Zoo Simulation GUI with Strategy Pattern

Project Overview

This project is a desktop application that simulates an interactive virtual zoo. Users can explore different animals and learn how their behaviors vary based on species—such as how they move, eat, or make sounds. The program uses object-oriented programming and implements the **Strategy Design Pattern** to allow each animal to have dynamic, interchangeable behaviors.

For example, just like we learned in class with different types of ducks (some can fly, others cannot), this zoo will model different **species of animals** with unique behaviors. A *Congo lion* might roar loudly, while an *African lion* might roar softly or remain silent. These behaviors are encapsulated in separate strategy classes that can be selected.

Project Goals

- Build a modular and extensible zoo simulation application.
- Apply the **Strategy Design Pattern** to separate and manage animal behaviors.
- Create a **graphical user interface (GUI)** where each **animal has its own button**.
  - When a user clicks on an animal (e.g., Lion), a new panel or set of buttons will appear to show **different species of that animal** (e.g., Congo Lion, African Lion).
  - Each species will have its own specific behaviors (movement, sound, diet).

  .
- Include **at least 10 animals**, each with **multiple species** and a variety of behavior combinations.

System Architecture

1. Main Components

- **ZooApplication**: Main class to start the application.
- **ZooGUI**: User interface built using a GUI toolkit (e.g., Java Swing/JavaFX).

  - Displays buttons for each animal (Lion, Elephant, Parrot, etc.).
  - When clicked, opens a species panel for that animal.
  - Allows interaction with behaviors: “Make Sound”, “Move”, “Eat”, etc.

- **Animal (Abstract Class or Interface)**: Base class for all animals using composition to hold behavior strategies.
  - Methods: `performMove()`, `performSound()`, `performEat()`
- **Behavior Interfaces**:
  - `MovementBehavior` → walk, fly, swim, jump, slither
  - `SoundBehavior` → roar, chirp, bark, silent, trumpet
  - `DietBehavior` → carnivore, herbivore, omnivore, insectivore
- **Species Subclasses**: Each species of an animal is a concrete subclass of `Animal` and assigned unique behavior strategies.

Example Scenario: Lion

User clicks on the “Lion” button.

- A new panel appears with buttons:
  - “Congo Lion”
  - “African Lion”
- Each button loads the selected species and shows its behaviors:
  - **Congo Lion**
    - Sound: Loud Roar
    - Movement: Walk
    - Diet: Carnivore
  - **African Lion**
    - Sound: Muffled Roar or Silent
    - Movement: Walk
    - Diet: Carnivore
- User can click:
  - **Make Sound** → Displays species-specific roar
  - **Move** → Shows how that lion species moves
  - **Eat** → Describes its diet

Mark Distribution (Out of 100)

Project Design and Architecture	20 marks
Correct Implementation of Strategy Pattern	20 marks
GUI Functionality and Usability	20 marks
Diversity of Animals and Behaviors	15 marks
Code Readability and Documentation	10 marks
Creativity and Extra Features (e.g., sound effects, species panel)	5 marks
Detailed Report	10 marks