Exercise with Abstract Classes in Java: Zoo Management System

Task Description

Create a simple zoo management system using abstract classes to model different types of animals. This exercise will help you understand how abstract classes work and when to use them in object-oriented programming.

Requirements

1. Create an abstract class `Animal` containing:
   - Protected instance variables `name` (String) and `age` (int)
   - A constructor initializing these variables
   - Accessor methods (getters) for both variables
   - An abstract method `makeSound()` that returns a String
   - An abstract method `move()` that returns a String
   - A concrete method `displayInformation()` that displays information about the animal
2. Create at least three concrete derived classes that extend the `Animal` class:
   - `Mammal` (example: Lion, Elephant)
   - `Bird` (example: Eagle, Penguin)
   - `Reptile` (example: Snake, Crocodile)
3. Each concrete class must:
   - Implement all abstract methods from the parent class
   - Add at least one unique attribute specific to that type of animal
   - Add at least one unique method specific to that type of animal
4. Create a `ZooManagement` class with a `main` method that:
   - Creates instances of each type of animal
   - Calls methods demonstrating polymorphism
   - Shows how abstract methods are implemented differently in each subclass

Example Implementation Structure

java

```java
// Abstract base class
public abstract class Animal {

}
// Example of a concrete subclass
public class Lion extends Mammal {

}
```

Additional Challenge

Add an additional abstract method `naturalHabitat()` in the `Animal` class and implement it in all subclasses to return a description of the animal's natural habitat.

This task helps students understand:

- How abstract classes differ from regular classes and interfaces
- When and why to use abstract classes
- How to implement abstract methods in concrete subclasses
- The concept of polymorphism in object-oriented programming

Demonstrate the functionality of the created classes in the main method.