



BioSim Project

Amir Arfan, Sebastian Becker

INNHold



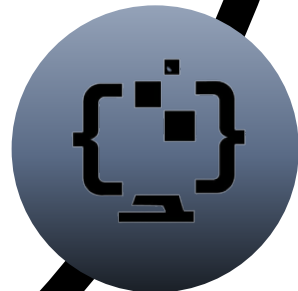
FREMGANGSMÅTE



STRUKTUR



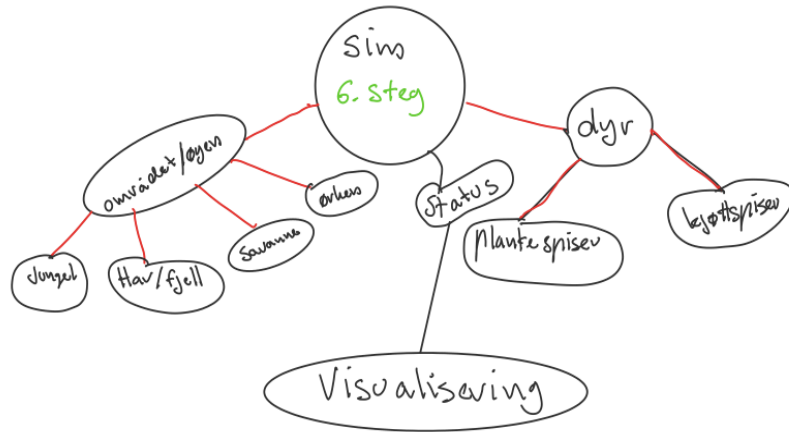
TESTING OG OPTIMALISERING



TILLEGGSFUNKSJONER

Et godt prosjekt begynner med planlegging!

klasser:



class geografi

↳ Input: Map

↳ Methods

- genrevet

class ocean and mountain

class desert

↳ Bool

class savannah

↳ Input: Fmax

class Jungel

↳ Input: Fmax

class Animal:

↳ Input: Tom

↳ Methods:

- Birth
- Fitness
- Weight
- Spise/
- Age
- Migration
- Death

Felles
funksjoner
for begge
typene

- class Herbivore (Animal)

↳ Input: Parameter,
location, age,
weight.

Arver
fra

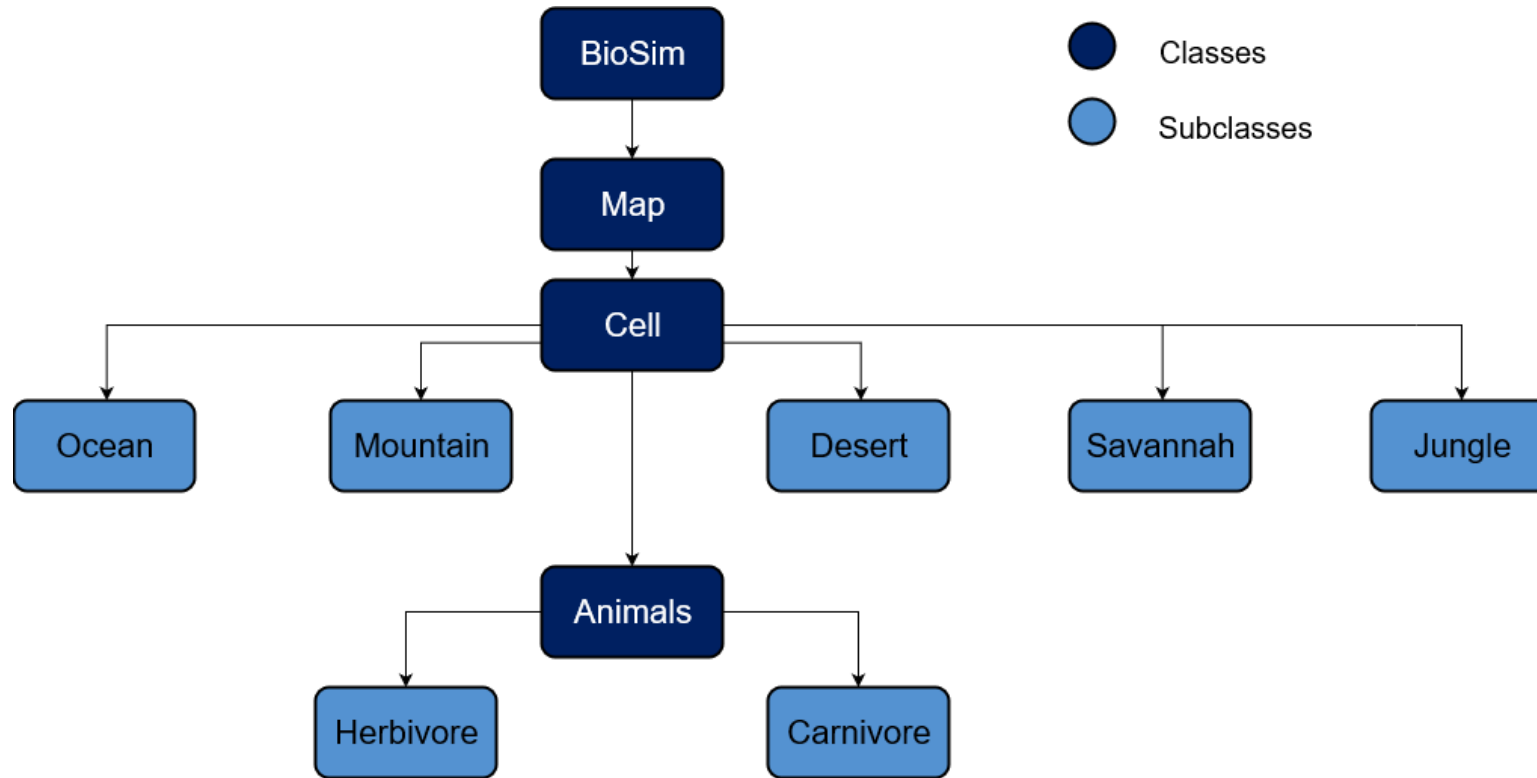
- class Carnivore (Animal)

↳ Input: Parameter,
location, age,
weight

Animal

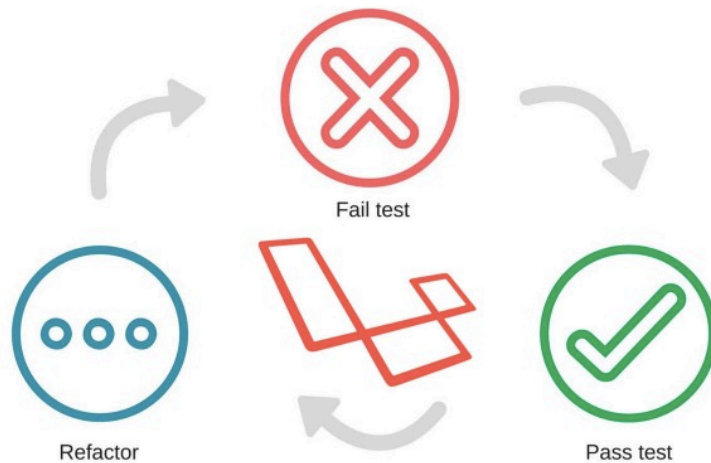
method
dripte

Veien til løsningen av BioSim prosjektet



Test driven development skaper en bedre kode




Test-Driven Development Cycle



Test Driven Development, lastet ned fra:

<https://www.freecodecamp.org/news/how-to-build-a-laravel-rest-api-with-test-driven-development-c4bb6417db3c/>

```
def test_gauss_distribution_pearson():...  
  
def test_gauss_distribution_shapiro():...
```

 animals.py	96% lines covered
 cell.py	97% lines covered
 map.py	100% lines covered

Nøkkelen til en mer effektiv kode



_calculate_fitness ×4028645 Total: 9984ms 25,9 % Own: 8005ms 20,8 %	→	_calculate_fitness ×6818311 Total: 3306ms 4,9 % Own: 2390ms 3,5 %
choices ×20444505 Total: 41610ms 42,7 % Own: 27011ms 27,7 %	→	uniform ×19877764 Total: 6299ms 9,3 % Own: 5186ms 7,6 %

“We should forget about small efficiencies, say about 97% of the time: **premature optimization is the root of all evil**”

— Donald Knuth



Bonusmateriale



BioSim

BioSim Parameters

Years to simulate Seed

Number of Herbivores Number of Carnivores

Map Type

Map len to add Carnivores

```
@classmethod
def _determine_sick(cls):
    """
    Determines if the animal is to become sick, using the 'p_sick'
    parameter and random.uniform method.

    Returns
    -----
    bool
    | True or False

    """
    p_sick = cls.param["p_sick"]
    return random.uniform(0, 1) < p_sick

def increase_eat_weight(self, fodder):
    """
    Increases the animal class instance's weight by :math:\beta times
    the current weight of the instance. The function also updates
    the fitness of the animal class instance using 'update_fitness'.

    Parameters
    -----
    fodder: int or float
    | Amount of food the animal instance is to eat.

    """

    self.is_sick = self._determine_sick()
    beta = self.param["beta"]
    loss_rate = self.param["loss_rate"]

    if self.is_sick:
        self._weight += beta * fodder * loss_rate
    else:
        self._weight += beta * fodder
    self.update_fitness()
```

