# Issues

# Code Snippets (V3+)

### \*\*Version 3 (V3) - Initial Evolution Features\*\*

\*\*1. Adjust Species Growth Rate\*\*

```java

public class Species {

private String name;

private int population;

private double growthRate;

public Species(String name, int population, double growthRate) {

this.name = name;

this.population = population;

this.growthRate = growthRate;

}

public void setGrowthRate(double growthRate) {

this.growthRate = growthRate;

}

public double getGrowthRate() {

return growthRate;

}

}

```

\*\*2. Adjust Species Reproductive Rate\*\*

```java

public class Species {

private String name;

private int population;

private double reproductiveRate;

public Species(String name, int population, double reproductiveRate) {

this.name = name;

this.population = population;

this.reproductiveRate = reproductiveRate;

}

public void setReproductiveRate(double reproductiveRate) {

this.reproductiveRate = reproductiveRate;

}

public double getReproductiveRate() {

return reproductiveRate;

}

}

```

\*\*3. Change Species Diet\*\*

```java

public class Species {

private String name;

private int population;

private String diet;

public Species(String name, int population, String diet) {

this.name = name;

this.population = population;

this.diet = diet;

}

public void setDiet(String diet) {

this.diet = diet;

}

public String getDiet() {

return diet;

}

}

```

\*\*4. Manage Species Movement\*\*

```java

public class Species {

private String name;

private int population;

private List<String> movementPatterns;

public Species(String name, int population, List<String> movementPatterns) {

this.name = name;

this.population = population;

this.movementPatterns = movementPatterns;

}

public void addMovementPattern(String pattern) {

movementPatterns.add(pattern);

}

public List<String> getMovementPatterns() {

return movementPatterns;

}

}

```

### \*\*Version 4 (V4) - Environmental Events and Scenario Simulations\*\*

\*\*1. Initiate Natural Disasters\*\*

```java

public class EnvironmentalEvent {

private String eventType;

private double impactSeverity;

public EnvironmentalEvent(String eventType, double impactSeverity) {

this.eventType = eventType;

this.impactSeverity = impactSeverity;

}

public void triggerEvent() {

System.out.println("Event: " + eventType + " with severity: " + impactSeverity);

}

}

```

\*\*2. Simulate Specific Scenarios\*\*

```java

public class ScenarioSimulator {

private List<String> scenarios = new ArrayList<>();

public void addScenario(String scenario) {

scenarios.add(scenario);

}

public void runScenarios() {

for (String scenario : scenarios) {

System.out.println("Running scenario: " + scenario);

}

}

}

```

### \*\*Version 5 (V5) - Advanced Evolutionary Dynamics\*\*

\*\*1. Simulate Mutation Events\*\*

```java

public class MutationEvent {

private String speciesName;

private String mutationType;

public MutationEvent(String speciesName, String mutationType) {

this.speciesName = speciesName;

this.mutationType = mutationType;

}

public void applyMutation() {

System.out.println("Applying mutation: " + mutationType + " to species: " + speciesName);

}

}

```

\*\*2. Implement Natural Selection\*\*

```java

public class NaturalSelection {

private List<Species> speciesList;

private double selectionPressure;

public NaturalSelection(List<Species> speciesList, double selectionPressure) {

this.speciesList = speciesList;

this.selectionPressure = selectionPressure;

}

public void applySelection() {

for (Species species : speciesList) {

System.out.println("Applying selection pressure: " + selectionPressure + " to species: " + species.getName());

}

}

}

```

\*\*3. Track Evolutionary Changes\*\*

```java

public class EvolutionTracker {

private List<Species> speciesList;

public EvolutionTracker(List<Species> speciesList) {

this.speciesList = speciesList;

}

public void printEvolutionaryChanges() {

for (Species species : speciesList) {

System.out.println("Species: " + species.getName() + ", Growth Rate: " + species.getGrowthRate() + ", Reproductive Rate: " + species.getReproductiveRate());

}

}

}

```

\*\*4. Adjust Genetic Variation\*\*

```java

public class GeneticVariation {

private String speciesName;

private double variationLevel;

public GeneticVariation(String species

# Code Structure

## Entities

This package will contain the core classes that represent the primary entities in your simulation.

- `Species`

- `Individual`

- `Population`

- `Event` (for specific events affecting entities)

## Environment

This package will include classes related to the environment, geographic locations, and how species interact with their surroundings.

- `Environment` (or `Location`)

- `Ecosystem`

- `Coordinates`

- `Climate` (if you're adding detailed climate attributes)

- `Vegetation`

- `Habitat`

## Control

This package will handle the overall control of the simulation, including time progression, events, and the main simulation loop.

- `Simulation`

- `TimeController`

- `EventManager`

- `InteractionManager`

- `SimulationSettings`

## Utils

Utility classes and methods that provide supporting functions across different parts of the simulation should go here.

- `MathUtils` (e.g., for distance calculations)

- `RandomGenerator` (if you need a custom random number generator)

- `Logger` (for logging simulation events)

## Enums

This package will contain all the enum types used across the simulation.

- `Diet`

- `MatingSeason`

- `SocialStructure`

- `ActivityPattern`

- `Habitat`

## Behaviors

If you plan to add more complex behaviors or interaction logic, this package will encapsulate that functionality.

- `Behavior`

- `Interaction`

- `PredationBehavior`

- `ReproductionBehavior`

## Events

If your simulation includes many different types of events (like natural disasters, migrations, etc.), it might be worth creating a separate package for them.

- `NaturalDisasterEvent`

- `MigrationEvent`

- `PopulationBoomEvent`

## UI

If you plan to add a user interface or even just text-based interactions, this package can encapsulate all UI-related classes.

- `ConsoleUI`

- `Menu`

- `UserInputHandler`

# Enums

## Diet

```java

public enum Diet {

HERBIVORE, // For animals that eat only plants

CARNIVORE, // For animals that eat only meat

OMNIVORE, // For animals that eat both plants and meat

INSECTIVORE, // For animals that eat insects

FRUGIVORE, // For animals that primarily eat fruits

NECTARIVORE, // For animals that primarily consume nectar

FISH, // For animals that primarily eat fish

SMALL\_MAMMALS, // For animals that primarily eat small mammals

AQUATIC\_PLANTS, // For animals that consume aquatic plants

SMALL\_INVERTEBRATES, // For animals that eat small invertebrates

BIRDS, // For animals that consume other birds

SEAL, // For animals that consume seals

CARIBOU // For animals that consume caribou

}

```

## MatingSeason

```java

public enum MatingSeason {

SPRING,

SUMMER,

FALL,

WINTER,

YEAR\_ROUND, // For species that mate throughout the year

LATE\_SUMMER // Specific for some species

}

```

## SocialStructure

```java

public enum SocialStructure {

SOLITARY, // For animals that live alone

PAIR, // For animals that form pair bonds

GROUPS, // For animals that form groups or herds

PACK, // For animals that live in packs

PRIDE // For animals that live in prides (e.g., lions)

}

```

## ActivityPattern

```java

public enum ActivityPattern {

DIURNAL, // Active during the day

NOCTURNAL, // Active during the night

CREPUSCULAR // Active during dawn and dusk

}

```

## Habitat

```java

public enum Habitat {

TROPICAL\_FORESTS,

TEMPERATE\_FORESTS,

SAVANNAS,

GRASSLANDS,

ICE\_CAPS,

SEAL\_HAULOUTS,

WETLANDS,

RIVERS,

LAKES,

COASTAL\_REGIONS,

CLIFFS,

URBAN\_AREAS,

TUNDRA,

OPEN\_FORESTS

}

```

# Grid

Yes, you can certainly use kilometers (or any other distance unit) for the radius of species distribution. To handle this, you'll need to convert between geographic coordinates and distance units. Here’s how you can adapt the `Species` class and distance calculations to use kilometers for the radius:

### \*\*1. Update the Species Class\*\*

Change the radius attribute to use kilometers and update the `isWithinDistribution` method to use the Haversine formula for distance calculation.

```java

public class Species {

private String name;

private int population;

private double growthRate;

private String diet;

private double latitude;

private double longitude;

private double radiusKm; // Radius of distribution area in kilometers

public Species(String name, int population, double growthRate, String diet, double latitude, double longitude, double radiusKm) {

this.name = name;

this.population = population;

this.growthRate = growthRate;

this.diet = diet;

this.latitude = latitude;

this.longitude = longitude;

this.radiusKm = radiusKm;

}

// Getters and setters

public double getRadiusKm() {

return radiusKm;

}

public void setRadiusKm(double radiusKm) {

this.radiusKm = radiusKm;

}

public double getLatitude() {

return latitude;

}

public double getLongitude() {

return longitude;

}

public void move(double newLatitude, double newLongitude, Grid grid) {

// Update the species' center location

this.latitude = newLatitude;

this.longitude = newLongitude;

// Optionally, update the grid if needed

}

// Check if a point is within the species' distribution radius

public boolean isWithinDistribution(double pointLatitude, double pointLongitude) {

double distance = calculateDistance(latitude, longitude, pointLatitude, pointLongitude);

return distance <= radiusKm;

}

// Calculate distance between two geographical points (Haversine formula)

private double calculateDistance(double lat1, double lon1, double lat2, double lon2) {

final int R = 6371; // Radius of the Earth in kilometers

double latDistance = Math.toRadians(lat2 - lat1);

double lonDistance = Math.toRadians(lon2 - lon1);

double a = Math.sin(latDistance / 2) \* Math.sin(latDistance / 2) +

Math.cos(Math.toRadians(lat1)) \* Math.cos(Math.toRadians(lat2)) \*

Math.sin(lonDistance / 2) \* Math.sin(lonDistance / 2);

double c = 2 \* Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

return R \* c; // Convert to kilometers

}

}

```

### \*\*2. Update the Grid Class\*\*

Make sure the grid can handle species with radius in kilometers. The grid should allow querying for species within a specified radius.

```java

public class Grid {

private int rows;

private int columns;

private Cell[][] cells;

private CoordinateConverter converter;

public Grid(double latMin, double latMax, double lonMin, double lonMax, int rows, int columns) {

this.rows = rows;

this.columns = columns;

this.cells = new Cell[rows][columns];

this.converter = new CoordinateConverter(latMin, latMax, lonMin, lonMax, rows, columns);

// Initialize each cell in the grid

for (int i = 0; i < rows; i++) {

for (int j = 0; j < columns; j++) {

cells[i][j] = new Cell();

}

}

}

public Cell getCell(double latitude, double longitude) {

int[] gridPos = converter.geoToGrid(latitude, longitude);

return cells[gridPos[0]][gridPos[1]];

}

public void setCell(double latitude, double longitude, Cell cell) {

int[] gridPos = converter.geoToGrid(latitude, longitude);

cells[gridPos[0]][gridPos[1]] = cell;

}

public List<Species> getSpeciesInRadius(double latitude, double longitude, double radiusKm) {

List<Species> speciesInRadius = new ArrayList<>();

for (int i = 0; i < rows; i++) {

for (int j = 0; j < columns; j++) {

Cell cell = cells[i][j];

for (Species species : cell.getSpeciesList()) {

if (species.isWithinDistribution(latitude, longitude)) {

speciesInRadius.add(species);

}

}

}

}

return speciesInRadius;

}

}

```

### \*\*3. Example Usage\*\*

Here’s how you might use the updated classes with kilometers for the radius:

```java

public class Main {

public static void main(String[] args) {

// Define real-world coordinates and grid dimensions

double latMin = -90.0, latMax = 90.0, lonMin = -180.0, lonMax = 180.0;

int rows = 180, columns = 360;

Grid grid = new Grid(latMin, latMax, lonMin, lonMax, rows, columns);

// Create a species with a distribution radius in kilometers

Species deer = new Species("Deer", 50, 0.1, "Herbivore", 40.0, -100.0, 50.0); // 50 km radius

grid.getCell(40.0, -100.0).addSpecies(deer);

// Move the species and update the grid

deer.move(41.0, -99.0, grid);

// Get species within a certain radius

List<Species> nearbySpecies = grid.getSpeciesInRadius(40.5, -99.5, 100.0); // 100 km radius

for (Species species : nearbySpecies) {

System.out.println("Nearby species: " + species.getName());

}

}

}

```

### \*\*Notes\*\*

- \*\*Distance Precision\*\*: The Haversine formula provides a good approximation for distances over the Earth's surface. For most simulations, this is sufficient.

- \*\*Coordinate Range\*\*: Ensure that your coordinate converter and grid size align with the real-world area you’re simulating.

- \*\*Performance\*\*: Checking distances and iterating over large grids can be computationally intensive. For larger simulations, consider optimizing data structures or algorithms.

By incorporating kilometers for the radius, you provide a more realistic model for species distribution in your simulation.

# Species List (old)

## Mammals / Herbivores

2. \*\*African Elephant\*\* (\*Loxodonta africana\*)

3. \*\*Giraffe\*\* (\*Giraffa camelopardalis\*)

4. \*\*American Bison\*\* (\*Bison bison\*)

5. \*\*Hippopotamus\*\* (\*Hippopotamus amphibius\*)

7. \*\*Zebra\*\* (\*Equus quagga\*)

8. \*\*Panda\*\* (\*Ailuropoda melanoleuca\*)

9. \*\*Kangaroo\*\* (\*Macropus rufus\*)

10. \*\*Rhinoceros\*\* (\*Rhinocerotidae family\*)

## Mammals / Carnivores

1. \*\*Lion\*\* (\*Panthera leo\*)

2. \*\*Tiger\*\* (\*Panthera tigris\*)

4. \*\*Polar Bear\*\* (\*Ursus maritimus\*)

5. \*\*Leopard\*\* (\*Panthera pardus\*)

6. \*\*Cheetah\*\* (\*Acinonyx jubatus\*)

7. \*\*Jaguar\*\* (\*Panthera onca\*)

8. \*\*Hyena\*\* (\*Hyaenidae family\*)

9. \*\*Puma/Mountain Lion\*\* (\*Puma concolor\*)

## Mammals / Omnivores

1. \*\*Brown Bear\*\* (\*Ursus arctos\*)

2. \*\*Raccoon\*\* (\*Procyon lotor\*)

3. \*\*Human\*\* (\*Homo sapiens\*)

4. \*\*Wild Boar\*\* (\*Sus scrofa\*)

5. \*\*Chimpanzee\*\* (\*Pan troglodytes\*)

6. \*\*Binturong\*\* (\*Arctictis binturong\*)

7. \*\*Giant Panda\*\* (\*Ailuropoda melanoleuca\*)

## Birds

1. \*\*Bald Eagle\*\* (\*Haliaeetus leucocephalus\*)

2. \*\*Peregrine Falcon\*\* (\*Falco peregrinus\*)

3. \*\*Golden Eagle\*\* (\*Aquila chrysaetos\*)

4. \*\*Harpy Eagle\*\* (\*Harpia harpyja\*)

5. \*\*Barn Owl\*\* (\*Tyto alba\*)

6. \*\*Emperor Penguin\*\* (\*Aptenodytes forsteri\*)

7. \*\*Peacock\*\* (\*Pavo cristatus\*)

8. \*\*Ostrich\*\* (\*Struthio camelus\*)

## Reptiles

1. \*\*Komodo Dragon\*\* (\*Varanus komodoensis\*)

2. \*\*King Cobra\*\* (\*Ophiophagus hannah\*)

3. \*\*Saltwater Crocodile\*\* (\*Crocodylus porosus\*)

4. \*\*Green Sea Turtle\*\* (\*Chelonia mydas\*)

5. \*\*American Alligator\*\* (\*Alligator mississippiensis\*)

## Amphibians

1. \*\*Axolotl\*\* (\*Ambystoma mexicanum\*)

2. \*\*Poison Dart Frog\*\* (\*Dendrobatidae family\*)

3. \*\*Tree Frog\*\* (\*Hylidae family\*)

## Fish

1. \*\*Clownfish\*\* (\*Amphiprioninae subfamily\*)

2. \*\*Great White Shark\*\* (\*Carcharodon carcharias\*)

3. \*\*Manta Ray\*\* (\*Mobula genus\*)

4. \*\*Salmon\*\* (\*Salmo genus\*)

## Invertebrates

1. \*\*Honeybee\*\* (\*Apis mellifera\*)

2. \*\*Monarch Butterfly\*\* (\*Danaus plexippus\*)

3. \*\*Jellyfish\*\* (e.g., \*Aurelia aurita\* - Moon Jellyfish)

4. \*\*Octopus\*\* (\*Octopus vulgaris\*)

## Plants

1. \*\*Oak Tree\*\* (\*Quercus genus\*)

2. \*\*Maple Tree\*\* (\*Acer genus\*)

3. \*\*Bamboo\*\* (\*Bambusoideae subfamily\*)

4. \*\*Cactus\*\* (\*Cactaceae family\*)

# Species List

#### \*\*Mammals (Carnivores/Predators)\*\*

1. \*\*Gray Wolf\*\* (\*Canis lupus\*)

2. \*\*Cougar (Mountain Lion)\*\* (\*Puma concolor\*)

3. \*\*American Black Bear\*\* (\*Ursus americanus\*)

4. \*\*Grizzly Bear\*\* (\*Ursus arctos horribilis\*)

5. \*\*Coyote\*\* (\*Canis latrans\*)

6. \*\*Bobcat\*\* (\*Lynx rufus\*)

7. \*\*Red Wolf\*\* (\*Canis rufus\*)

8. \*\*Lion\*\* (\*Panthera leo\*)

9. \*\*Tiger\*\* (\*Panthera tigris\*)

10. \*\*Leopard\*\* (\*Panthera pardus\*)

11. \*\*Spotted Hyena\*\* (\*Crocuta crocuta\*)

12. \*\*African Wild Dog\*\* (\*Lycaon pictus\*)

13. \*\*Snow Leopard\*\* (\*Panthera uncia\*)

14. \*\*Jaguar\*\* (\*Panthera onca\*)

15. \*\*Eurasian Lynx\*\* (\*Lynx lynx\*)

16. \*\*Polar Bear\*\* (\*Ursus maritimus\*)

17. \*\*Dhole\*\* (\*Cuon alpinus\*)

#### \*\*Mammals (Herbivores/Prey)\*\*

1. \*\*White-tailed Deer\*\* (\*Odocoileus virginianus\*)

2. \*\*Moose\*\* (\*Alces alces\*)

3. \*\*Elk\*\* (\*Cervus canadensis\*)

4. \*\*Caribou\*\* (\*Rangifer tarandus\*)

5. \*\*Bison\*\* (\*Bison bison\*)

6. \*\*Beaver\*\* (\*Castor canadensis\*)

7. \*\*Snowshoe Hare\*\* (\*Lepus americanus\*)

8. \*\*Wild Boar\*\* (\*Sus scrofa\*)

9. \*\*Muskox\*\* (\*Ovibos moschatus\*)

10. \*\*Mountain Goat\*\* (\*Oreamnos americanus\*)

11. \*\*Zebra\*\* (\*Equus quagga\*)

12. \*\*Wildebeest\*\* (\*Connochaetes taurinus\*)

13. \*\*Thomson's Gazelle\*\* (\*Eudorcas thomsonii\*)

14. \*\*Impala\*\* (\*Aepyceros melampus\*)

15. \*\*Cape Buffalo\*\* (\*Syncerus caffer\*)

16. \*\*Giraffe\*\* (\*Giraffa camelopardalis\*)

17. \*\*Sambar Deer\*\* (\*Rusa unicolor\*)

18. \*\*Chital (Spotted Deer)\*\* (\*Axis axis\*)

19. \*\*Wild Water Buffalo\*\* (\*Bubalus arnee\*)

20. \*\*Tapir\*\* (\*Tapirus spp.\*)

#### \*\*Birds (Predators)\*\*

1. \*\*Golden Eagle\*\* (\*Aquila chrysaetos\*)

2. \*\*Harpy Eagle\*\* (\*Harpia harpyja\*)

3. \*\*Martial Eagle\*\* (\*Polemaetus bellicosus\*)

4. \*\*Crowned Eagle\*\* (\*Stephanoaetus coronatus\*)

### \*\*Predator-Prey Pairings\*\*

#### \*\*North American Ecosystem\*\*

| \*\*Predator\*\* | \*\*Prey\*\* | \*\*Preference Rate (%)\*\* | \*\*Hunt Success Rate (%)\*\* |

|----------------------|-------------------------|-------------------------|---------------------------|

| \*\*Gray Wolf\*\* | White-tailed Deer | 35% | 50% |

| | Moose | 25% | 30% |

| | Elk | 20% | 40% |

| | Caribou | 15% | 35% |

| | Bison | 5% | 10% |

| \*\*Cougar\*\* | White-tailed Deer | 40% | 60% |

| | Elk | 30% | 45% |

| | Moose | 20% | 25% |

| | Beaver | 10% | 75% |

| \*\*American Black Bear\*\* | White-tailed Deer | 25% | 30% |

| | Elk | 20% | 25% |

| | Moose | 15% | 20% |

| | Snowshoe Hare | 40% | 60% |

| \*\*Grizzly Bear\*\* | White-tailed Deer | 20% | 40% |

| | Elk | 30% | 35% |

| | Moose | 25% | 30% |

| | Caribou | 15% | 25% |

| | Bison | 10% | 20% |

| \*\*Coyote\*\* | White-tailed Deer | 30% | 45% |

| | Snowshoe Hare | 50% | 60% |

| | Beaver | 15% | 50% |

| | Wild Boar | 5% | 20% |

| \*\*Bobcat\*\* | White-tailed Deer (Fawns) | 35% | 55% |

| | Snowshoe Hare | 50% | 60% |

| | Beaver | 15% | 45% |

#### \*\*African Savanna Ecosystem\*\*

| \*\*Predator\*\* | \*\*Prey\*\* | \*\*Preference Rate (%)\*\* | \*\*Hunt Success Rate (%)\*\* |

|----------------------|-------------------------|-------------------------|---------------------------|

| \*\*Lion\*\* | Zebra | 30% | 40% |

| | Wildebeest | 25% | 50% |

| | Thomson's Gazelle | 20% | 60% |

| | Cape Buffalo | 15% | 25% |

| | Giraffe | 10% | 20% |

| \*\*Leopard\*\* | Thomson's Gazelle | 30% | 55% |

| | Impala | 40% | 65% |

| | Wildebeest (Calves) | 20% | 45% |

| | Zebra (Fawns) | 10% | 35% |

| \*\*Spotted Hyena\*\* | Wildebeest | 35% | 55% |

| | Zebra | 25% | 45% |

| | Thomson's Gazelle | 25% | 50% |

| | Impala | 15% | 40% |

| \*\*African Wild Dog\*\* | Impala | 40% | 70% |

| | Thomson's Gazelle | 30% | 60% |

| | Wildebeest | 20% | 50% |

| | Zebra | 10% | 30% |

#### \*\*Asian Forest Ecosystem\*\*

| \*\*Predator\*\* | \*\*Prey\*\* | \*\*Preference Rate (%)\*\* | \*\*Hunt Success Rate (%)\*\* |

|----------------------|-------------------------|-------------------------|---------------------------|

| \*\*Tiger\*\* | Sambar Deer | 40% | 60% |

| | Chital (Spotted Deer) | 35% | 55% |

| | Wild Boar | 15% | 40% |

| | Gaur | 10% | 20% |

| \*\*Leopard\*\* | Chital (Spotted Deer) | 35% | 60% |

| | Sambar Deer | 30% | 50% |

| | Wild Boar | 25% | 45% |

| | Langur | 10% | 70% |

| \*\*Dhole\*\* | Chital (Spotted Deer) | 50% | 60% |

| | Sambar Deer | 30% | 45% |

| | Wild Boar | 20% | 40% |

#### \*\*South American Rainforest Ecosystem\*\*

| \*\*Predator\*\* | \*\*Prey\*\* | \*\*Preference Rate (%)\*\* | \*\*Hunt Success Rate (%)\*\* |

|----------------------|-------------------------|-------------------------|---------------------------|

| \*\*Jaguar\*\* | Tapir | 40% | 50% |

| | Capybara | 30% | 60% |

| | Peccary | 20% | 55% |

| | Howler Monkey | 10% | 70% |

| \*\*Harpy Eagle\*\* | Howler Monkey | 40% | 70% |

| | Sloth | 30% | 80% |

| | Capybara | 20% | 60% |

| | Peccary | 10% | 50% |

# Species Data

Certainly! Here are three examples of carnivorous mammals with their respective details:

### \*\*1. Lion (`Panthera leo`)\*\*

```java

class Species {

String scientificName = "Panthera leo";

String commonName = "Lion";

String[] taxonomy = {"Mammalia", "Carnivora", "Felidae"};

int averageWeight = 190; // In kg

int height = 120; // At the shoulder, in cm

int lifeSpan = 15; // Average lifespan in the wild

Diet[] diet = {Diet.MAMMALS}; // Primarily large herbivores

MatingSeason matingSeason = MatingSeason.YEAR\_ROUND;

int gestationDays = 110; // Approximately 3.5 months

int minOffspring = 1;

int maxOffspring = 4;

SocialStructure socialStructure = SocialStructure.PRIDE;

boolean isTerritorial = true;

ActivityPattern activityPattern = ActivityPattern.CREPSUCULAR;

List<Species> predators = new ArrayList<>(); // Humans, rarely other large predators

List<Species> competitors = new ArrayList<>(); // Hyenas, leopards

String range = "Sub-Saharan Africa, parts of India";

Habitat[] habitats = {Habitat.SAVANNAS, Habitat.GRASSLANDS, Habitat.OPEN\_FORESTS};

int movementRadius = 20; // In kilometers

int minDensity = 1;

int maxDensity = 5; // Density varies with habitat and prey availability

double cubMortalityRate = 0.4; // Mortality rate for cubs

int carryingCapacity = 10; // Example value, varies with prey density

int maxSpeed = 50; // In km/h

// Constructor, methods to add predators/competitors, etc.

}

```

### \*\*2. Gray Wolf (`Canis lupus`)\*\*

```java

class Species {

Diet[] diet = {Diet.MAMMALS, Diet.BIRDS, Diet.FISH}; // Opportunistic feeders

MatingSeason matingSeason = MatingSeason.WINTER;

int gestationDays = 63; // Approximately 2 months

int minOffspring = 4;

int maxOffspring = 7;

SocialStructure socialStructure = SocialStructure.PACK;

boolean isTerritorial = true;

ActivityPattern activityPattern = ActivityPattern.DIURNAL;

List<Species> predators = new ArrayList<>(); // Humans, larger predators like bears

String range = "North America, Europe, Asia";

Habitat[] habitats = {Habitat.TEMPERATE\_FORESTS, Habitat.TUNDRA, Habitat.GRASSLANDS};

int movementRadius = 30; // In kilometers

int minDensity = 1;

int maxDensity = 10;

double pupMortalityRate = 0.3; // Mortality rate for pups

int carryingCapacity = 20; // Example value

int maxSpeed = 60; // In km/h

// Constructor, methods to add predators/competitors, etc.

}

```

### \*\*3. Polar Bear (`Ursus maritimus`)\*\*

```java

class Species {

String scientificName = "Ursus maritimus";

String commonName = "Polar Bear";

String[] taxonomy = {"Mammalia", "Carnivora", "Ursidae"};

int averageWeight = 450; // In kg

int height = 150; // At the shoulder, in cm

int lifeSpan = 20; // Average lifespan in the wild

Diet[] diet = {Diet.SEAL, Diet.CARIBOU, Diet.BIRDS}; // Primarily seals

MatingSeason matingSeason = MatingSeason.SPRING;

int gestationDays = 240; // Approximately 8 months

int minOffspring = 1;

int maxOffspring = 3;

SocialStructure socialStructure = SocialStructure.SOLITARY;

boolean isTerritorial = true;

ActivityPattern activityPattern = ActivityPattern.DIURNAL;

List<Species> predators = new ArrayList<>(); // None significant for adults

List<Species> competitors = new ArrayList<>(); // Other polar bears

String range = "Arctic regions of the Northern Hemisphere";

Habitat[] habitats = {Habitat.ICE\_CAPS, Habitat.SEAL\_HAULOUTS};

int movementRadius = 50; // In kilometers

int minDensity = 0.1;

int maxDensity = 2;

double cubMortalityRate = 0.5; // Mortality rate for cubs

int carryingCapacity = 5; // Example value, varies with prey availability

int maxSpeed = 40; // In km/h

// Constructor, methods to add predators/competitors, etc.

}

```

These examples provide a range of details for different carnivorous mammals, including their dietary habits, social structures, and environmental requirements.

Certainly! Below are examples of three species for each of the other major groups: birds, reptiles, amphibians, and fish.

### \*\*1. Birds\*\*

#### \*\*Bald Eagle (`Haliaeetus leucocephalus`)\*\*

```java

class Species {

String scientificName = "Haliaeetus leucocephalus";

String commonName = "Bald Eagle";

String[] taxonomy = {"Aves", "Accipitriformes", "Accipitridae"};

int averageWeight = 5; // In kg

int wingspan = 210; // In cm

int lifeSpan = 20; // Average lifespan in the wild

Diet[] diet = {Diet.FISH, Diet.SMALL\_MAMMALS, Diet.CARRION};

MatingSeason matingSeason = MatingSeason.WINTER;

int gestationDays = 35; // Incubation period for eggs

int minOffspring = 1;

int maxOffspring = 3;

SocialStructure socialStructure = SocialStructure.SOLITARY;

boolean isTerritorial = true;

ActivityPattern activityPattern = ActivityPattern.DIURNAL;

List<Species> predators = new ArrayList<>(); // Rarely predated as adults

List<Species> competitors = new ArrayList<>(); // Other raptors

String range = "North America";

Habitat[] habitats = {Habitat.WETLANDS, Habitat.COASTAL\_REGIONS, Habitat.LARGE\_LAKES};

int movementRadius = 50; // In kilometers

int minDensity = 1;

int maxDensity = 2; // Low density due to territorial behavior

double chickMortalityRate = 0.5; // Mortality rate for chicks

int carryingCapacity = 10; // Example value, low due to large territory needs

int maxSpeed = 160; // In km/h when diving

// Constructor, methods to add predators/competitors, etc.

}

```

#### \*\*Peregrine Falcon (`Falco peregrinus`)\*\*

```java

class Species {

String scientificName = "Falco peregrinus";

String commonName = "Peregrine Falcon";

String[] taxonomy = {"Aves", "Falconiformes", "Falconidae"};

int averageWeight = 1.2; // In kg

int wingspan = 110; // In cm

int lifeSpan = 15; // Average lifespan in the wild

Diet[] diet = {Diet.BIRDS}; // Primarily other birds

MatingSeason matingSeason = MatingSeason.SPRING;

int gestationDays = 32; // Incubation period for eggs

int minOffspring = 2;

int maxOffspring = 4;

SocialStructure socialStructure = SocialStructure.SOLITARY;

boolean isTerritorial = true;

ActivityPattern activityPattern = ActivityPattern.DIURNAL;

List<Species> predators = new ArrayList<>(); // None significant for adults

List<Species> competitors = new ArrayList<>(); // Other raptors

String range = "Worldwide";

Habitat[] habitats = {Habitat.CLIFFS, Habitat.URBAN\_AREAS, Habitat.COASTAL\_REGIONS};

int movementRadius = 20; // In kilometers

int minDensity = 1;

int maxDensity = 5;

double chickMortalityRate = 0.6; // Mortality rate for chicks

int carryingCapacity = 20; // Example value

int maxSpeed = 390; // In km/h during hunting stoop (dive)

// Constructor, methods to add predators/competitors, etc.

}

```

#### \*\*Mallard Duck (`Anas platyrhynchos`)\*\*

```java

class Species {

String scientificName = "Anas platyrhynchos";

String commonName = "Mallard Duck";

String[] taxonomy = {"Aves", "Anseriformes", "Anatidae"};

int averageWeight = 1.2; // In kg

int wingspan = 90; // In cm

int lifeSpan = 5; // Average lifespan in the wild

Diet[] diet = {Diet.AQUATIC\_PLANTS, Diet.SMALL\_INVERTEBRATES};

MatingSeason matingSeason = MatingSeason.SPRING;

int gestationDays = 28; // Incubation period for eggs

int minOffspring = 5;

int maxOffspring = 12;

SocialStructure socialStructure = SocialStructure.GROUPS;

boolean isTerritorial = false;

ActivityPattern activityPattern = ActivityPattern.DIURNAL;

List<Species> predators = new ArrayList<>(); // Foxes, raptors, raccoons

List<Species> competitors = new ArrayList<>(); // Other waterfowl

String range = "Northern Hemisphere";

Habitat[] habitats = {Habitat.WETLANDS, Habitat.LAKES, Habitat.RIVERS};

int movementRadius = 10; // In kilometers

int minDensity = 10;

int maxDensity = 200;

double chickMortalityRate = 0.7; // Mortality rate for ducklings

int carryingCapacity = 100; // Example value

int maxSpeed = 88; // In km/h

// Constructor, methods to add predators/competitors, etc.

}

```

### \*\*2. Reptiles\*\*

#### \*\*American Alligator (`Alligator mississippiensis`)\*\*

```java

class Species {

String scientificName = "Alligator mississippiensis";

String commonName = "American Alligator";

String[] taxonomy = {"Reptilia", "Crocodylia", "Alligatoridae"};

int averageWeight = 230; // In kg

int length = 400; // In cm (total length)

int lifeSpan = 35; // Average lifespan in the wild

Diet[] diet = {Diet.FISH, Diet.BIRDS, Diet.MAMMALS};

MatingSeason matingSeason = MatingSeason.SPRING;

int gestationDays = 65; // Incubation period for eggs

int minOffspring = 20;

int maxOffspring = 50;

SocialStructure socialStructure = SocialStructure.SOLITARY;

boolean isTerritorial = true;

ActivityPattern activityPattern = ActivityPattern.NOCTURNAL;

List<Species> predators = new ArrayList<>(); // None significant for adults

List<Species> competitors = new ArrayList<>(); // Other large reptiles

String range = "Southeastern United States";

Habitat[] habitats = {Habitat.WETLANDS, Habitat.RIVERS, Habitat.LAKES};

int movementRadius = 2; // In kilometers

int minDensity = 1;

int maxDensity = 3;

double hatchlingMortalityRate = 0.8; // High mortality rate for young alligators

int carryingCapacity = 5; // Example value

int maxSpeed = 32; // In km/h on land

// Constructor, methods to add predators/competitors, etc.

}

```

#### \*\*Green Iguana (`Iguana iguana`)\*\*

```java

class Species {

String scientificName = "Iguana iguana";

String commonName = "Green Iguana";

String[] taxonomy = {"Reptilia", "Squamata", "Iguanidae"};

int averageWeight = 4; // In kg

int length = 200; // In cm (total length)

int lifeSpan = 15; // Average lifespan in the wild

Diet[] diet = {Diet.LEAVES, Diet.FRUITS, Diet.FLOWERS};

MatingSeason matingSeason = MatingSeason.LATE\_SUMMER;

int gestationDays = 90; // Incubation period for eggs

int minOffspring = 20;

int maxOffspring = 70;

SocialStructure socialStructure = SocialStructure.SOLITARY;

boolean isTerritorial = true;

ActivityPattern activityPattern = ActivityPattern.DIURNAL;

List<Species> predators = new ArrayList<>(); // Birds of prey, snakes, mammals

List<Species> competitors = new ArrayList<>(); // Other iguanas, herbivorous reptiles

String range = "Central and South America";

Habitat[] habitats = {Habitat.TROPICAL\_FORESTS, Habitat.RIVERSIDES};

int movementRadius = 1; // In kilometers

int minDensity = 2;

int maxDensity = 10;

double hatchlingMortalityRate = 0.9; // Very high mortality rate for young iguanas

int maxSpeed = 35; // In km/h

// Constructor, methods to add predators/competitors, etc.

}

```

#### \*\*Garter Snake (`Thamnophis sirtalis`)\*\*

```java

class Species {

String scientificName = "Thamnophis sirtalis";

String commonName = "Garter Snake";

String[] taxonomy = {"Reptilia", "Squamata", "Colubridae"};

int averageWeight = 0.15; // In kg

int length = 100; // In cm (total length)

int lifeSpan = 6; // Average lifespan in the wild

Diet[] diet = {Diet.SMALL\_MAMMALS, Diet.AMPHIBIANS, Diet

### \*\*2. Eastern Gray Squirrel (`Sciurus carolinensis`)\*\*

```java

class Species {

String scientificName = "Sciurus carolinensis";

String commonName = "Eastern Gray Squirrel";

String[] taxonomy = {"Mammalia", "Rodentia", "Sciuridae"};

int averageWeight = 0.5; // In kg

int height = 23; // Body length in cm (excluding tail)

int lifeSpan = 6; // Average lifespan in the wild

Diet[] diet = {Diet.NUTS, Diet.SEEDS, Diet.FRUITS, Diet.TWIGS};

MatingSeason matingSeason = MatingSeason.WINTER;

int gestationDays = 44; // Approximately 1.5 months

int minOffspring = 2;

int maxOffspring = 8;

SocialStructure socialStructure = SocialStructure.SOLITARY;

boolean isTerritorial = true;

ActivityPattern activityPattern = ActivityPattern.DIURNAL;

List<Species> predators = new ArrayList<>(); // Hawks, foxes, snakes

List<Species> competitors = new ArrayList<>(); // Other squirrels, chipmunks

String range = "Eastern United States, parts of Canada";

Habitat[] habitats = {Habitat.DECIDUOUS\_FORESTS, Habitat.URBAN\_AREAS, Habitat.PARKLANDS};

int movementRadius = 0.5; // In kilometers

int minDensity = 2;

int maxDensity = 15;

double fawnMortalityRate = 0.3; // Mortality rate for young squirrels

int maxSpeed = 20; // In km/h

boolean hasAntlers = false; // Not applicable

// Constructor, methods to add predators/competitors, etc.

}

```

### \*\*3. Plains Zebra (`Equus quagga`)\*\*

```java

class Species {

String scientificName = "Equus quagga";

String commonName = "Plains Zebra";

String[] taxonomy = {"Mammalia", "Perissodactyla", "Equidae"};

int averageWeight = 350; // In kg

int height = 140; // At the shoulder, in cm

int lifeSpan = 20; // Average lifespan in the wild

Diet[] diet = {Diet.GRASSES};

MatingSeason matingSeason = MatingSeason.YEAR\_ROUND;

int gestationDays = 365; // Approximately 12 months

int minOffspring = 1;

int maxOffspring = 1;

SocialStructure socialStructure = SocialStructure.SMALL\_FAMILY\_GROUPS;

boolean isTerritorial = false;

ActivityPattern activityPattern = ActivityPattern.DIURNAL;

List<Species> predators = new ArrayList<>(); // Lions, hyenas, cheetahs

List<Species> competitors = new ArrayList<>(); // Other grazing herbivores (e.g., wildebeests)

String range = "Sub-Saharan Africa";

Habitat[] habitats = {Habitat.GRASSLANDS, Habitat.SAVANNAS};

int movementRadius = 5; // In kilometers

int minDensity = 5;

int maxDensity = 50;

double fawnMortalityRate = 0.4; // Mortality rate for foals

int carryingCapacity = 200; // Example value

int maxSpeed = 65; // In km/h

boolean hasAntlers = false; // Not applicable

// Constructor, methods to add predators/competitors, etc.

}

```

### \*\*4. African Elephant (`Loxodonta africana`)\*\*

```java

class Species {

String scientificName = "Loxodonta africana";

String commonName = "African Elephant";

String[] taxonomy = {"Mammalia", "Proboscidea", "Elephantidae"};

int averageWeight = 5000; // In kg

int height = 300; // At the shoulder, in cm

int lifeSpan = 60; // Average lifespan in the wild

Diet[] diet = {Diet.GRASSES, Diet.LEAVES, Diet.BARK, Diet.FRUIT};

MatingSeason matingSeason = MatingSeason.YEAR\_ROUND;

int gestationDays = 660; // Approximately 22 months

int minOffspring = 1;

int maxOffspring = 1;

SocialStructure socialStructure = SocialStructure.HERDS;

boolean isTerritorial = false;

ActivityPattern activityPattern = ActivityPattern.DIURNAL;

List<Species> predators = new ArrayList<>(); // None significant for adults, lions for calves

List<Species> competitors = new ArrayList<>(); // Other large herbivores

String range = "Sub-Saharan Africa";

Habitat[] habitats = {Habitat.SAVANNAS, Habitat.GRASSLANDS, Habitat.FORESTS};

int movementRadius = 20; // In kilometers

int minDensity = 0.1;

int maxDensity = 2;

double fawnMortalityRate = 0.3; // Mortality rate for calves

int carryingCapacity = 10; // Example value, lower due to large resource needs

int maxSpeed = 40; // In km/h

boolean hasAntlers = false; // Not applicable, has tusks instead

// Constructor, methods to add predators/competitors, etc.

}

```

### \*\*5. Moose (`Alces alces`)\*\*

```java

class Species {

Diet[] diet = {Diet.LEAVES, Diet.TWIGS, Diet.BARK, Diet.AQUATIC\_PLANTS};

MatingSeason matingSeason = MatingSeason.FALL;

int gestationDays = 230; // Approximately 8 months

int minOffspring = 1;

int maxOffspring = 2;

SocialStructure socialStructure = SocialStructure.SOLITARY;

boolean isTerritorial = true; // Bulls during the rut

ActivityPattern activityPattern = ActivityPattern.CREPUSCULAR;

List<Species> predators = new ArrayList<>(); // Wolves, bears, humans

String range = "Northern Hemisphere (North America, Europe, Asia)";

Habitat[] habitats = {Habitat.BOREAL\_FORESTS, Habitat.MIXED\_WOODLANDS, Habitat.WETLANDS};

int movementRadius = 5; // In kilometers

int minDensity = 0.1;

int maxDensity = 3;

double fawnMortalityRate = 0.5; // Mortality rate for calves

int maxSpeed = 56; // In km/h

boolean hasAntlers = true; // Males have large antlers

// Constructor, methods to add predators/competitors, etc.

}

```

### \*\*6. Giant Panda (`Ailuropoda melanoleuca`)\*\*

```java

class Species {

String scientificName = "Ailuropoda melanoleuca";

String commonName = "Giant Panda";

String[] taxonomy = {"Mammalia", "Carnivora", "Ursidae"};

int averageWeight = 100; // In kg

int height = 90; // At the shoulder, in cm

int lifeSpan = 20; // Average lifespan in the wild

Diet[] diet = {Diet.BAMBOO}; // Primarily bamboo

MatingSeason matingSeason = MatingSeason.SPRING;

int gestationDays = 135; // Variable due to delayed implantation

int minOffspring = 1;

int maxOffspring = 2;

SocialStructure socialStructure = SocialStructure.SOLITARY;

boolean isTerritorial = true;

ActivityPattern activityPattern = ActivityPattern.CREPUSCULAR;

List<Species> predators = new ArrayList<>(); // None significant for adults, leopards for cubs

List<Species> competitors = new ArrayList<>(); // Other bamboo-eating species

String range = "Mountain ranges of central China";

Habitat[] habitats = {Habitat.TEMPERATE\_FORESTS, Habitat.BAMBOO\_FORESTS};

int movementRadius = 1; // In kilometers

int minDensity = 0.5;

int maxDensity = 2;

double fawnMortalityRate = 0.6; // Mortality rate for cubs

int carryingCapacity = 20; // Example value, lower due to specific diet

int maxSpeed = 32; // In km/h

boolean hasAntlers = false; // Not applicable

// Constructor, methods to add predators/competitors, etc.

}

``

The values provided earlier are based on general knowledge and typical ranges for white-tailed deer. However, let's verify and ensure these values align with real-world data:

### \*\*White-Tailed Deer (Odocoileus virginianus) - Realistic Attributes\*\*

3. \*\*Taxonomy\*\*: Mammalia (Class), Artiodactyla (Order), Cervidae (Family)

- \*\*Diet\*\*: Grasses, leaves, fruits, twigs, acorns, and occasionally crops like corn or soybeans.

### \*\*Behavioral Attributes\*\*

- \*\*Mating Season\*\*: Late fall, generally from October to December (referred to as the "rut").

- \*\*Gestation Period\*\*: ~200 days, leading to fawns being born in late spring (May-June).

- \*\*Offspring\*\*: Typically 1-2 fawns per birth; occasionally 3.

- \*\*Social Structure\*\*: Small family groups, usually consisting of a doe and her fawns; bucks are more solitary.

- \*\*Territoriality\*\*: Not strictly territorial, but males may defend a small area during the rut.

- \*\*Daily Activity\*\*: Crepuscular—most active during dawn and dusk.

### \*\*Environmental Interactions\*\*

- \*\*Predators\*\*: Wolves, coyotes, mountain lions, bears, humans (through hunting).

- \*\*Range\*\*: Native to North and Central America; found in a variety of habitats including forests, fields, and sometimes urban areas.

- \*\*Preferred Habitat\*\*: Deciduous forests, mixed woodlands, agricultural areas.

### \*\*Population Dynamics\*\*

- \*\*Movement Radius\*\*: Typically around 1-2 km, but can range farther during the rut or in search of food.

- \*\*Population Density\*\*:

- Can vary widely, but typically 5-30 deer per square kilometer depending on the habitat quality.

- \*\*Fawn Mortality Rate\*\*: High, especially in the first few months; often around 50%.

- \*\*Carrying Capacity\*\*: Dependent on food availability, predation, and habitat conditions.

### \*\*Speed\*\*

- \*\*Maximum Speed\*\*: Can run up to 48-56 km/h (30-35 mph) for short bursts.

### \*\*Updated Example in Code\*\*

Here’s how these verified values translate into the code:

```java

class Species {

String scientificName = "Odocoileus virginianus";

String commonName = "White-Tailed Deer";

String[] taxonomy = {"Mammalia", "Artiodactyla", "Cervidae"};

int averageWeight = 90; // In kg, average between males and females

int height = 100; // In cm

int lifeSpan = 5; // Average lifespan in the wild

Diet[] diet = {Diet.GRASSES, Diet.LEAVES, Diet.FRUITS, Diet.TWIGS};

MatingSeason matingSeason = MatingSeason.FALL;

int gestationDays = 200; // Approximately 6.5 months

int minOffspring = 1;

int maxOffspring = 2;

SocialStructure socialStructure = SocialStructure.SMALL\_FAMILY\_GROUPS;

boolean isTerritorial = false; // Generally not territorial

ActivityPattern activityPattern = ActivityPattern.CREPUSCULAR;

List<Species> predators = new ArrayList<>(); // To be filled with predator species

List<Species> competitors = new ArrayList<>(); // To be filled with competitor species

String range = "North and Central America";

Habitat[] habitats = {Habitat.DECIDUOUS\_FORESTS, Habitat.MIXED\_WOODLANDS, Habitat.AGRICULTURAL\_AREAS};

int movementRadius = 2; // In kilometers

int minDensity = 5;

int maxDensity = 30;

double fawnMortalityRate = 0.5; // 50%

int carryingCapacity = 100; // Example, depends on habitat quality

int maxSpeed = 56; // In km/h

boolean hasAntlers = true; // Males grow antlers

// Constructor, methods to add predators/competitors, etc.

}

```

### \*\*Summary\*\*

These values reflect a more realistic and scientifically accurate depiction of the white-tailed deer. The focus is on ensuring that these attributes can be used effectively within your simulation while representing real-world data. Adjustments like these will allow your simulation to model species behavior and interactions more accurately.

# Hunt Success Rate

private double calculateSuccessFactor(Species predator, Species prey, Environment environment) {

// Calculate base success rate considering various factors (size, speed, camouflage, etc.)

double baseSuccess = predator.getHuntingSkill() - prey.getEvasionSkill();

baseSuccess += (predator.getSpeed() - prey.getSpeed()) \* 0.1;

baseSuccess -= prey.getCamouflageLevel() \* 0.1;

baseSuccess += environment.getTerrainModifier(predator, prey);

}

```

Certainly! Calculating the predation success rate for predator-prey pairs is an important part of modeling predator-prey dynamics in a simulation. Let's break down how to approach this:

### \*\*Factors Influencing Predation Success Rate\*\*

1. \*\*Predator Attributes\*\*:

- \*\*Speed\*\*: Faster predators might have higher success rates.

- \*\*Strength\*\*: Stronger predators could overpower their prey more easily.

- \*\*Hunting Strategy\*\*: Pack hunters (e.g., wolves) may have higher success rates compared to solitary hunters.

- \*\*Experience\*\*: Older or more experienced predators might be more successful.

2. \*\*Prey Attributes\*\*:

- \*\*Speed\*\*: Faster prey might evade predators more often.

- \*\*Defense Mechanisms\*\*: Prey with better camouflage, physical defenses (like antlers), or behaviors (e.g., fleeing in herds) may have lower predation rates.

- \*\*Alertness\*\*: Prey with better senses or vigilance might detect predators earlier.

3. \*\*Environmental Factors\*\*:

- \*\*Terrain\*\*: Dense forests or rough terrain might lower predator success rates due to difficulty in maneuvering.

- \*\*Weather\*\*: Poor visibility or harsh conditions could reduce predator success.

- \*\*Time of Day\*\*: Nocturnal predators might have higher success rates at night.

4. \*\*Predator-Prey Relationship\*\*:

- \*\*Size Difference\*\*: Larger predators typically have higher success rates against smaller prey.

- \*\*Specific Adaptations\*\*: Some predators are specialized to hunt specific prey, increasing their success rates against those targets.

### \*\*Modeling Predation Success Rate\*\*

One possible formula for the success rate could be:

\[

\text{Success Rate} = \frac{A\_p \times E\_p \times (1 - D\_p)}{A\_{prey} \times E\_{prey} \times (1 - D\_{prey})} \times F\_{env} \times F\_{time}

\]

Where:

- \*\*\( A\_p \)\*\*: Attribute factor for the predator (e.g., speed, strength).

- \*\*\( E\_p \)\*\*: Experience or efficiency of the predator.

- \*\*\( D\_p \)\*\*: Defense mechanisms or difficulties for the predator (e.g., terrain difficulties).

- \*\*\( A\_{prey} \)\*\*: Attribute factor for the prey (e.g., speed, defense mechanisms).

- \*\*\( E\_{prey} \)\*\*: Efficiency or alertness of the prey.

- \*\*\( D\_{prey} \)\*\*: Defense mechanisms or evasive abilities of the prey.

- \*\*\( F\_{env} \)\*\*: Environmental factor (terrain, weather).

- \*\*\( F\_{time} \)\*\*: Factor for the time of day (or season).

### \*\*Example of Simplified Predation Success Rate\*\*

Consider a wolf (predator) and a white-tailed deer (prey):

- \*\*Wolf\*\*:

- Speed: 60% (relative to maximum possible)

- Strength: 70%

- Experience: 80%

- Pack Hunting: Yes

- \*\*White-tailed Deer\*\*:

- Speed: 70%

- Alertness: 60%

- Defense Mechanisms: 50% (due to agility and awareness)

Assuming favorable terrain and time:

\[

\text{Success Rate} = \frac{(0.6 + 0.7 + 0.8) \times (1 - 0.2)}{(0.7 + 0.6 + 0.5) \times (1 - 0.1)} \times 1.0 \times 1.0

\]

Simplifying:

\[

\text{Success Rate} = \frac{2.1 \times 0.8}{1.8 \times 0.9} \approx \frac{1.68}{1.62} \approx 1.037

\]

Since this is slightly above 1, it suggests the wolf has a moderate success rate against the deer under these conditions. You can then scale this by a percentage (e.g., a base success rate of 50%) to get a final probability.

### \*\*Adjusting the Success Rate for Multiple Prey or Predators\*\*

- \*\*Multiple Prey Options\*\*: The predator might prioritize easier targets, lowering the success rate for more difficult prey.

- \*\*Multiple Predators\*\*: Prey might face compounded risks, or multiple predators might interfere with each other, affecting the success rate.

### \*\*Conclusion\*\*

To apply this in your simulation:

- Calculate a base success rate using factors relevant to your species and environment.

- Adjust this rate based on real-world data or observed behaviors.

- Apply this success rate in your weekly simulation steps to determine the outcome of predation attempts.

This approach allows for a flexible, dynamic system that can adapt to different predator-prey pairs and environmental conditions.

# Carrying Capacity

Determining the carrying capacity for each species in your simulation involves considering the ecological role, habitat requirements, and typical densities in natural environments. Here’s a rough estimation of carrying capacities for the species we’ve discussed. These values are simplified and generalized for the purposes of your simulation.

### \*\*Suggested Carrying Capacities\*\*

| \*\*Species\*\* | \*\*Carrying Capacity (K)\*\* | \*\*Notes\*\* |

|----------------------|---------------------------|--------------------------------------------------|

| \*\*Bison\*\* | 800 | Open plains, large herds. |

| \*\*Elephant\*\* | 150 | High space and resource needs, fewer individuals.|

| \*\*Rabbit\*\* | 5000 | High reproduction rate, dense populations. |

| \*\*Squirrel\*\* | 3000 | Abundant in forests, smaller territory needs. |

| \*\*Cattle\*\* | 600 | Managed populations, found in grasslands. |

| \*\*Horse\*\* | 400 | Open plains, relatively large herds. |

| \*\*Giraffe\*\* | 200 | Requires large areas with tall trees. |

| \*\*Zebra\*\* | 1000 | Plains, large herds, similar to bison. |

| \*\*Gazelle\*\* | 1500 | Open grasslands, high population density. |

| \*\*Lion\*\* | 100 | Territorial, smaller groups (prides). |

| \*\*Tiger\*\* | 60 | Solitary, large territories. |

| \*\*Cheetah\*\* | 150 | Solitary or small groups, large territories. |

| \*\*Bear\*\* | 100 | Solitary, large home ranges. |

| \*\*Hyena\*\* | 200 | Territorial, lives in clans. |

| \*\*Leopard\*\* | 150 | Solitary, adaptable to various environments. |

| \*\*Cougar\*\* | 150 | Solitary, large territories. |

| \*\*Polar Bear\*\* | 100 | Arctic regions, solitary, vast territories. |

| \*\*Eagle\*\* | 1000 | Large territory but lower population densities. |

| \*\*Fox\*\* | 2000 | Highly adaptable, found in diverse habitats. |

| \*\*Hawk\*\* | 1500 | Smaller territories, higher population densities.|

| \*\*Owl\*\* | 1000 | Nocturnal, territorial, large home ranges. |

### \*\*Explanation of Carrying Capacities:\*\*

1. \*\*Habitat and Territory\*\*: Species with larger territory requirements or those that are solitary hunters (e.g., tigers, bears) have lower carrying capacities. Herd or pack animals (e.g., bison, deer) typically have higher carrying capacities.

2. \*\*Reproduction Rate\*\*: Species that reproduce rapidly and have many offspring, such as rabbits and squirrels, have higher carrying capacities.

3. \*\*Food Resources and Niche\*\*: Predators typically have lower carrying capacities because they are limited by the availability of prey. Herbivores and omnivores often have higher capacities, especially those that feed on abundant vegetation.

4. \*\*Environmental Constraints\*\*: Some species, like elephants or moose, have significant resource needs per individual, leading to lower carrying capacities even in suitable habitats.

These values are meant to be a starting point and can be adjusted based on the specific environment and the dynamics you wish to simulate. In a real ecosystem, carrying capacities can also change over time due to factors like climate changes, human activity, or natural events.