1 Explain why the following implemention is incorrect for a constructor of the `Car` class. (3)

```
public Car {
  private int nDoors;   // How many doors does the car have?
  private Color color;  // What color is the car?

  public void Car(int doors, Color color) {
    nDoors = doors;
    color = color;
  }

  /* Additional attributes and methods not shown. */
}
```

2 What happens when the following method is run using the `Car` class from Question #1. (You may assume all problems with the class constructor have been resolved.) (3)

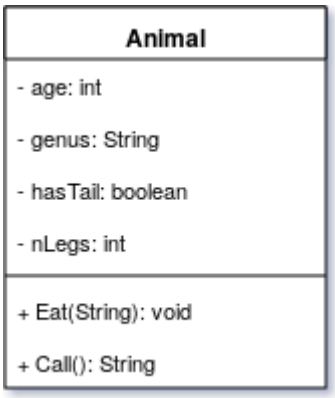
```
public static void main(String[] args) {
  // Create a new red, four-door car.
  Car car = new Car(4, Color.RED);

  // Paint the car blue!
  car.color = Color.BLUE;
}
```

3 Java requires that all programming code be part of a class. Some other object-oriented programming languages, such as C++, allow for "global" variables and methods; that is, C++ allows for programming code to exist outside of container classes. In your own words, what might be some benefits/drawbacks to Java's approach? (5)

4 For each of the following examples, indicate whether the described method or attribute should be `static` or non-`static`. (5)

a. A method of the `Employee` class that increases an employee's salary by 10%.
b. An attribute of the `Cat` class that contains a cat's genus.
c. A method of the `Television` class that changes a TVs channel.
d. A method of the `Algebra` class that solves a given (i.e., as a parameter) polynomial equation.
e. An attribute of the `Person` class that contains a person's height.

5 Implement the following class in Java. Ensure that the visibility of all attributes are `private` and all appropriate accessor methods are implemented. (10)



6 Create the `Point`, `Line`, and `Geometry` classes with the following specifications. (20)

`Point`:

- Contains two attributes: `x` and `y` to represent the rectangular coordinates of the point.
- Contains the appropriate constructor for a given set of coordinates.
- Contains the method: `distanceTo(Point p)` that will calculate the distance to the passed point, `p`.

`Line`:

- Contains two attributes: `m` and `b` to represent the slope and y-intercept of the line.
  Note: `b` should be a `Point` object.
- Contains an overloaded constructor for each of the following possible set of given values:
  - Slope, y-Intercept
  - Slope, Any point
  - Any two points
- Contains the method: `contains(Point p)` that will return `true` if the line contains the point, `p`, and `false` otherwise.

`Geometry`:

- Contains the method: `distance(Point p1, Point p2)` that will return the distance between the two given points.
- Contains the method: `midpoint(Point p1, Point p2)` that will return a `Point` that is the midpoint between the two given points.
- Contains the method: `perpendicularLine(Line l, Point p)` that will return a `Line` that is perpendicular to `l` that passes through `p`.