CS 141: Introduction to Computer Science
Winter Term, 2024

Class Implementation Practice Problems

Implement the classes below.

1. A class `Car` that models an automobile. It should have three integer attributes `originalValue`, `speed` and `damage`. These will store the value of the car if it is undamaged, its current speed, and the number of dollars worth of damage it has sustained. Write the following methods for this class:
   - A constructor that takes one integer argument called `value`. Your constructor should set the `originalValue` attribute to the constructor's argument and set the other two attributes to 0.
   - A method `getSpeed` that returns the car's current speed.
   - A method `accelerate` that takes an integer which is the amount to increase the speed by.
   - A method `crash`, which increases damage by 20 times the speed unless doing so makes it greater than the value. In this case, the damage should be made equal to the original value. The method should also change the speed to zero.
   - A method `currentValue`, which takes no arguments and returns the current value of the car, i.e. its original value minus the amount of damage it currently has.
2. A `Train` class to model trains. (This is especially important since we live in Galesburg.) The train's attributes are its number of locomotives and number of cars. (For the purposes of this project, locomotives are not counted as cars.) Write the entire class, including the following methods:
   - A constructor that creates a train with a single locomotive and no cars.
   - A method `increaseSize` that takes a number of cars to add. In addition, one locomotive is added for every 5 cars added during this call (rounded down).
   - A method `speed` that gives the train's speed in miles per hour, with fractional speeds allowed. The train's speed in miles per hour is 20 minus the ratio of its number of cars over the number of locomotives. For example, a train with 2 locomotives and 7 cars should be 16.5 miles per hour.
   - A method `whistleNoise` that takes no arguments and gives the noise made by the train's whistle (a whole number in unspecified units). The noise is 5 times the number of locomotives.
3. A `Plane` class with the following methods:
   - A constructor that takes no arguments and creates a plane that is empty. The gas tank (with capacity 1000 gallons) is full, however.
   - `changePassengers`, called when the plane lands and takes on new passengers. All previous passengers get off and the method takes an argument giving the number of passengers who replace them.
   - `travel` makes the `Plane` fly to a new destination. It takes an `int` giving the distance traveled in miles. The plane uses a gallon of fuel for each mile travled.
   - `costToFuel` takes the current price per gallon of fuel and returns the amount it would cost to fill the plane's fuel tank. (Both amounts are in cents— the computer doesn't know this (or care), but you need to so you can interpret the program's output.)
   - `buyFuel`, which is called when the owners decide to fill the plane's fuel tank. It takes an integer argument giving the current price of a gallon of fuel in cents.
   - `profitSoFar` takes no arguments and returns the amount of profit the `Plane` has earned since its creation. This profit is the amount passengers pay for flying minus the cost of purchased fuel. The plane earns 10 cents per mile traveled for each passenger carried.

4. The `VacuumCleaner` class that models a rechargable vacuum cleaner with a fixed capacity for dirt. The class has the following methods:

   - A constructor that takes an integer argument and creates a `VacuumCleaner` that is fully charged and has a dirt capacity equal to the argument. The argument is assumed to be positive.
   - A method `use` that takes an integer which is the amount of dirt that is being cleaned. This amount is assumed to be positive. If the vacuum cleaner is charged, it cleans up to the minimum of the argument value and its remaining capacity (i.e. its overall capacity minus the amount of dirt it already contains). If the vacuum cleaner is uncharged when the method is called, no dirt is cleaned. The amount of dirt cleaned is added to the vacuum cleaner's current contents and returned by the method. The vacuum also becomes uncharged.
   - A method `charge` that takes no arguments and makes the vacuum fully charged.
   - A method `empty` that takes no arguments and causes all the dirt in the vacuum cleaner to be removed. It returns the amount of dirt removed.