Big Assignment: Object-Oriented Programming

Project Design Document

Yusupov Boburjon
Neptun Code: YTAJDI

April 18, 2024

1. Project description: Wildlife Simulation

Wildlife conservation is the most critical issue in a major game reserve. Trained rangers venture out daily to spot wildlife, administer medicine to diseased animals, and protect the habitat from desecrating poachers. The reserve is a well-organized company where rangers, animals, habitats, and support vehicles coexist in symbiosis for fostering nature's fragile balance.

All animals have a species, health, and a stress level. Animals become stressed or injured due to natural events or poachers. The rangers can calm them down and treat them.

Rangers are diligent workers who protect the reserve's environment. Every ranger has a name, experience, and efficiency. They can assist an animal by improving its status or reducing stress. They can also detect animals that need help among all the animals and help them. If they are in the same habitat as a poacher, they find and fight the poacher. The winner is determined based on the strength of both the ranger and the poacher. The strength is initialized randomly.

Poachers are intruders who hunt down animals. Each poacher targets a specific species. Different poachers are dangerous to a different degree. Poachers disturb wildlife and cause them stress as they evade their hunters. They can also hurt the animals if they catch them.

Each habitat has a name, a capacity, and the animals living in it. Animals can enter and leave habitats. If there is an incursion by a poacher, it stresses all animals in the habitat.

The rangers travel in Jeep vehicles. The vehicle has an ID, fuel level, and a capacity. Vehicles can be refueled and deployed to specific habitats.

The simulation is run day-by-day. Each day, animals wander between habitats, rangers are sent to habitats to patrol them, and poachers go to habitats to hunt animals. The simulation reports at the end of the day the state of the reserve and the happenings on that day — for example, what animals were saved.

Play it out, roll out your rangers, and save the wildlife!

Input file examples
`animals.txt`

- Format
  Species Health StressLevel
- Example

```
Elephant 60 30
Giraffe 70 20
Zebra 80 25
```

`rangers.txt`

- Format
  Name Experience Efficiency
- Example

```
Balazs 5 7
Sofiia 6 6
```

`vehicles.txt`

- Format
  ID FuelLevel Capacity
- Example

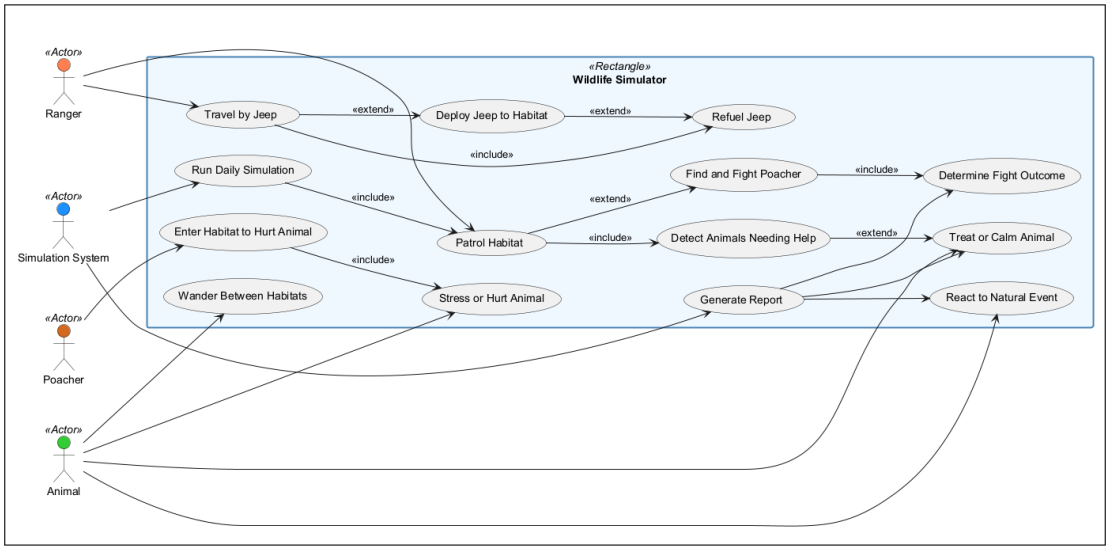
```
V1 100 2
V2 100 3
```

`habitats.txt`

- Format
  Name Capacity
- Example

```
Savannah 3
```

2. Use Case Diagram



Use Case Diagram

Use case diagram shows the interactions between the actors (Ranger, Poacher, Vehicle) and the system (Wildlife Simulation). The actors perform various actions such as patrolling, treating animals, hunting, and moving between habitats. The system manages the habitats, animals, and vehicles.

Ranger:

- Patrols the habitat to check for poachers and animals in need of help.
- Treats animals to improve their health and reduce stress.
- Fights poachers if they confront them in the same habitat.
- Moves between habitats to assist animals and patrol areas.
- Uses Jeep to travel between habitats.
- Refuels and deploys the Jeep to specific habitats.

Poacher:

- Incurses in the habitat to hunt animals.
- Hurts animals if they catch them and lowers their health level.
- Stresses animals in the habitat and increases their stress level.
- Moves between habitats to hunt animals.
- Fights rangers if they confront them in the same habitat.

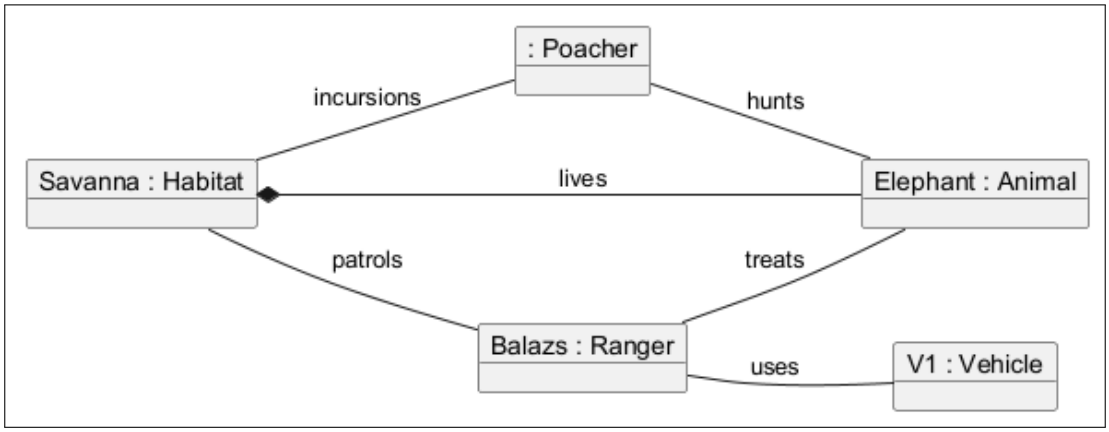
Animal:

- Wander between habitats.
- Get stressed or injured due to poachers or natural events.
- Get treated by rangers to improve their health and reduce stress.

Simulation System:

- Runs the simulation day-by-day.
- Generates report at the end of the day about the state of the reserve and happenings.

3. Object Diagram



Object Diagram

The **Object Diagram** illustrates the relationships and interactions among key entities in the *Wildlife Simulator* system. In object diagram and class diagram `Simulation` is excluded, main focus is to relation of classes and implementations of methods. `Simulation` is implemented in code. The primary objects involved are:

- **Balazs (Ranger)**
- **V1 (Vehicle)**
- **Savannah (Habitat)**
- **Elephant (Animal)**
- **Poacher**

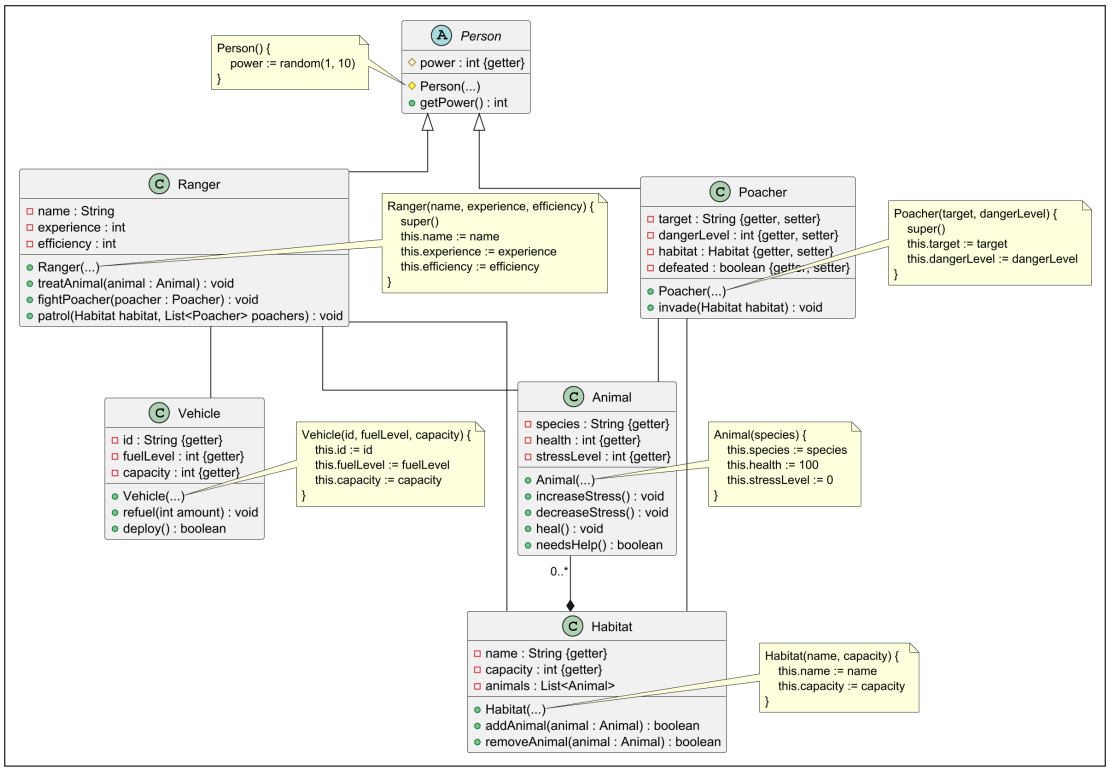
The relationships are characterized using standard UML semantics:

- **Composition (`*--`)**: The `Elephant` (Animal) is part of the `Savannah` (Habitat), indicating a strong lifecycle dependency. If the habitat ceases to exist, so does the animal.
- **Association (`--`)**:
  - The `Ranger` uses the `Vehicle` (e.g., to travel or patrol), showing a weak ownership relationship.
  - The `Ranger` can interact with `Animal` objects for treating or calming them.
  - The `Poacher` can target and hurt `Animals`.

This diagram helps visualize how individual components in the simulation are interconnected at runtime and supports understanding of object-level behavior and control flow.

4. Class Diagram

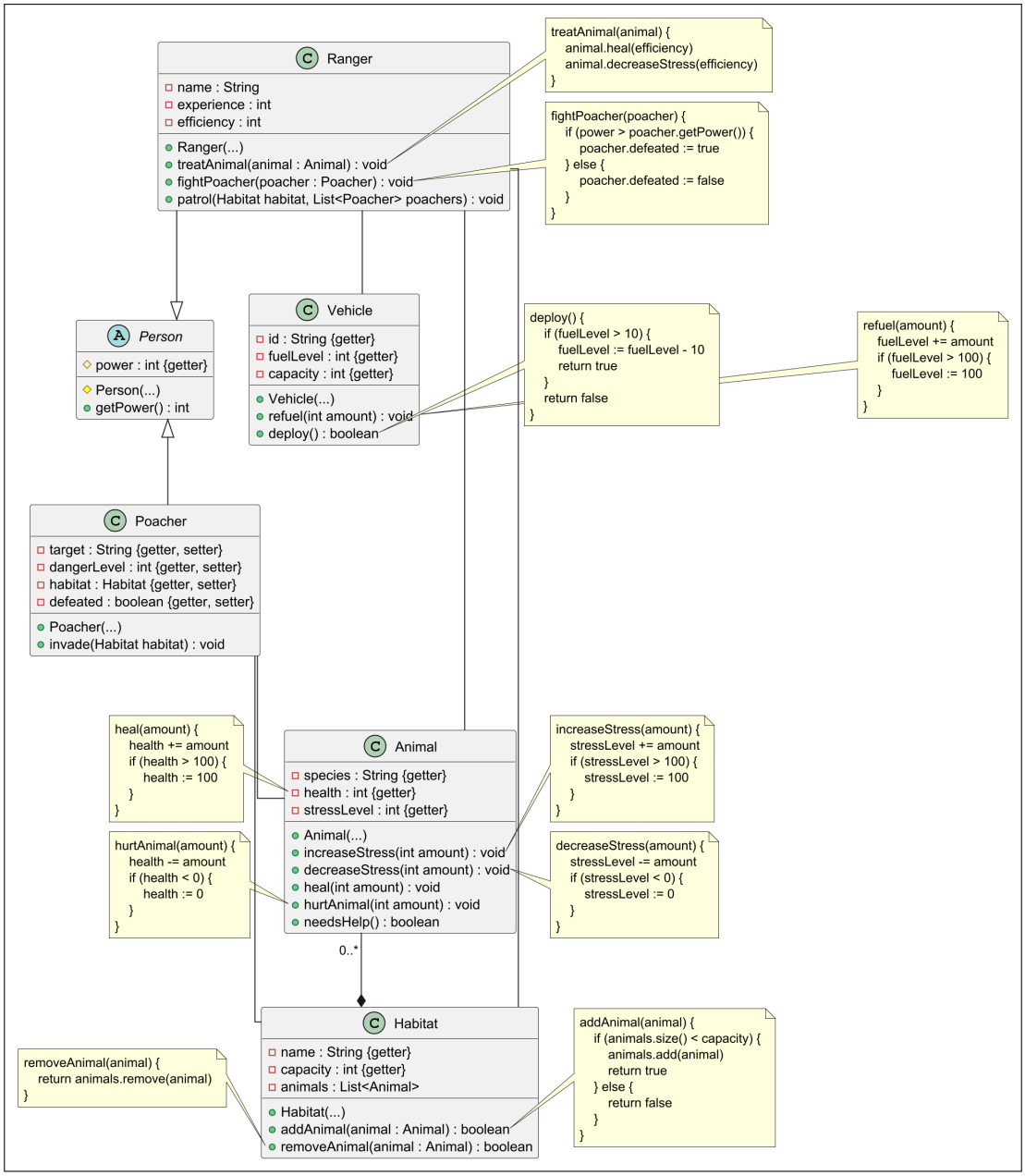
The implementations of the class diagrams are separated into three for better readability. The first contains implementations of constructors of classes, the second contains implementations of basic methods, and the third contains implementations of advanced methods.



Class Diagram 1

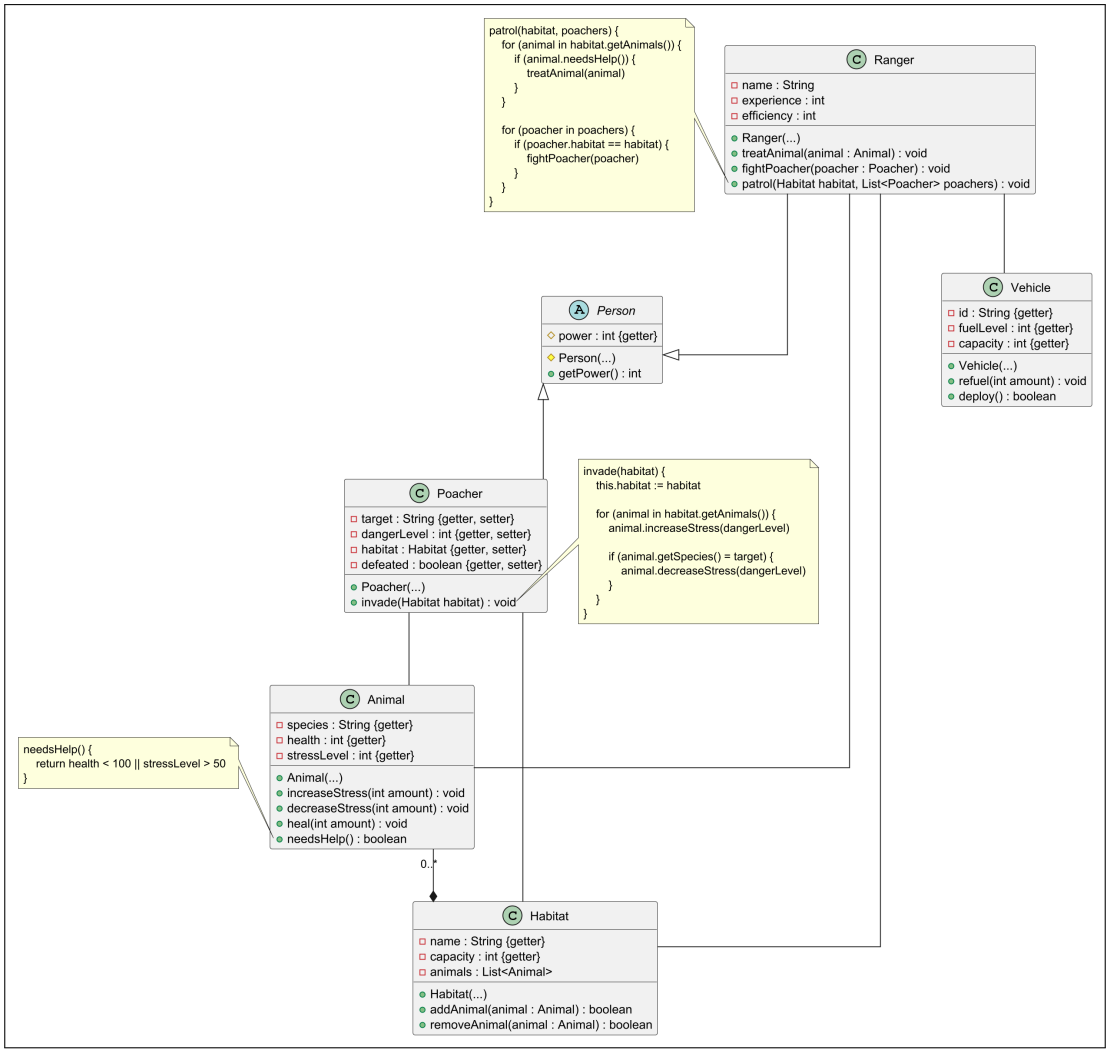
In the first class diagram you can see the implementations of constructors of classes. The classes are:

- **Person** (abstract class) - Abstract class Person has field `power` which is initialized randomly between 1 to 10.
- **Ranger** (inherits from Person)
- **Poacher** (inherits from Person)
- **Animal**
- **Vehicle**
- **Habitat**



Class Diagram 2

In the second class diagram you can see the implementations of basic methods of classes.



Class Diagram 3

In the third class diagram you can see the implementations of advanced methods of classes. The methods are:

- **Ranger**:
  - `patrol()` - patrols the habitat and treats the animals in need of help.
- **Poacher**:
  - `invade()` - invades the habitat and threatens the animals.