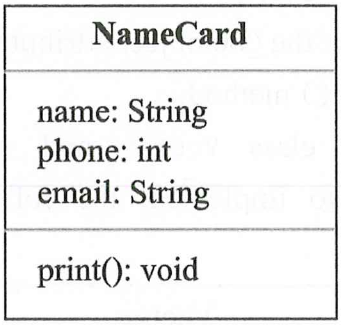
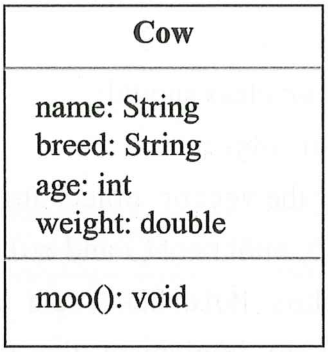
9. Write a Java class `NameCard` and a Java tester class `NameCardTester` to implement the following `NameCard` class design:



- Apply the encapsulation concept in your classes.
- The `print()` method should display information about a `NameCard` object.
- The `NameCardTester` class should:
  - o Make a `NameCard` object;
  - o Set values for the `NameCard` object attributes;
  - o Call the `print()` method.

10. Write a Java class `Cow` and a Java tester class `CowTester` to implement the following `Cow` class design:



- Apply the encapsulation concept in your classes.

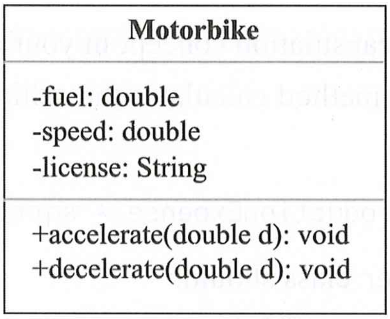
- The `moo()` method should display the text "Moo…".
- The `CowTester` class should:
  - o Make a `Cow` object;
  - o Set values of the `Cow` object attributes;
  - o Call the `moo()` method.

11. Write a Java class `Vector` and a Java tester class `VectorTester` to implement the following `Vector` class design:

Vector
x: double y: double
add(Vector v): Vector subtract(Vector v): Vector multiply(Vector v): Vector

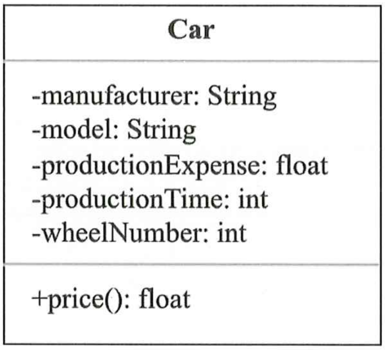
- Apply the encapsulation concept in your classes.
- The `add()`, `subtract()`, and `multiply()` methods return the addition, subtraction, and multiplication of two vectors, respectively.
- The `VectorTester` class should:
  - o Make a `Vector` object;
  - o Set values for the `Vector` object attributes;
  - o Call the `add()`, `subtract()` and `multiply()` methods.

12. Write a Java class `Motorbike` and a Java tester class `MotorbikeTester` to implement the following `Motorbike` class design:



- Apply the encapsulation concept in your classes.
- The `accelerate()` method increases the speed of the motorbike with a value given by its argument.
- The `decelerate()` method decreases the speed of the motorbike with a value given by its argument.
- The `MotorbikeTester` class should:
  - Make a `Motorbike` object;
  - Set values for the `Motorbike` object attributes;
  - Call the `accelerate()` and `decelerate()` methods;
  - Display the new state of the `Motorbike` object after calling the `accelerate()` and `decelerate()` methods.

13. Create a Java class `Car` and a Java tester class `CarTester` to implement the following `Car` class design:



- Apply the encapsulation concept in your classes.
- The `price()` method calculates the selling price of a car as follows:

```
price = 2 * productionExpense * sqrt(productionTime)
```

- The `CarTester` class should:
  - o Create an array of 5 `Car` objects;
  - o Display the information of all `Car` objects and the corresponding selling prices in the created array to the standard output.