ARBEIDSKRAV

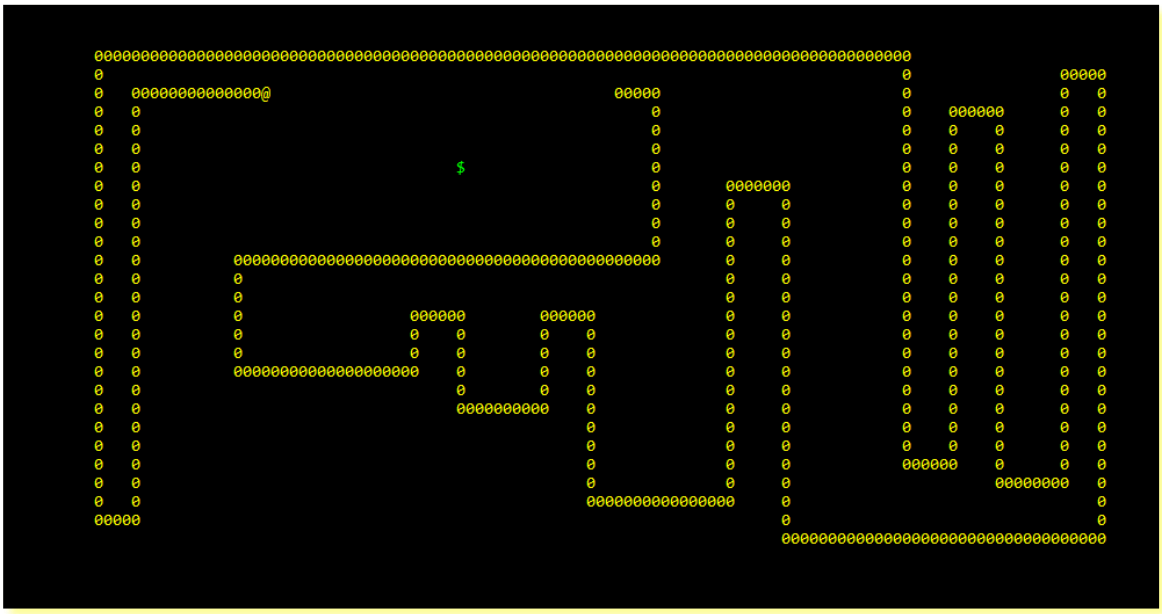
15.10.17

UML og refaktorering SnakeMess

- Markus Kristiansen Nyland - Michael Fredriksen - Vilde Solvoll -

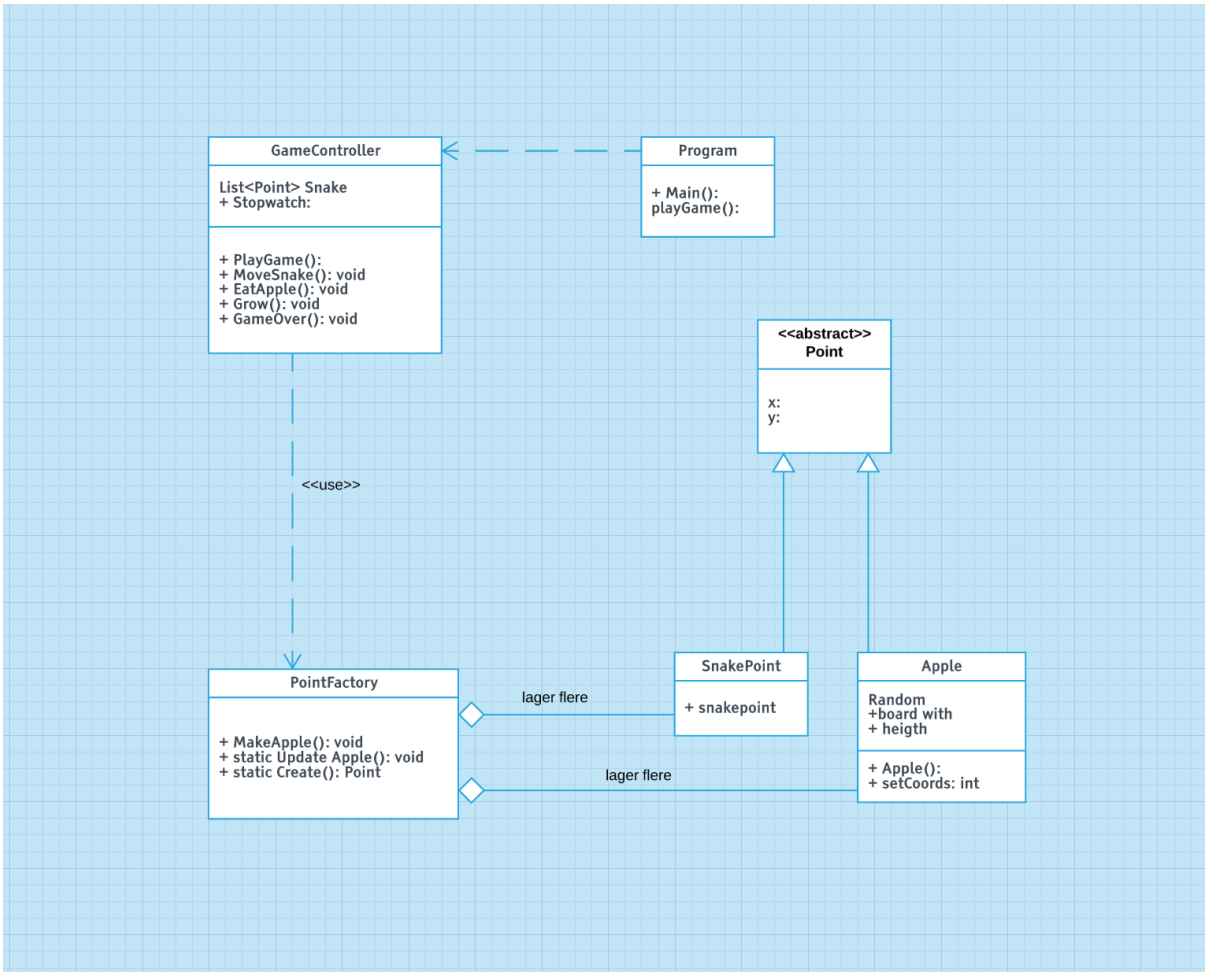
Innholdsfortegnelse

Spillet Snake



Spillet er bygget opp av punkter innenfor et definert område i konsollen. Punktene utgjør både snake selv, som er av typen List, og eplene Snake spiser. Snake kan bevege seg rundt på brettet ved at punkter opprettes og sletter punkter etter hverandre. Etter hvert som Snake flytter seg i x - eller y - retning, genereres det epler, ett nytt eple for hvert som blir spist. Snake kan ikke gå i motsatt retning av den retningen den er på vei. Dersom et eple blir spist, skal Snake vokse ett punkt, altså ett tegn større. Slik fortsetter spillet til Snake treffer seg selv eller veggene til området av spillbrettet. Spillet vinnes ved å ikke ha flere ledige plasser å sette et nytt eple på, da spillebrettet er fylt av kroppen til Snake.

Implementation Class Diagram



Prosessen

Kommentering av opprinnelig kode: Vi begynte med å kommentere den opprinnelige koden, slik at vi var sikre på at vi forsto hvordan programmet fungerte. Deretter byttet vi alle misvisende navn i koden slik at arbeidet videre med den skulle bli enklere.

Modellere ny løsning: Da koden var så ryddig og forståelig som mulig, modellerte vi en ny løsning slik vi tenkte ville bli mest hensiktsmessig med hensyn til GRASP prinsippene. Denne delen tok mye tid. Vi måtte veie prinsippene opp mot hverandre, spesielt punktene *low coupling* og *high cohesion*. Det ene prinsippet påvirker det andre.

Skrive den nye koden: Med et Implementation Class Diagram klart, begynte vi å klippe og lime inn den opprinnelige koden i de nye klassene. Vi opprettet klassene SnakePoint og Apple, som skulle arve fra Point. Deretter lagde vi klassen PointFactory som skal generere nye points. Da det oppsto buggs underveis, debugget vi koden for å se hvor feilen lå. Vi klarte til slutt å kjøre koden slik at spillet fungerte slik det opprinnelige spillet gjorde.

Spille Snake: Da vi fikk den nye koden til å fungere, satt vi oss til å spille spillet.

Refleksjoner

Hovedfokuset for vår løsning har vært å oppnå *low coupling*. Dette oppnås ved at GameController har et stort ansvarsområde. Det styrer alt fra å holde orden på ticks, til å kjøre sjekk av epler som blir spist og å printe alt på skjermen. Dette er oppgaver vi kunne delt ut til forskjellige klasser. Vi endte opp med en klasse som minner litt om den opprinnelige SnakeMess klassen pga dette.

Noe av det vi klarte å dra ut var opprettelsen av nye Points. Ved å lage en PointFactory løsnet vi på koblingen mellom både Apples og SnakePoint til Controlleren vår. Point factory hjelper også til med high cohesion, ved at vi delegerer opprettelse av Points til en egen klasse.

Ellers har vi sett på gode metode og variabelnavn for å gjøre koden mer lesbar. Hadde vi fordelt oppgavene på andre klasser hadde vi oppnådd ”higher” cohesion.