Problem Description

Hello! Please make sure to read all parts of this document carefully.

In this assignment, you will be applying your knowledge of constructor chaining, the this keyword, encapsulation, method overloading, visibility modifiers, the toString method, and setters and getters. As we get further into the course, we are starting to learn more elements of Object-Oriented Programming. You will create a Frog.java, Fly.java, and Pond.java file that will simulate how they interact with each other in the real world.

Solution Description

Create files Fly.java, Frog.java, and Pond.java that will simulate how a pond ecosystem works. You will be creating a number of fields and methods for each file. Based on the description given for each variable and method, you will have to decide whether or not the variables/method should be static, and whether it should be private or public. To make these decisions, you should carefully follow the guidelines on these keywords as taught in lecture. In some cases, your program will still function with an incorrect keyword.

Fly.java

This Java file defines fly objects that exists within the pond.

Variables

All variables must be not allowed to be directly modified outside the class in which they are declared, unless otherwise stated in the description of the variable. Hint: there is a specific visibility modifier that can do this!

The Fly class must have these variables.

mass - the mass of the Fly in grams (it must allow decimal values)

speed - how quickly this Fly can maneuver through the air while flying, represented as a double

Constructors

You must use constructor chaining in at least two of your constructors. Duplicate code cannot exist in multiple constructors.

A constructor that takes in mass and speed of a Fly.

A constructor that takes in only mass.

By default, the Fly will have 10 speed.

A constructor that takes in no parameters.

By default, the Fly will have 5 mass and 10 speed.

Methods

Do not create any other methods than those specified. Any extra methods will result in point deductions. All methods must have the proper visibility to be used where it is specified they are used.

Setters and getters (using appropriately named methods) for all variables in Fly.java.

toString - takes in no parameters and returns a String describing the Fly as follows:

(Note: replace the values in brackets [] with the actual value. Do not include the double quotes “” or the square brackets [] in the output. Specify all decimal values to 2 decimal points.)

If mass is 0: “I’m dead, but I used to be a fly with a speed of [speed].”

Otherwise: “I’m a speedy fly with [speed] speed and [mass] mass.”

grow - takes in an integer parameter representing the added mass. Then it increases the mass of the Fly by the given number of mass. As mass increases, speed changes as follows:

If mass is less than 20: increases speed by 1 for every mass the Fly grows until it reaches 20 mass.

If the mass is 20 or more: decreases speed by 0.5 for each mass unit added over 20.

isDead – if mass is zero, return true. Otherwise, return false.

**Frog.java**

This Java file defines frog objects that exist within the pond.

Variables

All variables must be not allowed to be directly modified outside the class in which they are declared, unless otherwise stated in the description of the variable. Hint: there is a specific visibility modifier that can do this!

The Frog class must have these variables:

name - the name of this Frog, which can be made of any combination of characters

age - the age of the Frog as an integer number of months

tongueSpeed - how quickly this Frog’s tongue can shoot out of its mouth, represented as a double

isFroglet - a value that represents if this Frog is young enough to be a froglet (the stage between tadpole and adult frog), which must only have two possible values - true or false. A Frog is a froglet if it is more than 1 month old but fewer than 7 months old. Whenever age is changed, this variable must be updated accordingly.

species - the name of the species of this Frog, which must be the same for all instances of Frog (Hint: there is a keyword you can use to accomplish this). By default, its value must be “Rare Pepe”.

Constructors

You must use constructor chaining in at least two of your constructors. Duplicate code cannot exist in multiple constructors.

A constructor that takes in name, age, and tongueSpeed and sets all variables appropriately.

A constructor that takes in name, ageInYears representing the age of the Frog in years as a double, and tongueSpeed and sets all variables appropriately.

When converting ageInYears to age (in an integer number of months), round down to the nearest month without using any method calls (Hint: Java can automatically do this for you with casting).

A constructor that takes in just a name.

By default, a Frog is 5 months old and has a tongueSpeed of 5.

Methods

You must use method overloading at least once. Do not create any other methods than those specified. Any extra methods will result in point deductions. All methods must have the proper visibility to be used where it is specified they are used.

grow - takes in a whole number parameter representing the number of months.

Then it ages the Frog by the given number of months and increases tongueSpeed by 1 for every month the Frog grows until it becomes 12 months old.

If the Frog is 30 months old or more, then decrease tongueSpeed by 1 for every month that it ages beyond 30 months.

You must not decrease tongueSpeed to less than 5.

Remember to update isFroglet accordingly

grow - takes in no parameters and ages the Frog by one month and updates tongueSpeed accordingly as for the other grow method

eat – takes in a parameter of a Fly to attempt to catch and eat.

Check if Fly is dead, and if it is dead then terminate the method.

The Fly is caught if tongueSpeed is greater than the speed of the Fly.

If the Fly is caught and the mass is at least 0.5 times the age of the Frog, the Frog ages by one month using the method grow. If the Fly is caught, the mass of the Fly must be set to 0.

If the Fly is NOT caught, the mass of the Fly must be increased by 1 while updating the speed of the Fly using only one Fly method.

toString - returns a String describing the Frog as follows:

(Note: replace the values in brackets [] with the actual value. Do not include the double quotes “” or the square brackets [] in the output. Specify all decimal values to 2 decimal points.)

If frog is a froglet, returns “My name is [name] and I’m a rare froglet! I’m [age] months old and my tongue has a speed of [tongueSpeed].”

Otherwise, returns “My name is [name] and I’m a rare frog. I’m [age] months old and my tongue has a speed of [tongueSpeed].”

Setter and getter for species which must change the value for all instances of the class. Points will be deducted if you include an unnecessary or inappropriate setter/getter.

Pond.java

This Java file is a driver, meaning it will contain and run Frog and Fly objects and “drive” their values according to a simulated set of actions. You can also use it to test your code. We require the following in your Pond class:

Methods

Main method must act as such:

Must create at least 4 Frog objects:

Frog with name Peepo.

Frog with name Pepe, age 10 months, and tongueSpeed of 15.

Frog with name Peepaw, age 4.6 years, and tongueSpeed of 5.

Frog of your liking :)

Must create at least 3 Fly objects:

Fly with 1 mass and speed of 3.

Fly with 6 mass.

Fly of your liking :)

Perform the following operations in this order:

Set the species of any Frog to “1331 Frogs”

Print out on a new line the description of the Frog named Peepo given by the toString method.

Have the Frog named Peepo attempt to eat the Fly with a mass of 6.

Print out on a new line the description of the Fly with a mass of 6 given by the toString method.

Have the Frog named Peepo grow by 8 months.

Have the Frog named Peepo attempt to eat the Fly with a mass of 6.

Print out on a new line the description of the Fly with a mass of 6 given by the toString method.

Print out on a new line the description of the Frog named Peepo given by the toString method.

Print out on a new line the description of your own Frog given by the toString method.

Have the Frog named Peepaw grow by 4 months.

Print out on a new line the description of the Frog named Peepaw given by the toString method.

Print out on a new line the description of the Frog named Pepe given by the toString method.

Allowed Imports

To prevent trivialization of the assignment, you are not allowed to import any classes or packages.

Feature Restrictions

There are a few features and methods in Java that overly simplify the concepts we are trying to teach or break our auto grader. For that reason, do not use any of the following in your final submission:

var (the reserved keyword)

System.exit