Please, Press Ctrl-A, F9 to update all fields or move cursor over the field and press F9 to activate TOC

Root Package

This diagram is the fifth iteration of the ZooKeeper narrative intended to model the use of classes and behaviors to model the definition of model behavior based on Use Case. It includes three sequence diagram that model the implementation of the use cases.

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

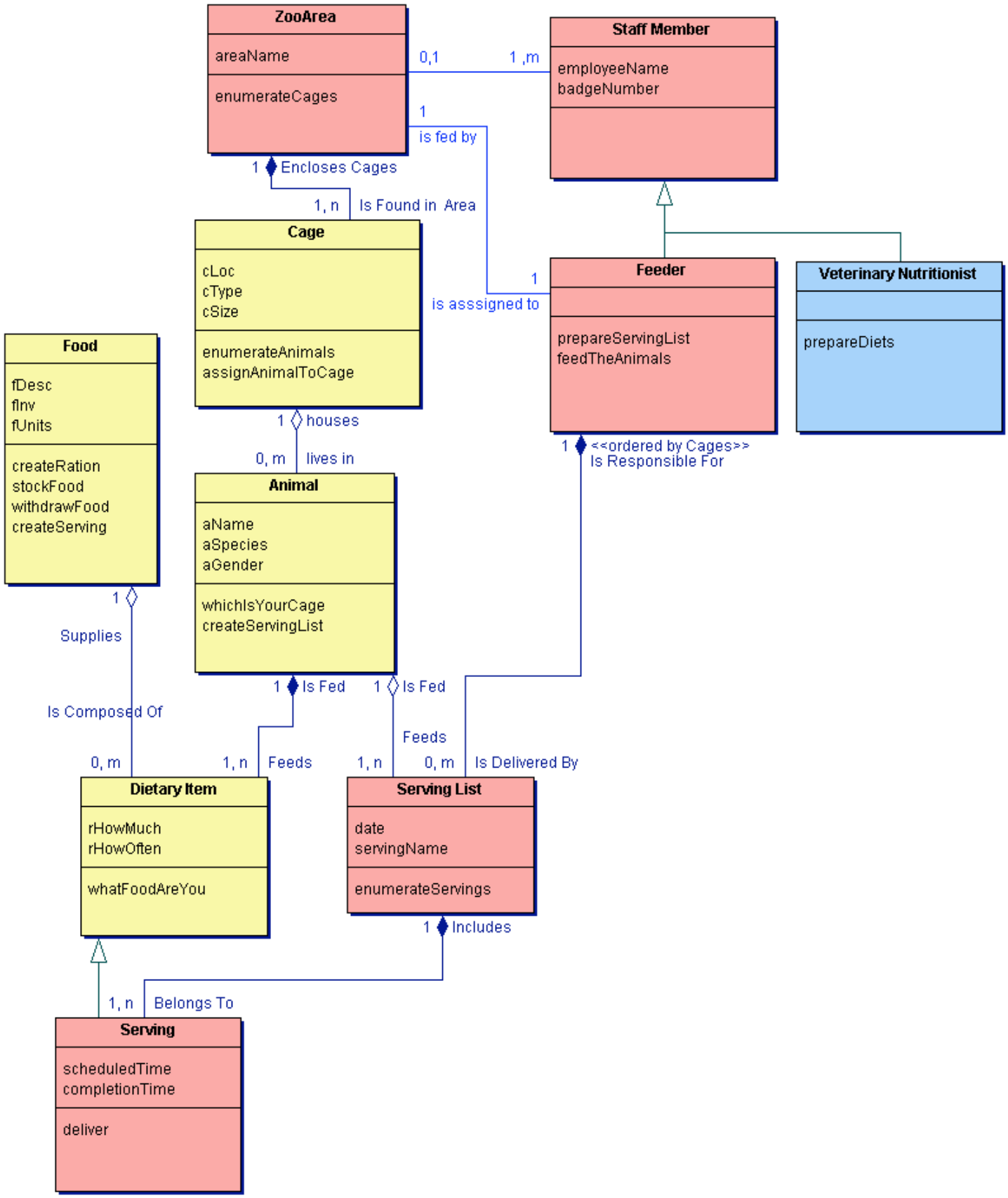
class Animal
class Cage
class Dietary_Item
class Feeder
class Food
class Serving
class Serving_List
class Staff_Member
class Veterinary_Nutritionist
class ZooArea

Class Diagrams

Class Diagram *<default>*

package: <default>

Model **ZooKeeper(Phase 5)** Tue Oct 23 14:15:16 EDT 2007



This diagram is the fifth iteration of the ZooKeeper narrative intended to model the use of classes and behaviors to model the definition of model behavior based on Use Case. It includes three sequence diagram that model the implementation of the use cases.

Class Nodes

Animal
Cage
Dietary_Item
Feeder
Food
Serving
Serving_List
Staff_Member
Veterinary_Nutritionist
ZooArea

Class Detail

Class *Animal*

```
public class Animal
```

This is an animal housed in the zoo.

Field 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 Dietary_Item	**lnkRation**
private Serving_List	**lnkServingList** This is a collection of one or more serving lists for a particular animal.

Method Summary	
public Serving_List	**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.

Field Detail

aGender

private int aGender

The gender of this animal.

---

aName

private int aName

The given name of the animal.

---

aSpecies

private int aSpecies

The biological species of this animal.

---

lnkRation

private Dietary_Item lnkRation

---

lnkServingList

private Serving_List lnkServingList

This is a collection of one or more serving lists for a particular animal.

Method Detail

createServingList

public `Serving_List 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.

---

whichIsYourCage

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.

Field 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	**lnkAnimal** A cage may be empty.

Method 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.

Field Detail

cLoc

private int cLoc

The location of the cage.

---

cSize

private int cSize

Cage size: small, medium, large.

---

cType

private int cType

Type of cage: moat, bars, unbarred.

---

lnkAnimal

private Animal lnkAnimal

A cage may be empty. Every animal must be in a cage.

Method Detail

assignAnimalToCage

public void assignAnimalToCage(Animal theAnimal)

This service allows a zookeeper to assign a particular animal to a particular cage.

---

enumerateAnimals

public Animal enumerateAnimals()

This service successively returns a link to each of the animals housed in it.

Class *Dietary_Item*

public class Dietary_Item

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

Field 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.

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

Field Detail

rHowMuch

private int rHowMuch

How many units of the designated food are alloted to one ration for the designated animal.

---

rHowOften

private int rHowOften

The number of times during the feeding period that this animal is given this ration (e.g. twice a week, everyday, etc.)

Method Detail

whatFoodAreYou

public Food whatFoodAreYou()

The dietary item identifies the food object to which it belongs.

Class *Feeder*

```
Staff_Member
  |
  +--Feeder
```

public class Feeder

Extends:

Staff_Member

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

Field Summary	
private Serving_List	**lnkServingList** A serving list is the sole responsibility of a single feeder staff member.

Method 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.

Field Detail

lnkServingList

private Serving_List lnkServingList

A serving list is the sole responsibility of a single feeder staff member. A feeder may be in the process of delivering a serving list or have completed same, thus having no current feeding list to work with.
The serving lists are ordered according to the adjacency of the cages in the area. This is accomplished by the order that the ZooArea returns each of the cages in its EnumerateCages service.

Method Detail

feedTheAnimals

public void feedTheAnimals()

This service actually brings the servings to each cage to feed the animals.

---

prepareServingList

public void prepareServingList(ZooArea theArea)

This service prepares a list of food servings derived from the dietary needs of each animal.

Class *Food*

```
public class Food
```

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

Field 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 Dietary_Item	**lnkRation** A collection of dietaryitem objects created from a specific food type.

Method 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.

Field Detail

fDesc

private int fDesc

A description of the food type (i.e. Meat, Fish, Grain, etc.)

---

fInv

private int fInv

The number of units of this food found in the food storage.

---

fUnits

private int fUnits

The type of units with which this food is measured.

---

lnkRation

private Dietary_Item lnkRation

A collection of dietaryitem objects created from a specific food type.

Method Detail

createRation

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.

---

createServing

public void createServing()

This service withdraws food from the food warehouse and prepares a single serving of same for its particular animal.

---

stockFood

public void stockFood()

This service updates the current inventory of this food when supplies are placed in the warehouse.

---

withdrawFood

public boolean withdrawFood()

This service notes the withdrawal of food of this type form the warehouse. If insufficient food is on hand the service fails.

Class *Serving*

```
Dietary_Item
  |
  +--Serving
```

public class Serving

Extends:

Dietary_Item

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

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

Method Summary	
public EasternStandardTime	**deliver**(EasternStandardTime Time)

Field Detail

completionTime

private EasternStandardTime completionTime

Time the serving was actually delivered.

---

scheduledTime

private EasterStandardTime scheduledTime

Time the serving is sheduled to be delivered.

Method Detail

deliver

public `EasternStandardTime` deliver(`EasternStandardTime` Time)

Class *Serving_List*

`public class Serving_List`

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

Field Summary	
private CalendarDay	**`date`** The calendar date that this serving list is intended to be fed to the animal.
private Serving	**`lnkRation`** A collection of servings to be delivered to a particular animal.
private String	**`servingName`** A string indicating the name of the serving list.

Method Summary	
public `Serving`	**`enumerateServings()`**

Field Detail

date

private CalendarDay date

The calendar date that this serving list is intended to be fed to the animal.

---

lnkRation

private Serving lnkRation

A collection of servings to be delivered to a particular animal.

---

servingName

private String servingName

A string indicating the name of the serving list.

Method Detail

enumerateServings

public `Serving` enumerateServings()

Class *Staff_Member*

`public class Staff_Member`

This is the general representation of a zoo staff member.

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

Field Detail

badgeNumber

private int badgeNumber

A unique identifying code used to verify employee identity.

---

employeeName

private int employeeName

Legal name of zoo staff member.

Class *Veterinary_Nutritionist*

```
Staff_Member
  |
  +--Veterinary_Nutritionist
```

```
public class Veterinary_Nutritionist
```

Extends:

Staff_Member

Method Summary	
public void	**prepareDiets()**

Method Detail

prepareDiets

public void prepareDiets()

Class *ZooArea*

```
public class ZooArea
```

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

Field Summary	
private String	**areaName** This is the name of the zoo area which encloses a series of cages.
private Cage	**lnkCage** This records the assignment of cages to an area.
private Staff_Member	**lnkFeeder** All areas have one or more staff assigned.

Field Summary	
private Feeder	**lnkFeeder1** There is one feeder employee assigned to each area of the zoo.

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

Field Detail

areaName

private String areaName

This is the name of the zoo area which encloses a series of cages.

---

lnkCage

private Cage lnkCage

This records the assignment of cages to an area. Every area has one or more cages. Every cage belongs to an area.

---

lnkFeeder

private Staff_Member lnkFeeder

All areas have one or more staff assigned. A staff member may or may not be assigned to a particular area.

---

lnkFeeder1

private Feeder lnkFeeder1

There is one feeder employee assigned to each area of the zoo. Although many staff members may actually participate in caring for the animals, one staff member, the feeder, is responsible for the preparation and delivery of their food.

Method Detail

enumerateCages

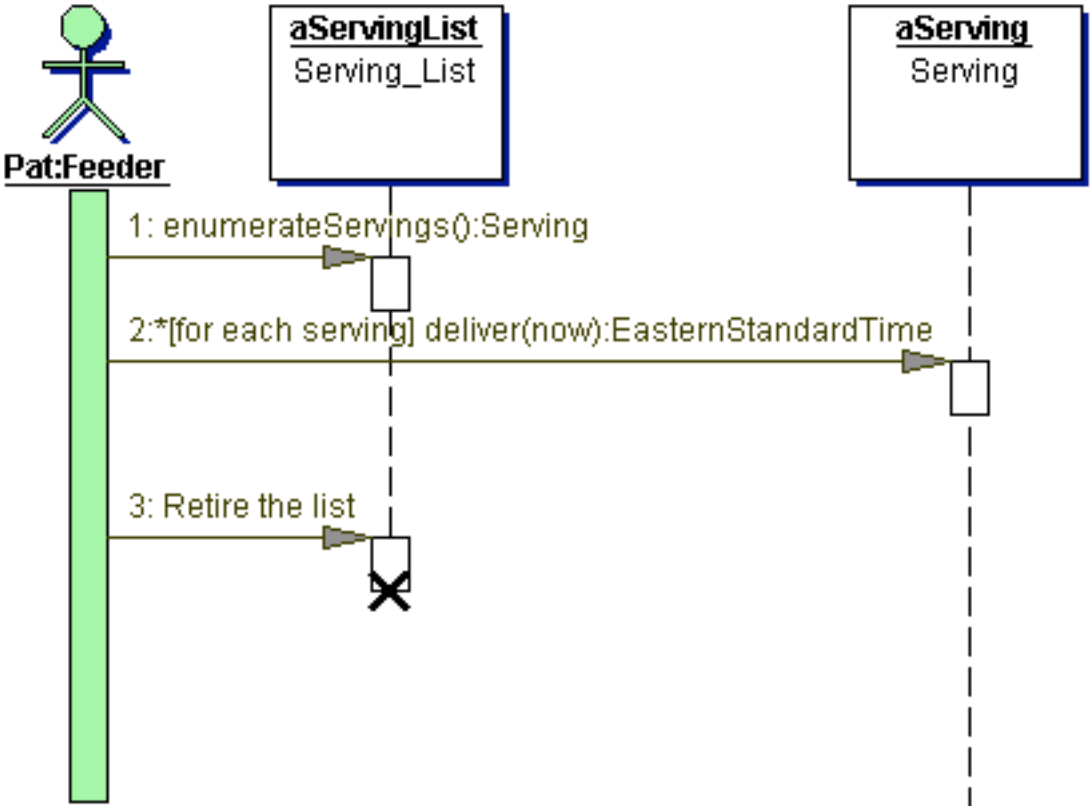
public Cage enumerateCages()

This service successively returns a link to each of the cages belonging to this area.

Interaction Diagrams

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 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:
Serving_List

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

diagram_uniqe_name:
<oiref:design#Class#ideu1d6e6ns2guue6nt759z.diagram:oiref>

Operation:
Serving_List.enumerateServings()

operationNameAsText:
'EnumerateServings():void'

normalizedIndex:
0.25

activationUid:
design:node:::ideu1d6e6ns2guue6nt759z.nodeidfppbke6ns2guue6nsgypv.nodeidijp1oe6ns2guue6nt9ioe:ideu1d6e6ns2guue6nt759z.nodeidfppbke6ns2guue6nsgypv

sendingInstant:
124

processingDuration:
20

to **Object** aServing

Synchronization:
call

Number:
2

diagram_uniqe_name:
<oiref:design#Class#ideu1d6e6ns2guue6nt759z.diagram:oiref>

Operation:
Serving.deliver(EasternStandardTime)

operationNameAsText:
'Deliver(EasternStandardTime):EasternStandardTime'

Iteration:
for each serving

Arguments:
now

normalizedIndex:
0.5

activationUid:
design:node:::ideu1d6e6ns2guue6nt759z.nodeidgr1xye6ns2guue6nsj0rn.nodeidjb2uge6ns2guue6ntcnau:ideu1d6e6ns2guue6nt759z.nodeidgr1xye6ns2guue6nsj0rn

sendingInstant:
163

processingDuration:
20

Retire the list to **Object** aServingList

Synchronization:
call

Number:
3

destruction message

diagram_uniqe_name:
<oiref:design#Class#ideu1d6e6ns2guue6nt759z.diagram:oiref>

normalizedIndex:
0.75

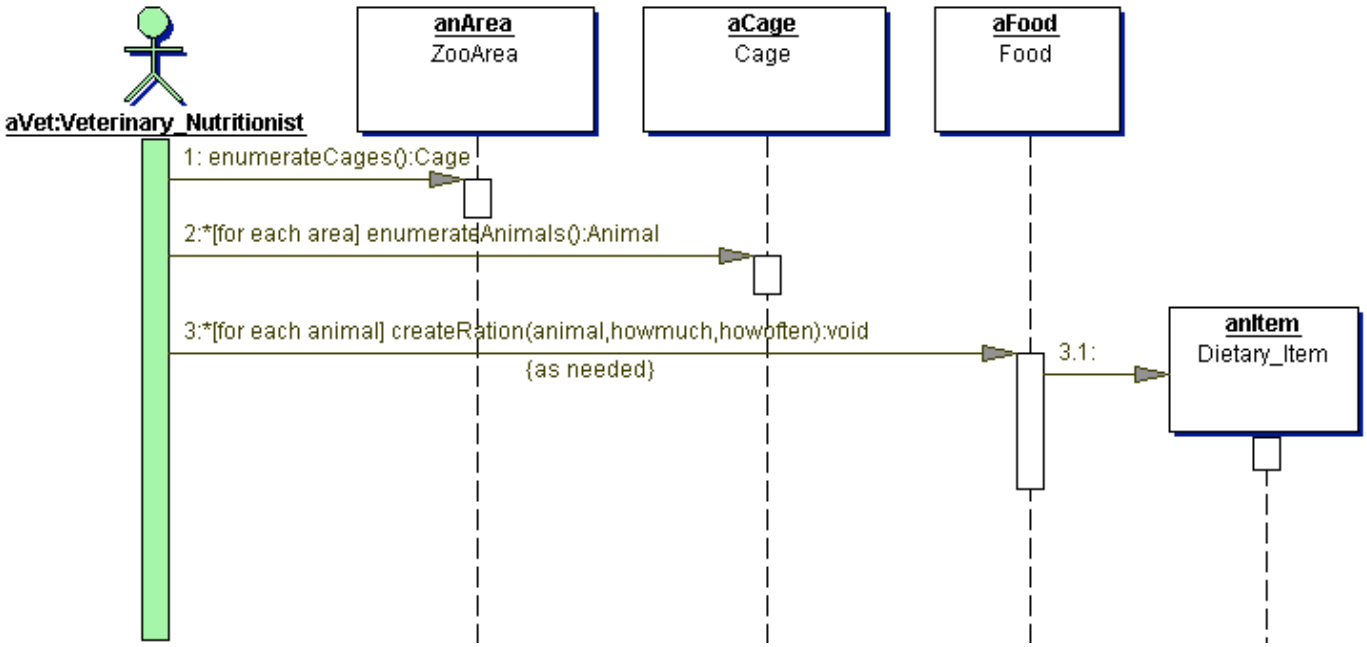
activationUid:
design:node:::ideu1d6e6ns2guue6nt759z.nodeidfppbke6ns2guue6nsgypv.nodeid4f5k2e6ns2guue6nted4j:ideu1d6e6ns2guue6nt759z.nodeidfppbke6ns2guue6nsgypv

sendingInstant:
229

processingDuration:
20

Sequence Diagram *Prepare Animal Diets*

package: <default>



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

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

diagram_uniqe_name:
<oiref:design#Class#id5g0ace6np30e5e6nr0dh8.diagram:oiref>

normalizedIndex:
0.5

predecessorUid:
design:link:::id5g0ace6np30e5e6nr0dh8.nodeid6a1wce6np30e5e6nroz1t.linkidc740ge6np30e5e6nrahih:id5g0ace6np30e5e6nr0dh8.nodeid6a1wce6np30e5e6nroz1t

activationUid:
design:node:::id5g0ace6np30e5e6nr0dh8.nodeid60yw1e6np30e5e6nrgicz.nodeid8uq4ee6np30e5e6nrheu5:id5g0ace6np30e5e6nr0dh8.nodeid60yw1e6np30e5e6nrgicz

sendingInstant:
222

processingDuration:
50

Object *anArea*

AnArea knows its cages.

Instantiates:
ZooArea

Object *anItem*

Instantiates:
Dietary_Item

created:
true

Object *aVet*

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

Instantiates:
Veterinary_Nutritionist

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

diagram_uniqe_name:
<oiref:design#Class#id5g0ace6np30e5e6nr0dh8.diagram:oiref>

Operation:
ZooArea.enumerateCages()

operationNameAsText:
'EnumerateCages():Cage'

normalizedIndex:
0.25

activationUid:
design:node:::id5g0ace6np30e5e6nr0dh8.nodeid9vh49e6np30e5e6nr36wv.nodeid9a6rhe6np30e5e6nr61j3:id5g0ace6np30e5e6nr0dh8.nodeid9vh49e6np30e5e6nr36wv

sendingInstant:
120

processingDuration:
20

to **Object** aCage

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

Synchronization:
call

Number:
2

diagram_uniqe_name:
<oiref:design#Class#id5g0ace6np30e5e6nr0dh8.diagram:oiref>

Operation:
Cage.enumerateAnimals()

operationNameAsText:
'EnumerateAnimals():Animal'

Iteration:
for each area

normalizedIndex:
0.5

activationUid:
design:node:::id5g0ace6np30e5e6nr0dh8.nodeid766vte6np30e5e6nr3udz.nodeid7e2qme6np30e5e6nr7rlo:id5g0ace6np30e5e6nr0dh8.nodeid766vte6np30e5e6nr3udz

sendingInstant:
160

processingDuration:
20

to **Object** aFood

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

Synchronization:
call

Number:
3

diagram_uniqe_name:
<oiref:design#Class#id5g0ace6np30e5e6nr0dh8.diagram:oiref>

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

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

Iteration:
for each animal

Constraint:
as needed

Arguments:
animal,howmuch,howoften

normalizedIndex:
0.75

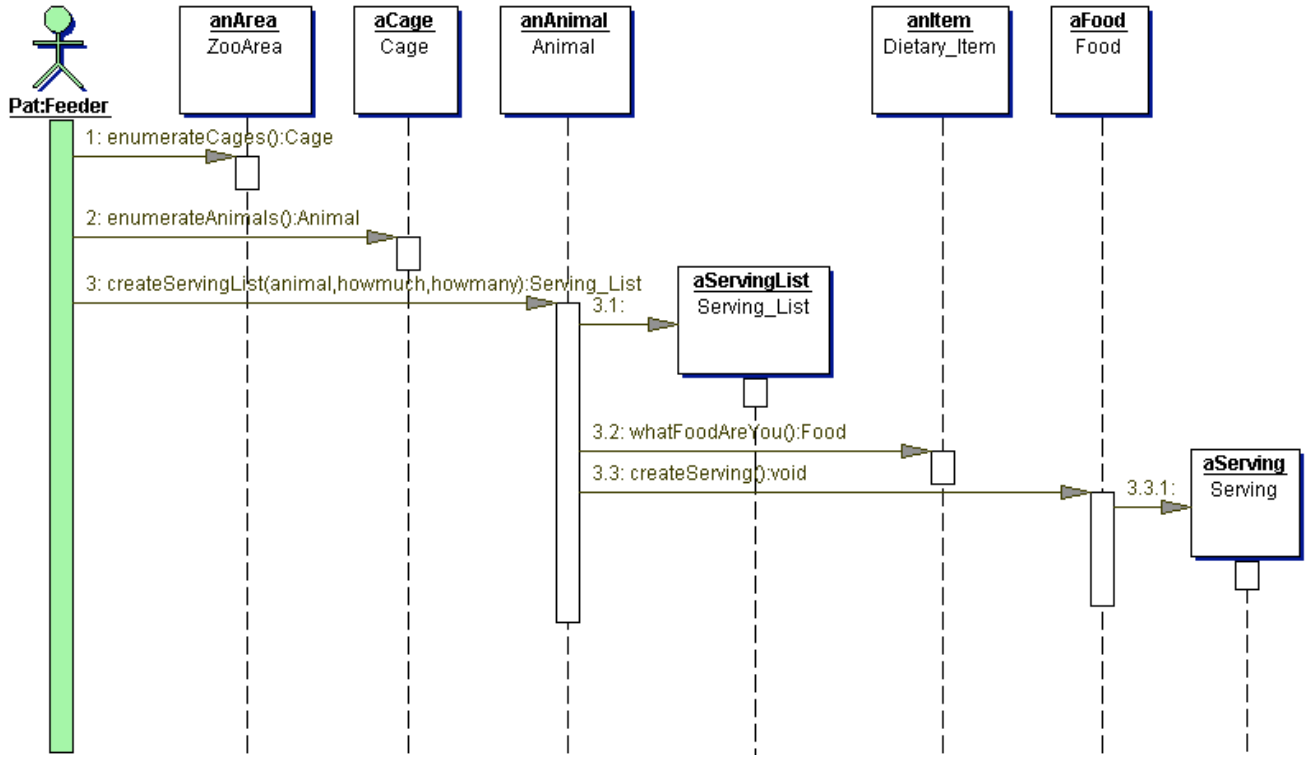
activationUid:
design:node:::id5g0ace6np30e5e6nr0dh8.nodeidbelcfe6np30e5e6nr5948.nodeid5yegoe6np30e5e6nrahl9:id5g0ace6np30e5e6nr0dh8.nodeidbelcfe6np30e5e6nr5948

sendingInstant:
211

processingDuration:
71

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 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

diagram_uniqe_name:
<oiref:design#Class#idzj74e6ns2guue6nse9sy.diagram:oiref>

normalizedIndex:
0.5

predecessorUid:
design:link:::idzj74e6ns2guue6nse9sy.nodeid9220ve6ns2guue6nsg2yt.linkid5iaa5e6ns2guue6nsq26r:idzj74e6ns2guue6nse9sy.nodeid9220ve6ns2guue6nsg2yt

activationUid:
design:node:::idzj74e6ns2guue6nse9sy.nodeidgr1xye6ns2guue6nsj0rn.nodeidevrppe6ns2guue6nsqgwy:idzj74e6ns2guue6nse9sy.nodeidgr1xye6ns2guue6nsj0rn

sendingInstant:
334

processingDuration:
50

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

diagram_uniqe_name:
<oiref:design#Class#idzj74e6ns2guue6nse9sy.diagram:oiref>

normalizedIndex:
0.25

predecessorUid:
design:link:::idzj74e6ns2guue6nse9sy.nodeiddo1jte6ns2guue6nsedtl.linkid8kdnme6ns2guue6nskycw:idzj74e6ns2guue6nse9sy.nodeiddo1jte6ns2guue6nsedtl

activationUid:
design:node:::idzj74e6ns2guue6nse9sy.nodeidfppbke6ns2guue6nsgypv.nodeid1xeese6ns2guue6nsme81:idzj74e6ns2guue6nse9sy.nodeidfppbke6ns2guue6nsgypv

sendingInstant:
223

processingDuration:
50

to **Object** anItem

Synchronization:
call

Number:
3.2

diagram_uniqe_name:
<oiref:design#Class#idzj74e6ns2guue6nse9sy.diagram:oiref>

Operation:
Dietary_Item.whatFoodAreYou()

operationNameAsText:
'WhatFoodAreYou():void'

normalizedIndex:
0.5

predecessorUid:
design:link:::idzj74e6ns2guue6nse9sy.nodeiddo1jte6ns2guue6nsedtl.linkid8kdnme6ns2guue6nskycw:idzj74e6ns2guue6nse9sy.nodeiddo1jte6ns2guue6nsedtl

activationUid:
design:node:::idzj74e6ns2guue6nse9sy.nodeidfrdyje6ns2guue6nshvz4.nodeidbyq4ae6ns2guue6nsnj9f:idzj74e6ns2guue6nse9sy.nodeidfrdyje6ns2guue6nshvz4

sendingInstant:
300

processingDuration:
20

to **Object** aFood

Synchronization:
call

Number:
3.3

diagram_uniqe_name:
<oiref:design#Class#idzj74e6ns2guue6nse9sy.diagram:oiref>

Operation:
Food.createServing()

operationNameAsText:
'CreateServing():void'

normalizedIndex:
0.75

predecessorUid:
design:link:::idzj74e6ns2guue6nse9sy.nodeiddo1jte6ns2guue6nsedtl.linkid8kdnme6ns2guue6nskycw:idzj74e6ns2guue6nse9sy.nodeiddo1jte6ns2guue6nsedtl

activationUid:
design:node:::idzj74e6ns2guue6nse9sy.nodeid15mfye6ns2guue6nsouns.nodeid5o5mye6ns2guue6nsq27l:idzj74e6ns2guue6nse9sy.nodeid15mfye6ns2guue6nsouns

sendingInstant:
325

processingDuration:
69

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:
Dietary_Item

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:
Serving_List

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

diagram_uniqe_name:
<oiref:design#Class#idzj74e6ns2guue6nse9sy.diagram:oiref>

Operation:
ZooArea.enumerateCages()

operationNameAsText:
'EnumerateCages():Cage'

normalizedIndex:
0.25

activationUid:
design:node:::idzj74e6ns2guue6nse9sy.nodeid4sbu1e6ns2guue6nsewf6.nodeid5b6qbe6ns2guue6nsk3k3:idzj74e6ns2guue6nse9sy.nodeid4sbu1e6ns2guue6nsewf6

sendingInstant:
121

processingDuration:
20

to **Object** aCage

Synchronization:
call

Number:
2

diagram_uniqe_name:
<oiref:design#Class#idzj74e6ns2guue6nse9sy.diagram:oiref>

Operation:
Cage.enumerateAnimals()

operationNameAsText:
'EnumerateAnimals():Animal'

normalizedIndex:
0.5

activationUid:
design:node:::idzj74e6ns2guue6nse9sy.nodeid7zgl5e6ns2guue6nsfgy0.nodeid8dn0ee6ns2guue6nskdi3:idzj74e6ns2guue6nse9sy.nodeid7zgl5e6ns2guue6nsfgy0

sendingInstant:
170

processingDuration:
20

to **Object** anAnimal

Synchronization:
call

Number:
3

diagram_uniqe_name:
<oiref:design#Class#idzj74e6ns2guue6nse9sy.diagram:oiref>

Operation:
Animal.createServingList(Animal,int,int)

operationNameAsText:
'CreateServingList(Animal,int,int):Serving_List'

Arguments:
animal,howmuch,howmany

normalizedIndex:
0.75

activationUid:
design:node:::idzj74e6ns2guue6nse9sy.nodeid9220ve6ns2guue6nsg2yt.nodeid8htt6e6ns2guue6nskyfe:idzj74e6ns2guue6nse9sy.nodeid9220ve6ns2guue6nsg2yt

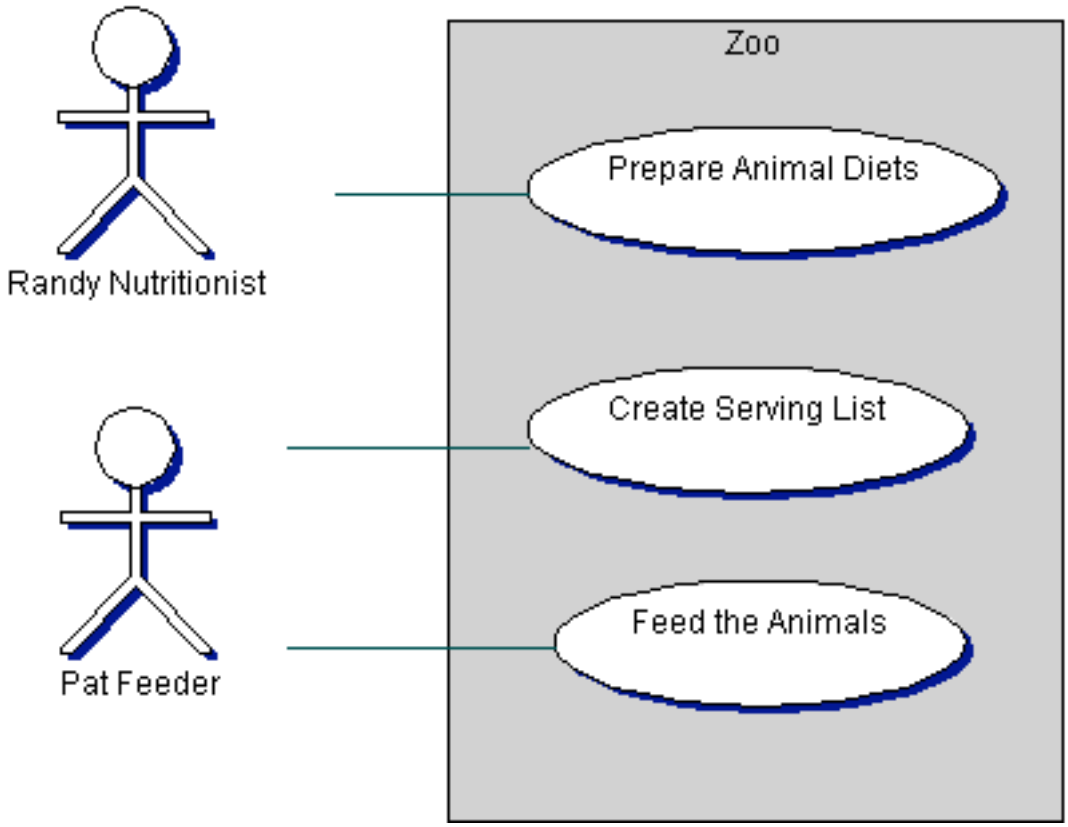
sendingInstant:
210

processingDuration:
194

UseCase Diagrams

UseCase Diagram *Feeding the Animals*

package: <default>



This use case describes the interaction of the Feeder with the zoo system in creating a list of animals to be 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
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.