**Project Overview**

For this project, you will be extending the Animal Care System of eZoo, a web-based zoo management platform for zoo employees and customers. You will be provided with a barebones application for tracking animals in the zoo

**Drive 1 Specification**

Before you can get started working with the eZoo project, it's key that you understand what is being asked of you and how the existing starter kit is structured.

eZoo is a web-based zoo management platform for zoo employees and customers. It is designed to keep track of daily activities within a zoo, and to serve as a central location for data that needs to be stored and shared. Employees will be able to keep track of all of the animals that come in and out of the zoo using the Animal Care System within the eZoo application.

In the next dive, you will download a starter kit. The starter kit is a functional application that you will be able to start and run immediately. The existing functionality is for the Animal Care System, which handles the addition of new animals to the zoo.

Your task for this level 100 eZoo project will be to add functionality to the existing program to support the creation of various feeding schedules. A "feeding schedule" will be a simple record that tracks an animal's dietary needs, a regular feeding time, and any medications that may be required. An individual feeding schedule might be assigned to multiple animals, but each animal will only have one feeding schedule. You will implement this functionality in parts, doing a little bit at the end of each dive. The specific functions you will be adding will allow users to...

* Create a new feeding schedule
* Update the information for an existing feeding schedule
* Assign a feeding schedule to an animal
* Delete an existing feeding schedule
* See which feeding schedule has been assigned to a given animal

The eZoo application saves all animal information to a database, and your feeding schedule functionality will have to store dietary information there too. In the next dive you'll learn all about databases and SQL, set up the eZoo starter kit, and create your database tables.

**Drive 2 Specification**

Now that your tables are designed and created, it's time to start writing the (Java) code that you will use to interface with them. First, let's start by reviewing the DAO (Data Access Objects) interface provided in the starter kit to deal with Animals.

We see that the **AnimalDAO.java** file is just an interface, which is *implemented*by the **AnimalDAOImpl** class. In the DAOUtilities class we have a method that will instantiate and return a new instance of AnimalDAOImpl for us... but it returns it inside a AnimalDAO interface reference. Why is that?

Well, recall that an interface is capable of calling its own declared methods against a subclass reference that has those methods defined. You should also recall that just like any polymorphic reference, you can use a single superclass reference for multiple types of subclass objects. In this way, we save ourselves some time and protect our application in the face of future upgrades.

Consider a situation in which the eZoo application is undergoing another update - this time, to add vetinarian-prescibed feeding schedules. Vetenarian prescriptions have a separate feeding schedule system, but their database interactions are otherwise exactly the same as a standard schedule. In this case, we might still use the same FeedingScheduleDAO interface to implement with our VetFeedingScheduleDAOImpl class. Then, no change is required in the code which calls the tagging database methods. We can use the same FeedingScheduleDAO interface reference to manipulate both a FeedingScheduleDAOImpl object *and* and VetFeedingScheduleDAOImpl object.

Your final task this week, is to write a java file that defines an interface. This interface should have methods for each functionality that the system needs to support. You won't need to write an implementation class yet, that's a task for the next dive.

You will need, at minimum:

* A method to add a given feeding schedule to the database.
* A method to delete a given feeding schedule to the database. This method should first remove all references to that feeding schedule from the ANIMALS table.
* A method to retrieve all feeding schedules from the database
* A method to retrieve a single feeding schedule from the database for a given animal
* A method to assign a given feeding schedule to a given animal.
* A method to remove a feeding schedule from a given animal.

NOTE: You can call the file FeedingScheduleDAO.

**Drive 3 Specification**

Last week, you created the tables for your tagging system and laid the groundwork for a system that will bring that data into your application for manipulation. To do that, you will need to store that data in a POJO - a Plain Old Java Object - for which a single instance represents a single tag. For reference, look at the Animal class contained in the Starter Kit - that's a POJO too.

For all intents and purposes, a POJO is just a Java class that defines some properties, and has getters and setters for accessing and manipulating those properties. Every instance of a POJO represents an instance of a tangible object or idea - in this case, each instance of an Animal class represents a single Animal. Put another way, each row of the Animal table corresponds to a single instance of an Animal object.

Your next task is to create a Feeding Schedule POJO that can store and map the data you're saving in your database tables, and relate it to the Animal class if necessary. You may decide to keep the two as separate entities, or you may decide to relate them through composition. In other words, you may decide to have all Animal objects contain a reference to a FeedingSchedule object.

NOTE: This is a simple class with properties and getters/setters for those properties. If you wish, you can name the class **FeedingSchedule.java**

**Drive 3.2 Specification**

In the last dive, you wrote the interfaces that declare the methods your application will use to pull data from, and send data to, your tagging tables. This week, your task will be to implement those interfaces and code your complete Data Access Layer.

As usual, feel free to use the existing AnimalDAOImpl class as a guideline for this process.

NOTE: You can call the file FeedingScheduleDAOImpl. This class should implement the DAO interface that you created earlier.

**Drive 4 Specification**

In the previous dives, you've established a pretty firm understanding of what operations a user is going to need to perform in the application. Your task is now to build the Java Servlets that will direct a user through the application by serving them new views, and process their data manipulation.

Remember, you can use the Starter Kit as a guideline for what servlets you will need. You can also add functionality to existing servlets if that fits your design better.

You will need, at minimum, servlets to do the following:

* CreateFeedingScheduleServlet - call the DAO method to create a new feeding schedule in the database.
* AssignFeedingScheduleServlet - call the DAO method to assign an existing feeding schedule to an animal. This may not be necessary if each feeding schedule is unique and singular for each animal, and you assign the animal when creating the feeding schedule.
* DeleteFeedingScheduleServlet - Call the DAO method to remove a feeding schedule from the database, and remove all references to that feeding schedule from Animals in the database.

**Drive 5 Specification**

In this dive, you've recieved a pretty thorough crash course on HTML, CSS, and the powerful tool that is the Java Server Page. Now it's time to implement what you've learned!

Your next task is to use your newfound knowledge of clientside technologies to design the JSPs that will display your application to the users. As usual, you have some discretion regarding how to approach this - you may choose to implement your feeding schedule functionality in new, separate JSPs, or you may decide to add that functionality to existing JSPs.

You will need, at minimum:

* A page to view all feeding schedules that have been created, and the animals to which they have been assigned. A delete button next to each feeding schedule should remove it and all references to it from the DB.
* A page to create a new feeding schedule.
* A page to update a feeding schedule.
* A page to assign or unassign a feeding schedule to an animal.