Lab 13

Classes and Objects

Objectives of this Lab:

1. To understand the “Big Project 1” and get started on your ideas.
2. Make the transition to Object Oriented Programming.
3. Go over some basic OOP concepts in Java:
    a. Classes
    b. Instance Variables
    c. Methods
    d. Object state
    e. The keyword “this” and what it does
    f. toString method

Introduce “Big Project 1”

Your TAs will go over the expectations for Big Project 1. This is your time to ask questions. This is the first part of a two part assignment. You will need to complete this first part before you can move on to the second part. This will be due on Monday, July 29th at 11:59pm to GitHub.

NOTE: We will be checking for plagiarism, so do your own work! You can talk about conceptual ideas with others, but don’t share code!

Introduction

Phase 1

Review the following Java programs as an example of a Java class and client code that uses it: MimicOct.java and UnderTheSea.java.

You have been working with classes in Java. One of the most important things to know about classes is that they are actually blueprints for objects. They basically contain all of the information as to how objects are built.

One of the most important features of objects is the fact that they have state and behavior. The way that they hold state is through instance variables. Instance variables are simply variables that "describe" the object and its state. There are several ways to handle the accessibility of an instance variable within an object, but in this lab we are going to work only with public instance variables. Public means that any class within the project that you're working in has permission to access and modify the instance variables. To access or modify an instance variable that is public, such as populationNumber, in an object called species, simply type species.populationNumber = someInteger.

Methods are the ways that objects perform their behavior. Our focus will be on non-static methods; these require an instance of the class in order to operate. Static methods are methods that don't need an instance of
an object to call it.

Phase 2

Please download Species.java and CircleOfLife.java. Use CircleOfLife.java for testing Species.java as you follow these steps:

Phase 3

Create a Species constructor that takes in a String for its name, an int for its population, and an int for its growth rate. Don't let someone initialize a species with a population above 1500 or below 1, or a growth rate outside of the range of 1 to 20 percent. If the population is initialized below 1, set it to 1, and if above 1500, set it to 1500. Similarly for the growth rate.

Phase 4

Return an appropriate String in the toString method that describes the state of a Species object.
Example String to return in the toString() method:
Name of species: cat
Population: 1200
Growth Rate: 12%
Once you have created a toString method, you can use it to display a Species object stored in a variable s as follows:
System.out.println(s);
which is a shortcut for
System.out.println(s.toString());

Phase 5

Create two species objects of whatever animal that you wish (do this in CircleOfLife.java).

Phase 6

Complete the code for **mergeSpecies(Species other)**. This is a non-static method (as all of the rest of them are) that takes another species as a parameter. This method adds the populations of the two species, changes the name of the species to the concatenation of the two names, and the growth rate to the maximum of the two growth rates

Phase 7

Complete the code for **grow()** and **populationInXYears(int x)** (hint: don't change the value of the population for the populationInXYears method -- simply return what the population will be in X years given the growth rate of that species.

EXAMPLE

If the species are rabbits and their growth rate is 10%, population is 100, and x is 2, you should see

The projected population for the rabbits in 2 years will be 121.

Grading

You will have 2 lab periods to complete this lab for full credit.