Introduction to Java

CS9053 Section I2

Wednesday 6 PM – 8:30 PM

Prof. Dean Christakos

Feb 16th, 2024

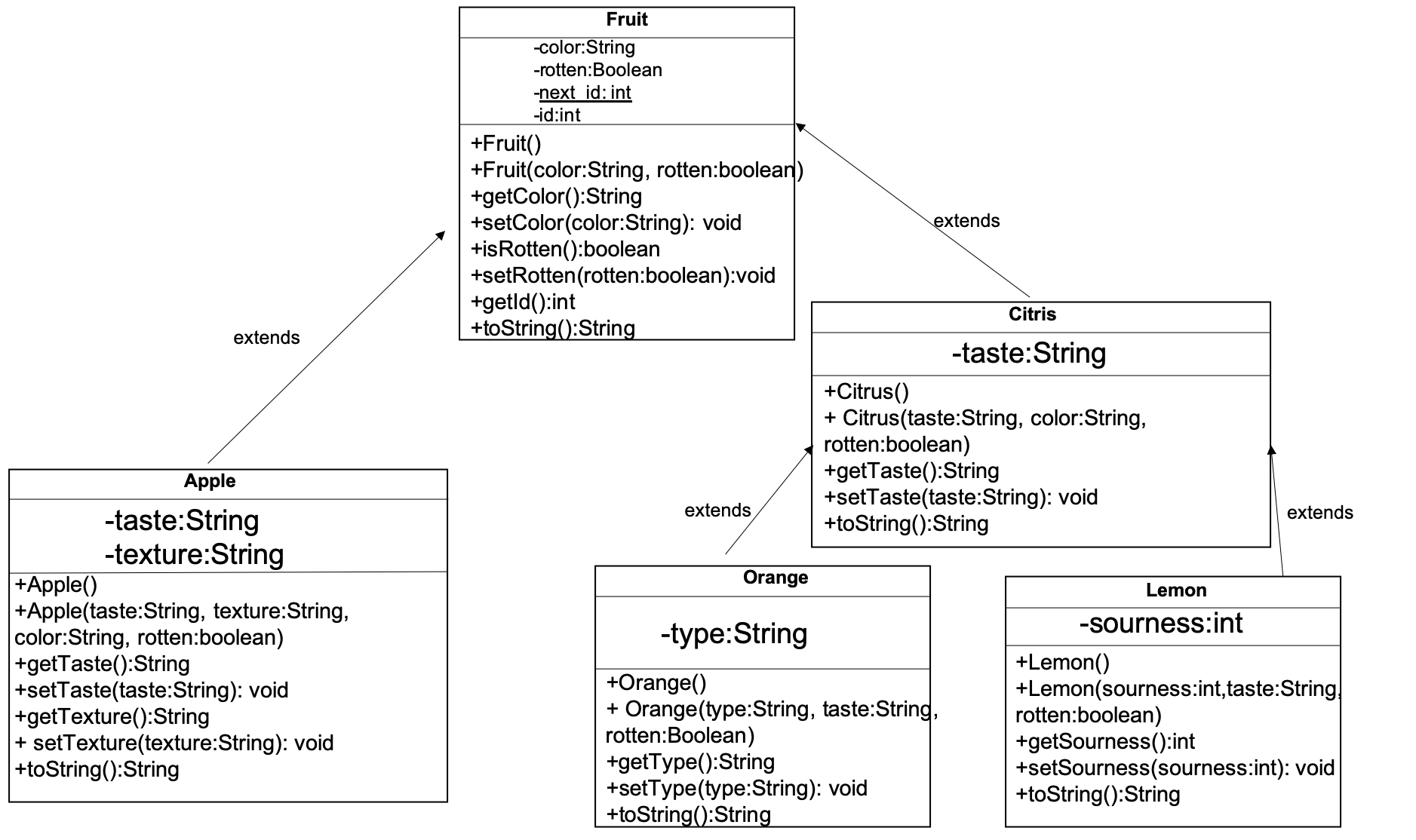
Due: Feb 23rd, 2024

**Assignment 4**

Part I: Inheritance

1. Here is a UML hierarchy of Fruits. Your first step is to implement this hierarchy. In constructors with args, you should use super() constructor with args to set the values in superclasses.

toString() should contain the name of the class and all of the fields/data for that class. You can decide whether this will work by a subclass calling the toString method of the superclass and including that in the subclass toString result (as in the GeometricObject hierarchy) or if the toString method accesses all the data/fields in the object itself.



Orange objects are always “orange” in color. Lemon objects are always “yellow” in color.

This uses the static “id” pattern. Every creation of a new Fruit object (of any kind) should generate a new and unique id from the static next\_id field, which is stored in the id field and accessible by getId

(Note: It’s spelled “Citrus”, and I spelled it wrong in the UML)

1. Now that you’ve implemented this, write equals methods for all the classes. The equals method should take an Object as an argument and return true if the field values of the class and its superclass(es) are equal.

Part II – Arraylists

1. Create an ArrayList of Fruit objects.
2. **Create 8 Fruit objects:**

1 non-rotten red Apple with a crisp texture and sweet taste

2 rotten green Apple objects with a soft texture and tart taste

3 non-rotten Lemon objects with a sour taste. “sourness” should be a random integer from 0-100 for each object

2 rotten Orange objects of type “mandarin” with a sweet taste.

Put those objects in the Arraylist. They should all be able to be added to the same Arraylist, regardless of their subclass. The Arraylist MUST be parameterized to accept Fruit objects. It should accept all Fruit objects but should not accept non-fruit objects. For example, this:

fruitArrayList.add(**new** Object());

should not compile

1. Print out the average sourness of all the Lemon objects in the Arraylist. You have to do this by looping through the array list, finding the Lemon objects, and their sourness
2. **Remove the matching objects:** Retain the 1st rotten green Apple object in an Apple variable. The goal is ultimately to remove all of the Apple objects in the Arraylist that match this variable:

To start: Loop through the Arraylist and print out (using toString) which objects in the Apple object is equal (in value) to the Apple object in your variable.

Also print out which object in the Arraylist is the **same object** as the one in your variable.

**You must figure out how to remove all the matching objects from the Arraylist.** There is no one correct way to do this. But there are incorrect ways.

1. **Print out the remaining objects:** Loop through the Arraylist again and print out (using toString) all the remaining objects in the Arraylist.