

ASSIGNMENT OVERVIEW

In this assignment you'll be creating a class **Animal** which can be used by a Zookeeper to keep track of animals. You'll also be creating a subclass **Fish** that extends **Animal**, another **Animal** subclass of your own design, and a **Zookeeper** class that allows one to inventory the various animals. Finally, you'll write a **ZookeeperRunner** class that demonstrates your Zookeeper class.

This assignment is worth 50 points and is due on the *crashwhite.polytechnic.org* server at 23:59:59 on the date given in class.

BACKGROUND

All **Animals** share some characteristics as a superclass, but there are differences in some animals as well. By writing an **Animal** superclass and extending it to create subclasses, we can manage an entire range of creatures.

This assignment requires the use of two techniques that you may not have seen yet

1. A **static int** will be used to create a numerical sequence of **idTags** for the animals in our zoo. Each new animal that gets constructed will have an **idTag** value that is one greater than the animal before it.
2. An *enumerated type* will be used to identify the **Environment** of the **Fish** subclass, where each fish lives in a **SALTWATER** or **FRESHWATER** environment.

PROGRAM SPECIFICATION

For this assignment you will need to:

- a. Write an **Animal** class that has a single constructor with **idTag** (a static **int** that is incremented as new **Animals** are constructed), **animalType** (**String**) and **value** (**double**) instance variables, and **getValue**, **setValue**, **getIdTag**, and **getAnimalType** methods. **Animal** should also modify the **toString** method appropriately.

toString example:

```
"Animal[idTag=1004,animalType=Tiger,value=25000.0]"
```

- b. Write a **Fish** class that extends **Animal**, and includes an enumerated **Environment** that may be **FRESHWATER** or **SALTWATER** as an explicit construction parameter. (See the Assignment Notes below for information on how to implement this.) **Fish** should have a **getWaterType** method, and modify the **toString** method appropriately.

toString example:

```
"Fish[idTag=1005,animalType=Shark,value=1050.0][waterType=SALTWATER]"
```

- c. Write an additional class of your own choosing that extends **Animal**, and that includes at least one original instance variable associated with that class. Your additional class should have methods that interact with it, and a **toString** method that returns an appropriate result.

- d. Write a **Zookeeper** class that uses an **ArrayList** called **zoo** to manage a collection of **Animals**. Methods in this class include **addAnimal**, **getAnimalInventory**, and **findMostValuable**, which return the **Animal** object in the array that has the highest monetary value.

DELIVERABLES

Zookeeper.zip

This single file will be a zipped directory (folder) of your project. It will include:

- your **Animal.java** superclass
- your **Fish.java** subclass
- the **Animal** subclass you wrote independently
- a **Zookeeper.java** class
- a **ZookeeperRunner.java** class that can be used to test your files
- a **package.BlueJ** file (optional)

To submit your assignment for grading, copy your file to your directory in **/home/studentID/forInstructor/** at *crashwhite.polytechnic.org* before the deadline.

ASSIGNMENT NOTES

- Write the initial **Animal** class first, being sure to include instance fields (variables) and methods as needed. To initialize the static **idTag** field:

```
private static int idTagCurrent = 1000;
```

After that, in the constructor, the following two lines will allow us to continue constructing **Animals** with automatically created **idTags**:

```
this.idTag = idTagCurrent;  
idTagCurrent++;
```

- Write the **Fish** class to inherit from the **Animal** class. The **Fish** class inherits all the qualities of the **Animal** class, but as an additional attribute, **waterType**, which will have a value of either *saltwater* or *freshwater*.

What *type* should the variable **waterType** be? We could make it a **String**, but that's probably not the best design choice here—what if somebody misspells the **waterType**, or makes up a new **waterType** ("brackish") that our program is not designed to work with? We really want the **waterType** to be one of only two possible values, so to do this, we're going to establish an *enumerated type*, ie. a data type that can have one of only a few enumerated values. To implement this:

```
public class Fish extends Animal  
{  
    // Declare the enumerated type  
    // Note that enumerated values are in all caps, because  
    // they're constants and can't be changed.  
    public enum Environment {SALTWATER, FRESHWATER};  
}
```

```
// Declare the variable that will be of that type
private Environment waterType;

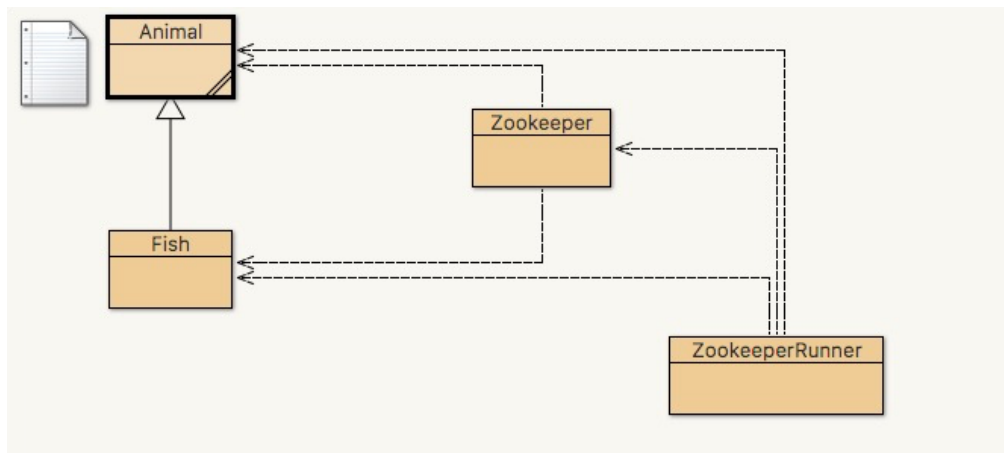
/**
 * Constructs a new fish of the given breed, value, and waterType
 * @param animalType typically a "shark","guppy","marlin",etc.
 * @param value the monetary value of the fish
 * @waterType either FRESHWATER or SALTWATER
 */

public Fish(String animalType, double value, Environment waterType )
{
    super(animalType, value);
    this.waterType = waterType;
}
}
```

- How can we use this new enumerated **Environment** in other classes or programs? They won't be aware of the type or its value unless we use a fully-qualified name:
Fish.Environment.SALTWATER, or **Fish.Environment.FRESHWATER**.
- The **Zookeeper.java** file allows us to manage the **Animals**. It establishes the **ArrayList zoo**, and defines the methods identified in the specification above that will allow us to manipulate the animals in the zoo.
What type of objects are we going to store in the **ArrayList zoo**? If we choose **Animal**, are we going to be able to store the **Fish** in there as well?

GETTING STARTED

1. With paper and pencil, and perhaps in collaboration with a partner, identify what the main components are that you'll need to include in your program. Have a reasonably clear idea of how your files will interact with each other before you start coding.



2. Sketch out the basic flow of your program using a flowchart, and write some pseudocode that you can use to begin implementing those main components.
3. Create an **Animal.java** class that you will use to construct and manipulate the Animals, and test it using BlueJ's Code Pad.

4. Enter pseudocode as comments in the editor, then fill in more details for various parts of the code.
5. Test each bit of code as you go, making sure that one piece works before you proceed on to the next section. You'll repeatedly run through this edit-compile-test, edit-compile-test process to progressively find bugs and fix them *while* you're writing your program, not afterwards.
6. Once a day or so, archive/zip your BlueJ Zookeeper folder and save a backup copy of it on another device or machine: a flash drive, your home folder on the *crashwhite.polytechnic.org* server, etc.
7. When your program is completed (but before the deadline), copy a final archived package to the server as indicated above.

QUESTIONS FOR YOU TO CONSIDER (NOT HAND IN)

1. The *enumerated data type* can come in handy for all sorts of thing, typically short lists for which the values won't be changing. Days of the week is a common example:

```
public enum Days {MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY,SUNDAY};
private Days today;
```

What other potential uses for the enumerated data type would you think might be useful?

SAMPLE TESTER

```
/**
 * The ZookeeperRunner runs the Zookeeper class to test management of
 * a series of Animals, some of them Fish.
 *
 * @author Richard White
 * @version 2016-01-06 (revised)
 */
public class ZookeeperRunner
{
    public static void main(String[] args)
    {
        System.out.println("Managing the Zoo with Zookeeper class");
        Zookeeper theZoo = new Zookeeper();
        Animal tiger = new Animal("tiger", 1500.00);
        theZoo.addAnimal(tiger);
        theZoo.addAnimal(new Fish("shark", 2000.00, Fish.Environment.SALTWATER));
        theZoo.addAnimal(new Animal("earthworm", 0.25));
        theZoo.addAnimal(new Fish("guppy", 0.10, Fish.Environment.FRESHWATER));
        System.out.println("The animals in the zoo include");
        System.out.println(theZoo.getAnimalInventory());
        System.out.println("The most valuable animal in the zoo is " +
                           theZoo.findMostValuable().getAnimalType());
    }
}
```

INTERACTIONS (from Tester above)

Managing the Zoo with Zookeeper class

The animals in the zoo include

ANIMAL INVENTORY

IdTag:1000

Animal:tiger

Value:1500.0

IdTag:1001

Animal:shark

Value:2000.0

WaterType:SALTWATER

IdTag:1002

Animal:earthworm

Value:0.25

IdTag:1003

Animal:guppy

Value:0.1

WaterType:FRESHWATER

The most valuable animal in the zoo is shark