CIS 1068 Assignment 9
Warm Up with Objects

Due: Tuesday, March 29

70 points

Cars (35 points)

Implement a class `Car`, which contains the fields (5 points):

- `make`, e.g. Ford, Subaru, Toyota ...
- `model`, e.g., Escape, Outback, Camry ...
- `year`
- `MPG` miles per gallon
- `milesDriven`, the total number of miles ever driven in this car.
- `fuelCapacity` in gallons, i.e., the size in gallons of the fuel tank.
- `fuelRemaining`, which represents the amount of fuel remaining in the gas tank.

Implement at least the following methods within the `Car` class (5 points each):

- a constructor, which initializes each of the fields
- `fillTank(double g)`, which adds up to `g` gallons of gas to the fuel tank, but not more than the car's fuel capacity.
- `drive(double m)`, which simulates driving `m` miles in the car, adding to the total number of miles driven, and reducing the amount of gas in the car according to this car's average `MPG`.
- `toString( )`, which returns a `String` representation of the car.
- `getFuelRemaining( )`, which returns the amount of fuel left in the tank.

For example, we should be able to do something like the following:

```
Car oldJunker = new Car("Ford", "Pinto", 1972, 17.5, 132480, 12, 8); // creates a new Car object
oldJunker.drive(5); // drives the Car 5 miles
oldJunker.fillTank(1); // put in a gallon of gas
System.out.println(oldJunker.getFuelRemaining()); // prints the amount of fuel left
System.out.println(oldJunker); // prints the attributes of the car to the screen
```

Write a short driver program to test your Car class with a short array of Cars (5 points).

Fractions (35 points)

Write a class used to represent fractions. It should contains fields for numerator and denominator, and it should be able to perform some simple arithmetic operations.

The fields (5 points) are:

- numerator
- denominator

Implement at least the following methods:

`public Fraction(int n, int d)`
: (5 points) Constructor that creates a Fraction with numerator and denominator d. If d is 0, throw an ArithmeticException. We haven't yet had a chance to cover exceptions in as much depth as we soon will, but this can be done simply in your method with the statement `throw new ArithmeticException();`

`public int getNum()`

(2.5 points) Returns the value of the numerator field

`public int getDenom()`

(2.5 points) Returns the value of the denominator field

`public void setNum(int n)`

(2.5 points) Sets the numerator field to the value given in n

`public void setDenom(int d)`

(2.5 points) Sets the denominator field to the value given in d. If d is 0, throw an ArithmeticException.

`public Fraction add(Fraction a)`

(5 points) Returns the fraction that is the sum of the subject of the method and a. For example (new Fraction(3,4)).add(new Fraction(1,4)) is 16/16 i.e 1/1 We sum the fractions a/b and c/d as (a*d+b*c)/b*d then reduce.

`public boolean equals(Fraction a)`

(5 points) Returns true if subject of method and argument of call are equal. Fractions a/b and c/d are equal if a*d and b*c are equal or since the fractions are normalized, if a==c and b==d.

`public String toString()`

(5 points) Returns a `String` representation of the fraction. For example, if the numerator is 1 and the denominator is 2, the `String` "1/2" is returned.

Fractions should be stored in reduced form. For example, 2/4, 3/6, 4/8, etc. should all be stored as 1/2. 4/10 should be stored as 2/5. Use the Euclidean Algorithm for determining the greatest common divisor, so that you can store fractions in reduced form.

Write a simple main driver method to test your fraction class and each of its methods (5 points).

Note about files

There are several possible ways to organize your files for this assignment. The simplest and recommended way would be that you have four files:

1. **`Car.java`** - contains a `Car` class definition (i.e., what we've been calling a blueprint or a cookie cutter)
2. **`CarMain.java`** - contains your test program
3. **`Fraction.java`** - contains a `Fraction` class definition
4. **`FractionMain.java`** - contains a test program