#

# Assignment:

#

# Implement the methods as specfied in the following class, plus any you need to use to make your life easier.

# Make explicit any dependencies on external libraries so that I can run your code locally.

#

# Los Angeles recently sampled water quality at various sites, and recorded the

# presence of contaminants. Here's an excerpt of the table:

# (from: http://file.lacounty.gov/bc/q3\_2010/cms1\_148456.pdf)

# (All chemical values are in mg/L)

# | id | site | chloroform | bromoform | bromodichloromethane | dibromichloromethane |

# | 1 | LA Aquaduct Filteration Plant Effluent | .00104 | .00000 | .00149 | .00275 |

# | 2 | North Hollywood Pump Station (well blend) | .00291 | .00487 | .00547 | .0109 |

# | 3 | Jensen Plant Effluent | .00065 | .00856 | .0013 | .00428 |

# | 4 | Weymouth Plant Effluent | .00971 | .00317 | .00931 | .0116 |

#

# These four chemical compounds are collectively regulated by the EPA as Trihalomethanes,

# and their collective concentration cannot exceed .080 mg/L

#

# (from http://water.epa.gov/drink/contaminants/index.cfm#List )

class WaterSample

# This class intends to ease the managing of the collected sample data,

# and assist in computing factors of the data.

#

# The schema it must interact with and some sample data should be delivered

# with your assignment as a MySQL dump

def self.find(sample\_id)

# spec

# sample2 = WaterSample.find(2)

# sample2.site.should == "North Hollywood Pump Station (well blend)")

# sample2.chloroform.should == 0.00291

# sample2.bromoform.should == 0.00487

# sample2.bromodichloromethane.should == 0.00547

# sample2.dibromichloromethane.should == 0.0109

end

# Some Trihalomethanes are nastier than others, bromodichloromethane and

# bromoform are particularly bad. That is, water that has .060 mg/L of

# Bromoform is generally more dangerous than water that has .060 mg/L of

# Chloroform, though both are considered "safe enough".

#

# We could build a better metric by adjusting the contribution of each

# component to the Trihalomethane limit based on it's relative "danger".

# Furthermore, consider we want to try several different combinations of

# weights (factors).

#

# Sample table of Factor weights:

# | id | chloroform\_weight | bromoform\_weight | bromodichloromethane\_weight | dibromichloromethane\_wieight |

# | 1 | 0.8 | 1.2 | 1.5 | 0.7 |

# | 2 | 1.0 | 1.0 | 1.0 | 1.0 |

# | 3 | 0.9 | 1.1 | 1.3 | 0.6 |

# | 4 | 0.0 | 1.0 | 1.0 | 1.7 |

#

# In statistics, a factor is a single value representing a combination of

# several component values. We may gather several different variables, which

# semantically indicate a similar idea, and, to make analysis simpler, we can

# combine these several values into a single "factor" and disregard the

# constituents

#

# The weights that we should use in our factor could be a complex question.

# Ultimately it depends on what we're modeling.

#

# For example, let's say the city has the option of installing one of several

# different filtration units to remove a specific Triahlomethane (but the city

# can't afford all of the filters). We can use differently weighted factors to

# simulate each of these and do a cost / benefit analysis informing the city's

# decision on which filtration unit to purchase.

#

# Let's say someone from the city has already computed various factors they

# want to analyze, and put them in a factor\_weights table.

# In our case, we'll use a linear combination of the nth factor weights to compute the

# samples nth factor.

#

#

#

# Return the value of the computed factor with id of factor\_weights\_id

def factor(factor\_weights\_id)

# spec:

# sample2 = WaterSample.find(2)

# sample2.factor(6) #computes the 6th factor of sample #2

# => .0213

# Note that the factor for this example is from data not in the sample data

# above, that's because I want you to be sure you understand how to compute

# this value conceptually.

end

# convert the object to a hash

# if include\_factors is true, inlcude all computed factors in the hash

def to\_hash(include\_factors = false)

# spec:

# sample2.to\_hash

# => {:id =>2, :site => "North Hollywood Pump Station (well blend)", :chloroform => .00291, :bromoform => .00487, :bromodichloromethane => .00547 , :dibromichlormethane => .0109}

# sample2.to\_hash(true)

# #let's say only 3 factors exist in our factors table, with ids of 5, 6, and 9

# => {:id =>2, :site => "North Hollywood Pump Station (well blend)", :chloroform => .00291, :bromoform => .00487, :bro modichloromethane => .00547 , :dibromichlormethane => .0109, :factor\_5 => .0213, :factor\_6 => .0432, :factor\_9 => 0.0321}

end

end