Zoo Project Report

November 19, 2008

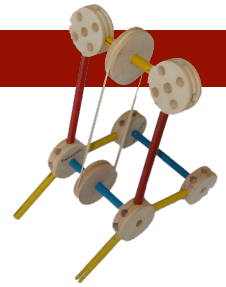


Table Of Contents

The Zoo Problem Description

The Bentley Zoo is composed of a collection of exotic and domestic animals housed in modern enclosures for the protection of the animals and the visiting public.

The enclosures (also known as cages to the zoo staff) are grouped in three areas of the zoo grounds and designated for favorite animal foods: Apple, Banana, and Carrot. There are main enclosures designated by area and number (e.g. A1, B2, C3). Within these buildings are separate animal cages and also numbered (e.g. A1.1, A1.2, B2.3, etc.). Some enclosures have special characteristics such as barred or unbarred confinement or open enclosures protected by moats. Cages may be small, medium or large in size.

Animals are categorized by species (e.g. Lion, Panther, Monkey, etc.). When a new animal arrives at the zoo it is given a name (if it doesn't already have one) which is unique within their species (.e.g. Molly the panther, Sam the monkey, etc.). Each animal's gender is recorded to manage their social groups. Each animal is assigned to a specific enclosure.

Animals are fed from food stores maintained in the zoo's warehouse. These foods are purchased in various denominations (pounds, bales, or bushels). Some foods for exotic zoo residents are actually purchased by the container without any quantity metric.

Because of the varying ages and health conditions of each animal their diets are regulated individually. An animal's diet consists of one or more rations of a particular food delivered in specific quantities per day on 1 to 7 days each week. One of the keeper's primary tasks is to prepare the delivery of animal rations to their respective cages on their designated feeding schedule.

We now turn our focus to the behavior in the problem domain that we wish to model.

"The zoo is staffed by employes who care for the grounds and the animals. Each employee has a badge with a unique number. There are special employees who are trained to care and feed the animals. These employees are called feeders.

The feeder is responsible for delivering a balanced collection of food servings to each animal according the their dietary requirements. These requirements lead to the storage of foods in the zoo's warehouse. Every time a food item is added or removed from the warehouse the inventory is updated.

Each feeder is responsible for a group of animals housed in one of the designated zoo areas. At regular

intervals a feeder will prepare a list of all the meals to be served to the animals on his/her feeding rounds and subsequently deliver those meals to the animals in their cages."

We need to explore three activities that are part of normal operations at the Bentley zoo.

The first is the setting up of diets for all the animals. This is a specialized task that requires some advanced knowledge about the animals and their needs. Therefore, we're adding a new employee to the zoo model, a veterinary nutritionist who is qualified to prescribe special diets for each of the animals. We want to model the process that the veterinarian goes through to set up each diet for the animals. Use your imagination and remember we're looking for something that's useful not necessarily perfect. We can refine the behavioral model with the users later on.

The second task to model is the preparation of the list of food deliveries to the animals by the feeder staff. We're looking for the list of each set of foods that need to be delivered by any particular feeder. Remember that feeder staff are assigned to specific zoo areas. Remember that for efficiency it would be very nice if the list of meals were arranged in the order of the cages that the animals live in. Use your imagination and remember we're looking for something that's useful not necessarily perfect. We can refine the behavioral model with the users later on.

The third task is the actual delivery of the meals to the animals. The feeder needs to account for every meal delivered and keep track of the time that each animal was fed. You can assume that the serving list was prepared in advance of this activity so the task here is rather straightforward, go through the list and serve the food! You can discard the serving list once you've completed the delivery process. Use your imagination and remember we're looking for something that's useful not necessarily perfect. We can refine the behavioral model with the users later on.

The documentation that follows represents the development of an object-oriented model of the above description rendered in UML-2. As with most OO models this version represents an evolving draft that would continue to be refined depending on its eventual purpose: overview, analysis, design, or implementation. The goal of modeling is to reach "a useful model." It is not possible to define a "perfect" or "correct" model. A model is an evolving understanding documented by the modeling stakeholders. The base documentation which is adapted and reformatted here was generated by Together Version 6.2™, Borland, Inc.

Class Diagrams

UML version of Zoo problem generated using Together 6.2 by Borland, Inc. This is draft of the model in progress on its way to be being "useful." LJW

Class Diagrams

diagram <default>

Interaction Diagrams

diagram Feed the Animals

diagram Prepare Animal Diets

diagram Prepare Serving List

UseCase Diagrams

diagram Feeding the Animals

Classes

class Animal

class Cage

class DietaryItem

class Feeder

class Food

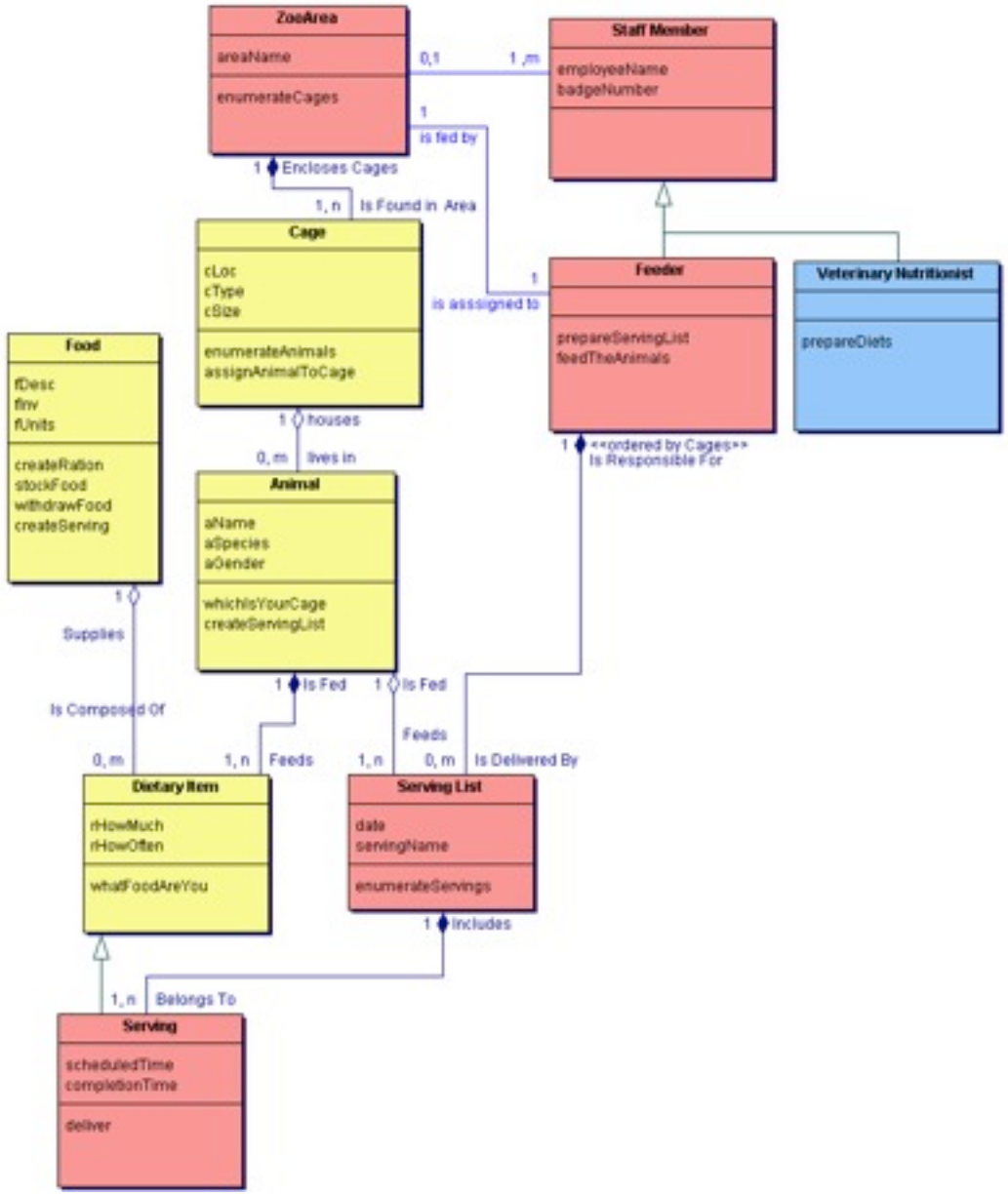
class Serving

class ServingList

class StaffMember

class VeterinaryNutritionist

class ZooArea



Class Detail

Class *Animal*

`public class Animal`

This is an animal housed in the zoo.

Attribute Summary	
private int	**aGender** The gender of this animal.
private int	**aName** The given name of the animal.
private int	**aSpecies** The biological species of this animal.
private DietaryItem	**[association] lnkDietaryItem** This is a collection of one or more dietary items for a particular animal.
private ServingList	**[association] lnkServingList** This is a collection of one or more serving lists for a particular animal.

Service Summary	
public ServingList	**createServingList**(Animal theanimal, int howmuch, int howoften) This service creates a list of food servings based upon the specific dietary items designated for this animal.
public Cage	**whichIsYourCage**() This service returns a link to the cage in which this animal is domiciled.

Class *Cage*

`public class Cage`

This is an enclosure that houses an animal.

Attribute Summary	
private int	**cLoc** The location of the cage.
private int	**cSize** Cage size: small, medium, large.
private int	**cType** Type of cage: moat, bars, unbarred.
private Animal	**[association] lnkAnimal** A cage may be empty.

Service Summary	
public void	**assignAnimalToCage**(Animal theAnimal) This service allows a zookeeper to assign a particular animal to a particular cage.
public Animal	**enumerateAnimals**() This service successively returns a link to each of the animals housed in it.

Class *DietaryItem*

`public class DietaryItem`

This is a particular ration definition of food for a specific animal.

Attribute Summary	
private int	**rHowMuch** How many units of the designated food are alloted to one ration for the designated animal.
private int	**rHowOften** The number of times during the feeding period that this animal is given this ration (e.g.

Service Summary	
public Food	**whatFoodAreYou**() The dietary item identifies the food object to which it belongs.

Class *Feeder*

```
StaffMember
  |
  +--Feeder
```

`public class Feeder`

Extends:

StaffMember

This is a specially trained staff member who is responsible for the care and feeding of the animals.

Attribute Summary	
private ServingList	**[association] lnkServingList** A serving list is the sole responsibility of a single feeder staff member.

Service Summary	
public void	**feedTheAnimals**() This service actually brings the servings to each cage to feed the animals.
public void	**prepareServingList**(ZooArea theArea) This service prepares a list of food servings derived from the dietary needs of each animal.

Association Note!
** << this case tool adds a placeholder for the association to object(s) of the class the attribute name references - this is not necessary in OUR models - However, you must document the "purpose" and cardinality of all associations as a separate section in the prose documentation >>

Class *Food*

```
public class Food
```

This is a category of food which is stored in the zoo warehouse for the feeding of the animals.

Attribute Summary	
private int	**fDesc** A description of the food type (i.e.
private int	**fInv** The number of units of this food found in the food storage.
private int	**fUnits** The type of units with which this food is measured.
private DietaryItem	**[association] lnkDietaryItem** A collection of dietaryitem objects created from a specific food type.

Service Summary	
public void	**createRation**(Animal theanimal, int howmuch, int howoften) This service creates a dietary item for a specifc animal designating the amount and frequency of this ration for that animal.
public void	**createServing**() This service withdraws food from the food warehouse and prepares a single serving of same for its particular animal.
public void	**stockFood**() This service updates the current inventory of this food when supplies are placed in the warehouse.
public boolean	**withdrawFood**() This service notes the withdrawal of food of this type form the warehouse.

Class *Serving*

```
DietaryItem
  |
  +--Serving
```

```
public class Serving
```

Extends:

DietaryItem

A specialization of DietaryItem indicating a physical instance of food to be given to an animal.

Attribute Summary	
private EasternStandardTime	**completionTime** Time the serving was actually delivered.
private EasterStandardTime	**scheduledTime** Time the serving is sheduled to be delivered.

Service Summary	
public Eastern Standar dTime	**deliver**(EasternStandardTime Time)

Class *ServingList*

```
public class ServingList
```

This is a zoo staff member whose responsibility is to manage the feeding of the animals in the zoo.

Attribute Summary	
private calendar Day	**date** The calendar date that this serving list is intended to be fed to the animal.
private Serving	**[association] lnkDietaryItem** A collection of servings to be delivered to a particular animal.
private String	**servingName** A string indicating the name of the serving list.

Service Summary	
public Serving	**enumerateServings**()

Class *StaffMember*

```
public class StaffMember
```

This is the general representation of a zoo staff member.

Attribute Summary	
private int	**badgeNumber** A unique identifying code used to verify employee identity.
private int	**employeeName** Legal name of zoo staff member.

Class *VeterinaryNutritionist*

```
StaffMember
  |
  +--VeterinaryNutritionist
```

```
public class
VeterinaryNutritionist
```

Extends:

StaffMember

Service Summary	
public void	**prepareDiets**()

Class ZooArea

```
public class ZooArea
```

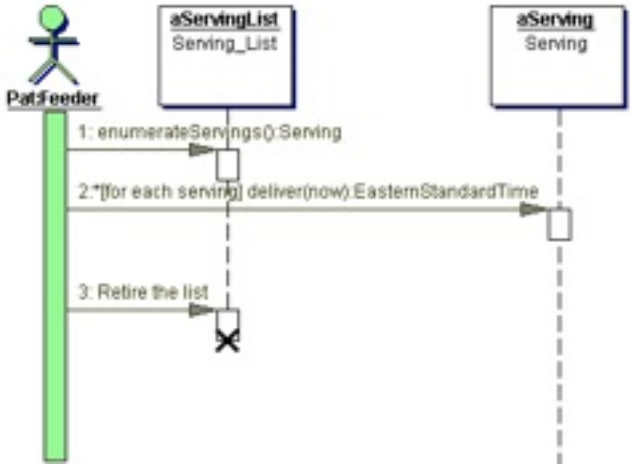
This is a collection of cages designated as an area for assigning zoo staff.

Attribute Summary	
private String	**areaName** This is the name of the zoo area which encloses a series of cages.
private Cage	**[association] lnkCage** This records the assignment of cages to an area.
private StaffMember	**[association] lnkFeeder** All areas have one or more staff assigned.
private Feeder	**[association] lnkFeeder1** There is one feeder employee assigned to each area of the zoo.

Service Summary	
public Cage	**enumerateCages**() This service successively returns a link to each of the cages belonging to this area.

Sequence Diagram *Feed the Animals*

package: <default>



This sequence diagram models the delivery and record keeping of the feeder in his/her feeding rounds.

Object Summary

- aServing
- aServingList
- Pat

Object Detail

Object *aServing*

A serving is a physical instance of food to be delivered to an animal in their cage.

Instantiates:
Serving

Object *aServingList*

A serving list is a collection of servings for a particular animal.

Instantiates:
ServingList

destroyed:
true

Object *Pat*

Pat is responsible for preparing the serving list of food for a particular area he/she is assigned.

Instantiates:
Feeder

Stereotype:
actor

backgroundColor:
153,255,153

Message Detail

to **Object** aServingList

Synchronization:
call

Number:
1

Operation:
ServingList.enumerateServings()

operationNameAsText:
'enumerateServings():void'

to **Object** aServing

Synchronization:
call

Number:
2

Operation:
Serving.deliver(EasternStandardTime)

operationNameAsText:
'deliver(EasternStandardTime):EasternStandardTime'

Iteration:
for each serving

Arguments:
now

Retire the list to **Object** aServingList

Synchronization:
call

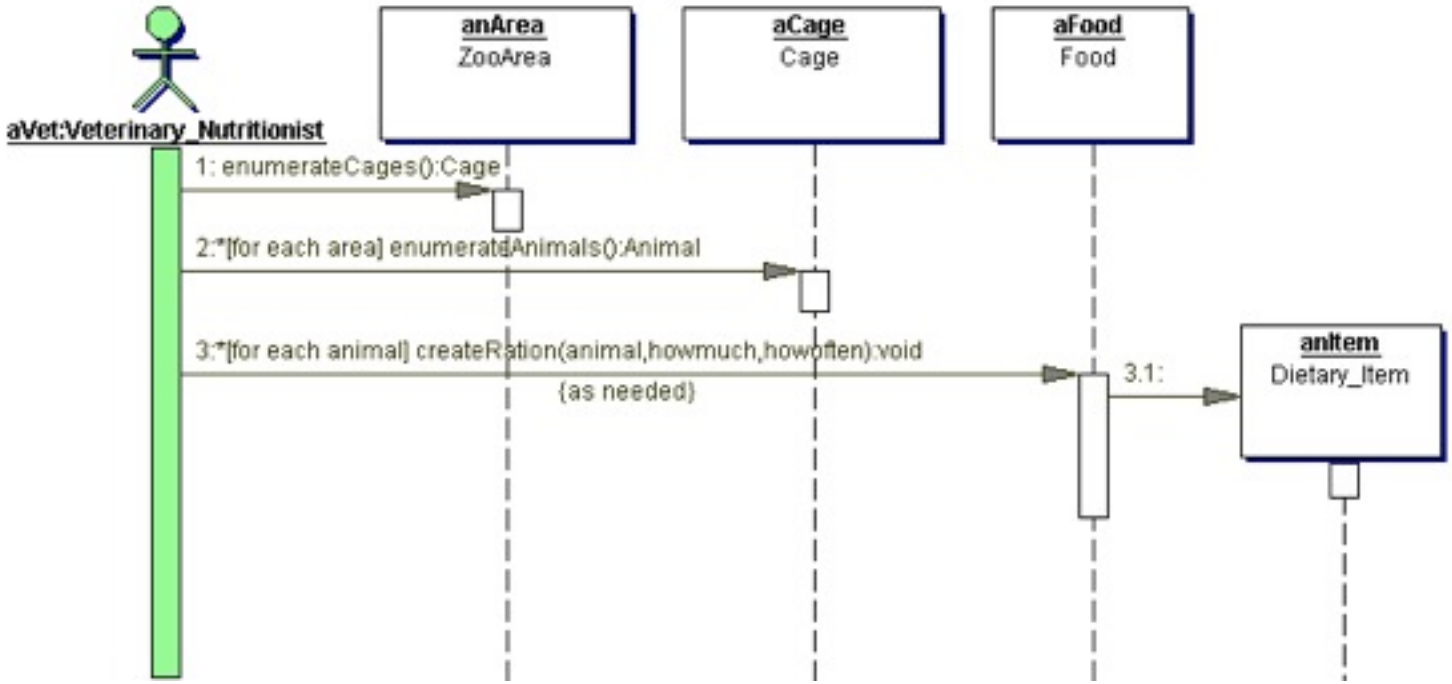
Number:
3

destruction message

Sequence Diagram *Prepare Animal Diets*

package: <default>

This sequence diagram models the process of assigning diets to each animal in the zoo.



Object Summary

- aCage
- aFood
- anArea
- anItem
- aVet

Object Detail

Object *aCage*

ACage knows the animals that inhabit it.

Instantiates:
Cage

Object *aFood*

AFood knows how to create a dietary item.

Instantiates:
Food

Message Detail

to **Object** anItem

Documentation:
A new dietary item is created for this animal with this food.

Synchronization:
call

Number:
3.1

creation message

Object *anArea*

AnArea knows its cages.

Instantiates:
ZooArea

Object *anItem*

Instantiates:
DietaryItem

created:
true

Object *aVet*

This is the staff member of the zoo responsible for defining the diet of each animal.

Instantiates:
VeterinaryNutritionist

Stereotype:
actor

backgroundColor:
153,255,153

Message Detail

to **Object** anArea

Documentation:
An area is requested to identify all the cages defined therein.

Synchronization:
call

Number:
1

Operation:
ZooArea.enumerateCages()

operationNameAsText:
'enumerateCages():Cage'

to **Object** aCage

Documentation:
The cage is requested to identify all its animal inhabitants.

Synchronization:
call

Number:
2

Operation:
Cage.enumerateAnimals()

operationNameAsText:
'enumerateAnimals():Animal'

Iteration:
for each area

to **Object** aFood

Documentation:
aFood is instructed to create a new dietary item for this animal.

Synchronization:
call

Number:
3

Operation:
Food.createRation(Animal,int,int)

operationNameAsText:
'createRation(Animal,int,int):void'

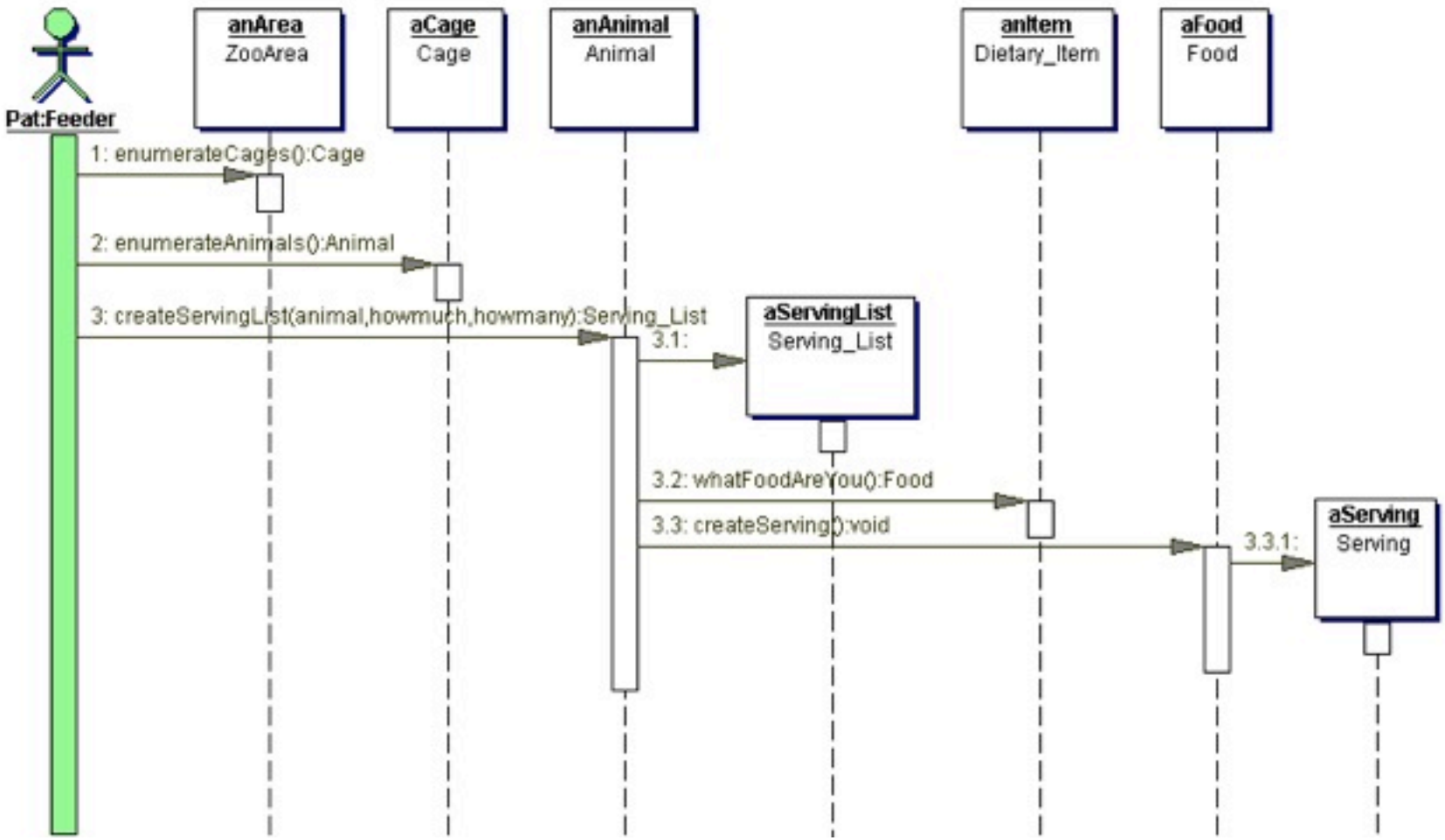
Iteration:
for each animal

Constraint:
as needed

Arguments:
animal,howmuch,howoften

Sequence Diagram *Prepare Serving List*

package: <default>



The feeder must build the list of servings to be prepared and later fed to the animals in his/her area of responsibility.

Object Summary

- aCage
- aFood
- anAnimal
- anArea
- anItem
- aServing
- aServingList
- Pat

Object Detail

Object *aCage*

A cage is responsible for knowing which animal are assigned to it.

Instantiates:
Cage

Object *aFood*

The food object is responsible for creating individual servings of itself.

Instantiates:
Food

Message Detail

to **Object** aServing

Synchronization:
call
Number:
3.3.1
creation message

Object *anAnimal*

An animal is responsible for creating a serving list that will hold all the servings for itself.

Instantiates:
Animal

Message Detail

to **Object** aServingList

Synchronization:
call
Number:
3.1
creation message

to **Object** anItem

Synchronization:
call
Number:
3.2
Operation:
DietaryItem.whatFoodAreYou()
operationNameAsText:
'whatFoodAreYou():void'

to **Object** aFood

Synchronization:
call
Number:
3.3
Operation:
Food.createServing()
operationNameAsText:
'createServing():void'

Object *anArea*

AnArea is responsible for knowing which cages are in it.

Instantiates:
ZooArea

Object *anItem*

A dietary item is responsible for knowing which food it is derived from.

Instantiates:
DietaryItem

Object *aServing*

A serving is a physical instance of food to be delivered to an animal in their cage.

Instantiates:
Serving

created:
true

Object *aServingList*

A serving list is a collection of servings for a particular animal.

Instantiates:
ServingList

created:
true

Object *Pat*

Pat is responsible for preparing the serving list of food for a particular area he/she is assigned.

Instantiates:
Feeder

Stereotype:
actor
backgroundColor:
153,255,153

Message Detail

to **Object** anArea

Synchronization:
call
Number:
1
Operation:
ZooArea.enumerateCages()
operationNameAsText:
'enumerateCages():Cage'

to **Object** aCage

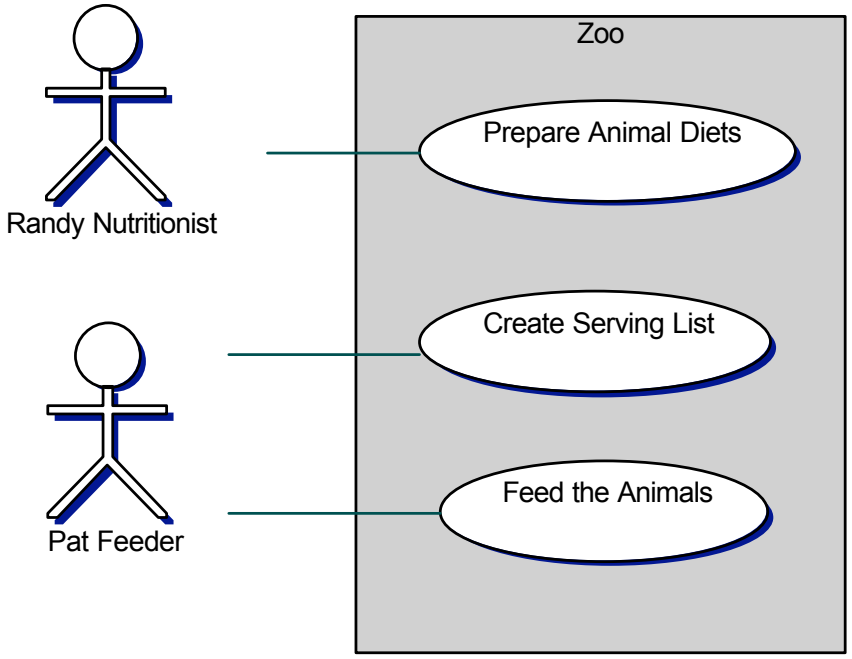
Synchronization:
call
Number:
2
Operation:
Cage.enumerateAnimals()
operationNameAsText:
'enumerateAnimals():Animal'

to **Object** anAnimal

Synchronization:
call
Number:
3
Operation:
Animal.createServingList(Animal,int,int)
operationNameAsText:
'createServingList(Animal,int,int):ServingList'
Arguments:
animal,howmuch,howmany

UseCase Diagrams *Feeding the Animals, Prepare Animal Diets, Create Serving List*

package: <default>



This use case describes the interaction of the Feeder with the zoo system to get the animals fed.

Diagram Contents Summary

Actor Pat Feeder

Actor Randy Nutritionist

System Boundary Zoo

UseCase Create Serving List

UseCase Feed the Animals

UseCase Prepare Animal Diets

Actor Detail

Actor *Pat Feeder*

"Pat" is a typical feeder employee of the zoo.

"Communicates" links

to **UseCase** Create Serving List

to **UseCase** Feed the Animals

Actor *Randy Nutritionist*

"Communicates" links

to **UseCase** Prepare Animal Diets

System Boundary Detail

System Boundary *Zoo*

The zoo system boundary represents the information system functions that support the zoo operations.

backgroundColor:
200,200,200

UseCases

UseCase *Create Serving List*

The feeder is responsible for feeding a group of animals housed in the part of the zoo for which he/she is responsible. This use case describes the "Pat" visible activities that the system exposes to Pat.

preconditions:

The employee is a feeder. All the animals have been assigned to cages. All the cages have been assigned to areas in the zoo. All the dietary items for each animal have been defined.

postconditions:

A serving list has been created that lists all animals in the feeder's area of responsibility. A complete list of serving objects has been created which satisfies the collecftive needs of the animals in the feeder's charge.

normalFlow:

1. Feeder gets a list of cages in the area he/she is responsible for.
2. Feeder gets list of animals in each of the cages in his/her area.
3. Feeder instructs each animal in his/her list to create an individual serving list using the defined diet.
4. The serving list for each animal is check against available food stores for adequacy.
5. The complete serving list is ready for scheduled delivery.

alternateFlow:

4.a There are insufficient food stores for a particular animal.
5.a Some animals are omitted from the final feeding list for lack of food.

UseCase *Feed the Animals*

preconditions:

The feeder has prepared a serving list for all animals in his/her area.

postconditions:

Every serving on the feeder's serving list has been delivered and the feeding times have been recorded.

normalFlow:

1. Iterate through the serving items in the serving list (these should be ordered by cages and areas).
2. deliver the serving to the animal.
3. Record the time the animal is fed.

UseCase *Prepare Animal Diets*

preconditions:

All animals have been assigned to cages. All cages have been assigned to areas. All necessary food stores have been defined.

postconditions:

Every animal has one or more defined dietary items to direct their feedings.

normalFlow:

1. Iterate through the areas.
2. Iterate through the cages.
3. for each animal create a dietary item for that animal based on available food stores.