Noen spørsmål

Position.move tar inn en *direction* og *distance* og hvis distansen ≠ 0 tar den *.getMovement* av retningen noe som isolerer ut x- og y koordinatene, vi ender da opp med en et *Position* objekt.

Position klassen i lab5 har ikke *equalsTo* metoden som brukes til å sjekke om to posisjoner er like. I lab5 brukes heller ikke *hashCode* metoden. Når man har en stor mengde data kan det med fordel lagres i hashmap for mer effektiv håndtering.

Lab5 har heller ikke *isZero* eller *toString* metodene.

AbstractMovingObject ekstender *AbstractSimObject* og er dermed en underklasse av denne. Bevegende objekter har feltvariabelen *speed* og metoden *accelerateTo*.

Posisjonene arves av underklasser. Instanser av disse klassene får sine egne unike posisjoner og endres på dette (instanse) nivået.

Metoden *distanceTo* og *directionTo* finner avstanden og retningen fra en posisjon til en annen.

Hvis vi satt speed til å være private måtte vi brukt setters og getters for å kunne arve den og endre på den. Dette kunne vært fordelaktig mpt. f.eks. restriksjoner. Da kunne vi for eksempel endret hastighet kun dersom den nye hastigheten møtte spesifikke krav. Vi bruker isedet protected når vi vil arve feltvariable. AbstractSimObject har et *getDir* metode som finner retningen men ikke *setDir*. Vi kan implmentere en *setDir* metode og evt hatt begrensinger for hva som er lov. Vi kan kan også endre retningen ved å referere til en instanse av klassen slik vi gjør nå.

Å endre posisjonen er noe som skjer for hver enkelt instanse av en klasse. Det er best å unngå public metoder for tilfeller hvor endringen kun gjelder et spesifikt objekt fordi vi vil ikke at metoden skal kunne kalles offentlig

1.7 - spørsmål

Vi kunne i prinsippet unngått grensesnitt og hatt færre .java filer, og i enkle tilfeller kunne det tenkes at dette var en fordel. Grunnen til at vi gjør det slik er at vi vil gjøre koden gjenbrukbar og generell. Dvs. vi kan lett ekspandere koden og har mer kontroll. F.eks. hvis vi ville lage forksjellige SimAnimal med forskjellige egenskaper kan vi fortsatt arve metoder, override noen, og lage nye metoder for nye dyr. For større programmer er det lurt å bruke grensesnitt. Vi vil også unngå å duplisere kode, dvs. skrive samme koden i flere klasser. F.eks. vi lager en **FoodComparator.java** klasse. Vi vil ikke ha denne som anonym klasse dersom

vi har mange forskjellige varianter av SimAminal.java (f.eks SimAnimal2, SimAnimal3 etc) da det fører til at samme koden blir skrevet om igjen. Samme gjelder som grensesnittene. For større programmer er det ikke fleksibelt å ikke bruke grensesnitt. Hvis du skulle endre en overordnet egenskap måtte denne endres i all klasser hvor den brukes. Om man bruker grensesnitt måtte man bare endret egenskapen i den overordnede klassen.

Del II

Tilfeldighet

Først vil jeg implementere litt tilfeldighet for SimAnimal:
Tilfeldig synsvinkel fra 80 til 110 grader.
Tilfeldig synslengde mellom 300 og 500.
Tilfeldig hastighet når dyrene "løper/svømmer" fort mellom 3 og 5.

Tilfeldig hastigeheter sørger også for mindre opphoping av dyr. Om alle har lik hastighet (og andre variable) vil de hope seg opp på ett punkt og gå etter eksakt samme mat hele tiden.

Animal State med Enumeration

Jeg ønsker å implementere enum state (i **AnimalState.java**) for SimAnimal (og rovdyret SimCarnivor) for forskjellige tilstander dyrene opplever

ALIVE - når dyret level i beste velgående
STARVING - når dyret har for lavt energi går det i survival mode og søker mat mer effektivt ved at sysnfeltet blir bedre (bredere og lengre) og hastigheten øker.
RECOVER - når dyret har sultet men har funnet nok mat går det i recover state. Her går sansene og hastigheten tilbake til normal.
CHASING - egenskap for rovdyr. Når rovdyret har fått et mål i sikte vil det sette kurs mot dette og øke hastigheten
EATING - egenskap for rovdyr. Når et byttedyr er innen rekkevidde starter rovdyret å spise på det. Da går hastigheten ned men energien blir fornyet.
DEAD - tilstanden rett før this.destroy();

Rovdyr - SimCarnivor.java

Jeg vil så implementere et rovdyr (**SimCarnivor.java**) basert på SimAnimal klassen. Rovdyret går alltid etter nærmeste SimAnimal innenfor synsfeltet. Jeg har gjort SimAnimal til å implementere Edible slik at de kan bli spist av rovdyrene.

Rovdyret vil alltid scanne etter nærmeste SimAnimal og starte CHASE om det er i nærheten. Om et dyr går tom for energi dør det. Når et byttet blir sett starter

CHASE. Når byttet er innen rekkevidde starter rovdyret å spise. Da vil byttet miste energi mens og begge dyrene blir sakket ned. Byttedyret vil starte å løpe vekk fra rovdyret om det kommer for nærme (slik som det unnviker repellants). Byttet har også en 50% sjanse for å få en hastighetsboost (10x) når det er en viss avstand (200) fra rovdyret. Dette simulerer at byttedyrene klarer å rømme vekk og sprinte fra rovdyret. Rovdyret drenerer byttet for energi mens det spiser det, om byttet når 0 i energi (pga rovdyret) blir det spist og råvdyret får 20+ energi (med max 100).

Rovdyret er en Pacman figur.
https://commons.wikimedia.org/wiki/File:Pacman.svg
For å få til animasjon satt jeg en counter som øker for vært step. Verdien på denne coutneren trigger bytte mellom to bilder. Byttet skjer oftere når rovdyret er i tilstand CHASING eller STARVING.

Kjønn og parring

Jeg implementerer så kjønn for SimAnimal. Når et nytt SimAnimal blir laget får det et tilfeldig kjønn. Lager også en mulighet for parring. SimAnimal som ikke er feminine vil oppsøke det motsatte kjønn om det er under 400 unna. Dersom det er en av motsatt kjønn i umiddelbar nærhet, og begge parter har energi >80 og en timer har telt ned, så kan dyrene parre seg. Når dette skjer skapet det et nytt SimAnimal der paringen skjedde. Parringstimeren blir resatt og det må gå minst 30 sekunder før neste parring kan skje for feminint kjønn og 15 sekunder for maskulin kjønn. Når et nytt dyr blir "født" blir tellerene nullstilt slik at vi ikke får pluttselig eksonensiell vekst og inavl.

Sult og død

Det siste jeg implementerer er at dyr som sulter i hjel dør og blir til kadavre. Pacman er som kjent åtseleter og vil gladlig spise kadavre også. Dersom et kadaver ligger i mer enn 15 sekunder forsvinner det.

Tester

Ekstra tester er gjort i **AnimalStatesTest.java**