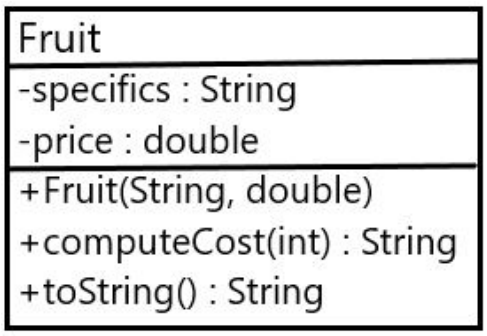
Lesson 4

Mark Eric Dieckmann

February 8, 2023

Lesson 4 revises inner classes. Like inheritance, inner classes allow us to structure code such that it becomes easier to read and handle. We cover static and non-static inner classes and test if the global variables and the methods of the outer class are visible in the inner class.

1 Task: Implement the outer class Fruit



specifics provides a compact description of the fruit.

price is the price per fruit.

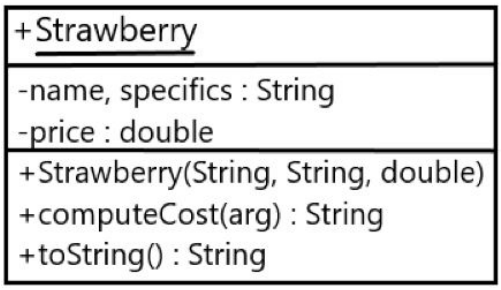
Fruit(arg1, arg2) initializes both instance variables with its arguments.

computeCost(arg) returns "Fruit: " followed by the product $arg \cdot price$.

toString() returns the formatted string "Specifics:" followed by *specifics* on 7 characters followed by ", Price: " followed by *price* on 5 characters and two after the comma.

2 Task: Create the static inner class Strawberry

Static inner classes can be used even if there is no instance of the outer class. We can thus not use the instance variables and instance methods of the outer class inside the inner class. There is only one static class in the heap but we can create many instances with it. That is different from class variables and methods, where only one copy exists. A static inner class is like a class file on the disk.



name, *specifics*, and *price* are the instance variables of the strawberry.

Strawberry(arg1, arg2, arg3) initializes the instance variables with the arguments.

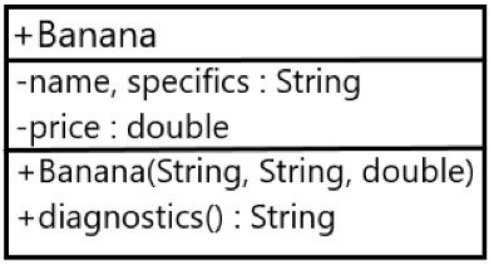
computeCost(arg) returns "Strawberry: " followed by $arg \cdot price$.

toString() should call the *toString()* method of the outer class. How do we get it? Remember that instance methods get compiled when we create an instance.

Uncomment the first block in **Lesson4** to test your subclass.

3 Task: Create the inner classes Apple and Banana

We need to have an instance of the outer class to initialize the inner classes. Uncomment the second block in **Lesson4** to create and test the instance of **Fruit**.



Apple does not declare its own variables *specifics* and *price* so it gets them from the outer class. **Banana** declares its own variables, which hides the ones from the outer class. Like when local variables hide global ones inside a method (we test that).

Apple has a *toString()* method, which creates the formatted string "Name:" followed by *name* on 7 characters followed by "Specifics:" followed by *specifics* on 7 characters followed by "Price:" followed by *price* with format 5.2f. Append to it the string " (outer class call) " followed by a call to the *toString()* method of the outer class. Create for this purpose a reference variable with the address of the outer class, which you can use like *super* in the inner class.

Apple has no own *computeCost(arg)* method and it can therefore not hide the one from the outer class. We check if this makes a difference. Uncomment the third block in **Lesson4** to check if the inner class can access *computeCost(arg)* of the outer class.

Banana has the method *diagnostics()* which returns the string "Diagnostics: " followed by the values of *name, specifics, price*. Uncomment the fourth block in **Lesson4** to check if the inner class hides the variable values of the outer class and to see what the *toString()* method gives.