt for å sjekke om den gir true/sann verdi igjen osv.
5. Programkontrolløren hopper ut av for-løkka så snart testen gir false.

I oppgaven bil løkka bli gjennomført når tellevariablen «i» har verdier lik 1, 2, 3, 4, … ,99. Altså i alt 99 ganger. For hvert løkkegjennomløp vil den nye verdien til «i» bli addert til variabelen sum. (Merk at vi foran løkka må passe på å gi sum en riktig startverdi!) Når «i» er blitt lik 100, vil testen gi false/usann verdi. Da vil programmet hoppe ut av løkka. Verdien 100 vil derfor ikke bli addert til summen. Resultatet blir at vi får beregnet akkurat den summen vi ønsket. Totalt skal summen bli 4950.

Kort oppsummert

En array er en éndimensjonal tabell eller en liste og består av et bestemt antall dataelementer av samme type. Elementene er indeksert fra 0 og oppover. Arrayer består av ulike typer variabler som f.eks. int, boolean, string osv.

En loop består av én eller flere instruksjoner som skal utføres gjentatte ganger inntil en eller annen betingelse ikke lenger er oppfylt. Et eksempel på en loop er for-løkke, og brukes dersom vi ven hvor mange ganger en loop skal gjenta seg.