Задание.

Создайте набор классов, описывающих обмен валют. Пункт обмена валют обслуживает клиентов по разным схемам: обмен может быть прямой, с наценкой, с наценкой и начислением налога. Клиент имеет определенный баланс в валюте на своем счету и может обменять его на другую валюту в соответствии с правилами обменного пункт. Классы должны накапливать статистику по обменным операциям, доходу пункта обмены и накопленной сумме налога.

Решение.

```
################################################################################
###
# Класс обменного пункта
################################################################################
###
class ExchangeOffice
  # "статический" в терминологии C++ метод класса
  # (а на самом деле - синглетон-метод объекта ExchangeOffice объекта Class)
  # очистка статистики
  def self.reset
    # хэш для количеств дохода в различных валютах
    @@profits = {}
    # хэш для количеств налога в различных валютах
    @@taxes = {}
  end

  # добавление дохода val в указанной валюте cur в статистику
  def self.add_profit cur, val
    # Если в кэше нет записи для валюты cur
    if @@profits[ cur ].nil?
      # Создаем запись
      @@profits[ cur ] = val
    # Если в кэше уже есть запись для указанной валюты
    else
      # Добавляем доход в нужную запись
      @@profits[ cur ] += val
    end
  end

  # добавление налога val в указанной валюте cur в статистику
  def self.add_tax cur, val
    # Если в кэше нет записи для валюты cur
    if @@taxes[ cur ].nil?
      # Создаем запись
      @@taxes[ cur ] = val
```

```
    # Если в хэше уже есть запись для указанной валюты
    else
      # Добавляем налог в нужную запись
      @@taxes[ cur ] += val
    end
  end

  # распечатка доходов во всех валютах
  def self.stat_profits
    puts "  profits:"
    # Обход пар хэша доходов и вывод сумм доходов во всех валютах
    @@profits.each_pair { |key, val|
      puts "    \"#{key}\": #{val.to_s}"
    }
  end

  # распечатка налогов во всех валютах
  def self.stat_taxes
    puts "  taxes:"
    # Обход пар хэша налогов и вывод сумм налогов во всех валютах
    @@taxes.each_pair { |key, val|
      puts "    \"#{key}\": #{val.to_s}"
    }
  end

  # распечатка статистики (доходов и налогов) обменного пункта
  def self.stat
    puts "Exchange office's current state:"
    self.stat_profits
    self.stat_taxes
    puts
  end

  # при инициализации (переопределнии) класса очищаем
  # (или инициализируем) статистику
  self.reset
end
##############################################################################
###

##############################################################################
###
# Класс прямого обмена
class DirectExchange
  # Приказ создать геттеры для параметров
  attr_reader :curIn
  attr_reader :curOut
  attr_reader :exVal

  # очистка статистики
  def self.reset
    # список обменных операций
    @@exchanges = []
```

```
  end

  # при инициализации (переопределнии) класса очищаем
  # (или инициализируем) статистику
  self.reset


  # конструктор
  # получает curIn - строка названия принимаемой для обмена валюты
  #          curOut - строка названия выдаваемой в ходе обмена валюты
  #          exVal - курс обмена (количество curOut, получаенной за единицу
curIn)
  def initialize curIn, curOut, exVal
    @curIn = curIn
    @curOut = curOut
    @exVal = exVal
    # ключ для хеша - строка из всех параметров
    # это же представление будем выдавать в качестве строкового представления
    # в методе to_s
    @hash_key = "\"#{@curIn}\"-> \"#{@curOut}\": #{@exVal}"
  end

  # формирование строки-записи об операции обмена количества валюты s
  def ex_op_desc s
    "\"#{@curIn}\" -> \"#{@curOut}\": amount #{s}"
  end

  # Преобразование объекта в строковое описание
  def to_s
    @hash_key
  end

  # Добавление описания операции простого обмена из строки str
  def self.add_op_desc str
    @@exchanges.push str
  end

  # метод расчета результата обмена количества s валюты curIn на валюту
curOut
  # по данной схеме. Результат: пара типа полученной валюты и ее количества
  def calc_change_res s
    # Возврат пары: типа полученной валюты и ее количества
    [ @curOut, s * @exVal ]
  end

  # метод обмена количества s валюты curIn на валюту curOut по данной схеме
обмена
  # с занесением необходимых данных в статистику
  # и начислением доходов и налогов
  def change s
    # Описание операции
    op_desc = ex_op_desc s
    # Добавление описания операции в список операций обмена
```

```
    DirectExchange.add_op_desc op_desc
    # Результат операции
    calc_change_res s
  end

  # вывод информации обо всех обменных операциях
  def self.stat
    puts "All exchanges:"
    # Вывод описаний всех операций
    @@exchanges.each{ |str|
      puts "  " + str
    }
    puts
  end

  # вывод информации обо всех доходах пункта обмена
  def self.income
    ExchangeOffice.stat_profits
  end

  # вывод информации обо всех налогах пункта обмена
  def self.tax
    ExchangeOffice.stat_taxes
  end
end
##############################################################################
###

##############################################################################
###
# Класс метода обмена с наценкой - потомок DirectExchange
class ExchangeWithCharge < DirectExchange
  # Приказ создать геттер для добавленного параметра
  attr_reader :charge

  # очистка статистики
  def self.reset
    # список обменных операций с начислением наценки
    @@exchanges_with_charge = []
  end

  # при инициализации (переопределнии) класса очищаем
  # (или инициализируем) статистику
  self.reset

  # конструктор
  # получает curIn - строка названия принимаемой для обмена валюты
  #          curOut - строка названия выдаваемой в ходе обмена валюты
  #          exVal - курс обмена (количество curOut, получаенной за единицу
curIn)
  #          charge - значение из [0.0; 1.0] - наценка
  def initialize curIn, curOut, exVal, charge
    # вызов конструктора DirectExchange
```

```
    super curIn, curOut, exVal
    @charge = charge
    # ключ для хеша - строка из всех параметров
    # это же представление будем выдавать в качестве строкового представления
    # в методе to_s
    @hash_key = "\"#{@curIn}\"-> \"#{@curOut}\" (charge #{@charge}):
#{@exVal}"
  end

  # формирование строки-записи об операции обмена количества валюты s с
наценкой
  def ex_op_desc s
    "\"#{@curIn}\" -> \"#{@curOut}\": amount #{s}, charge #{@charge}"
  end

  # Преобразование объекта в строковое описание
  def to_s
    @hash_key
  end

  # Добавление описания операции обмена из строки str
  # в список операций с наценкой
  def self.add_op_desc str
    @@exchanges_with_charge.push str
  end

  # метод расчета результата обмена количества s валюты curIn на валюту
curOut
  # по данной схеме. Результат: пара типа полученной валюты и ее количества
  # и начислением доходов и налогов
  def calc_change_res s
    # Наценка отправляется в доход пункта обмена во входной валюте
    ExchangeOffice.add_profit( @curIn, s * charge )
    # Возврат пары: типа полученной валюты и ее количества
    # за вычетом наценки
    super( s * (1-charge) )
  end

  # метод обмена количества s валюты curIn на валюту curOut по данной схеме
обмена
  # с занесением необходимых данных в статистику
  def change s
    # Описание операции
    op_desc = ex_op_desc s
    # Добавление описания операции в список операций обмена
    DirectExchange.add_op_desc op_desc
    # Добавление описания операции в список операций обмена с наценкой
    ExchangeWithCharge.add_op_desc op_desc
    # Вычисление результата
    calc_change_res s
  end

  # вывод информации обо всех обменных операциях с наценкой
```

```
  def self.stat
    puts "All exchanges with charge:"
    # Вывод описаний всех операций с наценкой
    @@exchanges_with_charge.each{ |str|
      puts "  " + str
    }
    puts
  end
end
##############################################################################
###

##############################################################################
###
# Класс метода обмена с наценкой и начислением налога - потомок
ExchangeWithCharge
class ExchangeWithChargeAndTax < ExchangeWithCharge
  # Приказ создать геттер для добавленного параметра
  attr_reader :tax

  # очистка статистики
  def self.reset
    # список обменных операций с начислением наценки и налога
    @@exchanges_with_charge_and_tax = []
  end

  # при инициализации (переопределнии) класса очищаем
  # (или инициализируем) статистику
  self.reset

  # конструктор
  # получает curIn - строка названия принимаемой для обмена валюты
  #          curOut - строка названия выдаваемой в ходе обмена валюты
  #          exVal - курс обмена (количество curOut, получаенной за единицу
curIn)
  #          charge - значение из [0.0; 1.0] - наценка
  #          tax - значение из [0.0; 1.0] - ставка налога
  def initialize curIn, curOut, exVal, charge, tax
    # вызов конструктора ExchangeWithCharge
    super curIn, curOut, exVal, charge
    @tax = tax
    # ключ для хеша - строка из всех параметров
    # это же представление будем выдавать в качестве строкового представления
    # в методе to_s
    @hash_key = "\"#{@curIn}\"-> \"#{@curOut}\" (charge #{@charge}, tax
#{@tax}): #{@exVal}"
  end

  # формирование строки-записи об операции обмена количества валюты s с
наценкой и налогами
  def ex_op_desc s
    "\"#{@curIn}\" -> \"#{@curOut}\": amount #{s}, charge #{@charge}, tax
#{@tax}"
```

```
  end

  # Преобразование объекта в строковое описание
  def to_s
    @hash_key
  end

  # Добавление описания операции обмена из строки str
  # в список операций с наценкой и налогами
  def self.add_op_desc str
    @@exchanges_with_charge_and_tax.push str
  end

  # метод расчета результата обмена количества s валюты curIn на валюту
curOut
  # по данной схеме. Результат: пара типа полученной валюты и ее количества
  # и начислением доходов и налогов
  def calc_change_res s
    # Вызов обмена с учетом наценки
    res = super( s )
    # Налог отправляется в хранилище пункта обмена в валюте выдачи
    ExchangeOffice.add_tax( @curOut, res[1] * tax )
    # Возврат пары: типа полученной валюты и ее количества
    # за вычетом наценки и налога
    [ res[0], res[1] * (1-tax) ]
  end

  # метод обмена количества s валюты curIn на валюту curOut по данной схеме
обмена
  # с занесением необходимых данных в статистику
  def change s
    # Описание операции
    op_desc = ex_op_desc s
    # Добавление описания операции в список операций обмена
    DirectExchange.add_op_desc op_desc
    # Добавление описания операции в список операций обмена с наценкой
    ExchangeWithCharge.add_op_desc op_desc
    # Добавление описания операции в список операций обмена с наценкой и
налогами
    ExchangeWithChargeAndTax.add_op_desc op_desc
    # Вычисление результата
    calc_change_res s
  end

  # вывод информации обо всех обменных операциях с наценкой и налогами
  def self.stat
    puts "All exchanges with charge and tax:"
    # Вывод описаний всех операций с наценкой и налогами
    @@exchanges_with_charge_and_tax.each{ |str|
      puts "  " + str
    }
    puts
  end
```

```
end
################################################################################
###

################################################################################
###
# Класс пользователя пункта обмена
class Client
  # конструктор класса получает его имя
  def initialize name
    @name = name
    # хэш валют в кармане пользователя
    @cur_hash = {}
    # список успешных операций обмена для данного пользователя
    @exchanges = []
  end

  # сколько валюты указанного типа у пользователя?
  def get_cur_val cur
    # Если такой валюты у пользователя нет
    if ( @cur_hash[ cur ].nil? )
      0
    # А если она есть
    else
      # Возврат ее количества
      @cur_hash[ cur ]
    end
  end

  # добавить указанное количество val к количеству
  # валюты cur у пользователя
  def add_val cur, val
    # Если такой валюты у пользователя нет
    if ( @cur_hash[ cur ].nil? )
      # Заведем ее
      @cur_hash[ cur ] = val
    # А если она есть
    else
      # Добавим количество
      @cur_hash[ cur ] += val
    end
  end

  # клиент получает доход в валюте cur в объеме val
  def income( currency, s )
    # Увеличить запас валюты указанного типа
    add_val currency, s
    # Результат - сам пользователь
    self
  end

  # строкое представление имени пользователя
  def to_s
```

```
    @name
  end

  # вывод информации о сумме средств клиента по каждой из валют
  def status
    puts "Contents of '#{to_s}' pocket:"
    # Обход хэша и вывод типов валюты и ее количества
    @cur_hash.each_pair { |cur, val|
      puts "    \"" + cur + "\": " + val.to_s
    }
    puts
  end

  # информация о всех успешных обменных операциях пользователя
  def stat
    puts "All successful exchange operations of '#{to_s}':"
    # Обход списка и вывод записей об операциях
    @exchanges.each { |op|
      puts "    #{op}"
    }
    puts
  end

  # обмен суммы s по схеме обмена scheme
  def change( s, scheme )
    # Если у пользвателя есть достаточное количество валюты нужного типа
    if get_cur_val( scheme.curIn ) >= s
      # забираем у пользователя обмениваемую сумму
      add_val( scheme.curIn, -s )
      # вычисляем результат обмена
      res = scheme.change( s )
      # добавляем пользователю результы обмена
      add_val( res[ 0 ], res[ 1 ] )
      # добавка сообщения об успешной операции в статистику
      # строку описания генерирует схема обмена
      @exchanges.push( scheme.ex_op_desc s )
      # Сообщение об успехе
      puts "Succeded change for '#{to_s}': #{s.to_s} from '#{scheme.curIn}'
to '#{scheme.curOut}'"
    # У пользователя нет столько валюты нужного типа
    else
      # Сообщение о неудаче
      puts "Failed change for '#{to_s}': #{s.to_s} from '#{scheme.curIn}' to
'#{scheme.curOut}'"
    end
    # Результат - сам пользователь
    self
  end

end
##############################################################################
###
```

```
# Прямой обмен рублей на юани со ставкой 20 рублей за юань
@exYuangEur = DirectExchange.new( "YUANG", "EUR", 1.0/20.0 )
puts "exYuangEur: " + @exYuangEur.to_s
puts "exYuangEur.curIn=\"" + @exYuangEur.curIn.to_s + "\""
puts "exYuangEur.curOut=\"" + @exYuangEur.curOut.to_s + "\""
puts "exYuangEur.exVal=" + @exYuangEur.exVal.to_s
puts

# Обмен евро на рубли со ставкой 95 рублей за евро, наценкой 8%
@exEurRub = ExchangeWithCharge.new( "EUR", "RUB", 95.0, 0.08 )
puts "exEurRub: " + @exEurRub.to_s
puts "exEurRub.curIn=\"" + @exEurRub.curIn.to_s + "\""
puts "exEurRub.curOut=\"" + @exEurRub.curOut.to_s + "\""
puts "exEurRub.exVal=" + @exEurRub.exVal.to_s
puts "exEurRub.charge=" + @exEurRub.charge.to_s
puts

# Обмен рублей на юани с наценкой 5%, налогом 4% и ставкой 8 рублей за юань
@exRubYuang = ExchangeWithChargeAndTax.new( "RUB", "YUANG", 1.0/8, 0.05, 0.04
)
puts "exRubYuang: " + @exRubYuang.to_s
puts "exRubYuang.curIn=\"" + @exRubYuang.curIn.to_s + "\""
puts "exRubYuang.curOut=\"" + @exRubYuang.curOut.to_s + "\""
puts "exRubYuang.exVal=" + @exRubYuang.exVal.to_s
puts "exRubYuang.charge=" + @exRubYuang.charge.to_s
puts "exRubYuang.tax=" + @exRubYuang.tax.to_s
puts

#Вывод статистики пункта обмена
ExchangeOffice.stat

# Перевод 1000 рублей в юани
# 50 рублей дохода, 4.75 юаней налогов
# результат - 114 юаней
puts "exRubYoung.change( 1000 )"
puts @exRubYuang.change( 1000 ).to_s

# Перевод 100 юаней в евро.
# без доходов, налогов
# результат - 5 евро
puts "exRubYoung.change( 100 )"
puts @exYuangEur.change( 100 ).to_s

# Перевод 700 евро в рубли
# 56 евро доходов, без налогов
# результат - 61180 рублей
puts "exEurRub.change( 700 )"
puts @exEurRub.change( 700 ).to_s

# Перевод 10000 рублей в юани
# 500 рублей дохода, 47.5 юаней налогов
# результат - 1140 юаней
```

```
puts "exRubYoung.change( 10000 )"
puts @exRubYuang.change( 10000 ).to_s

#Вывод статистики пункта обмена
# Доходы: рублей 50 + 500 = 550
#         евро   56
# Налоги: юани   4.75 + 47.5 = 52.25
ExchangeOffice.stat
#Вывод статистики операций обмена
DirectExchange.stat
ExchangeWithCharge.stat
ExchangeWithChargeAndTax.stat

# Доходы через income любого из трех классов *Exchange*
DirectExchange.income
# Налоги через tax метод любого из трех классов *Exchange*
ExchangeWithChargeAndTax.tax

# Создаем двух клиентов и наделяем их деньгами
@clAnna = Client.new( "Anna" )
@clAnna.income( "RUB", 1015.72 )
@clAnna.income( "YUANG", 100.00 )
@clVasya = Client.new( "Vasya" )
@clVasya.income( "EUR", 1.02 )

# Содержимое кошельков
@clAnna.status
@clVasya.status

# Анна пытается перевести 1100 рублей в юани
# Ничего не получается
# Теперь перевод 1000 рублей, которые у нее есть
@clAnna.change( 1100, @exRubYuang ).change( 1000, @exRubYuang )
# Вася переводит свои евро в рубли
@clVasya.change( 1.02, @exEurRub )

# Содержимое кошельков
@clAnna.status
@clVasya.status

# Выполненные переводы
@clAnna.stat
@clVasya.stat

#Вывод статистики пункта обмена
ExchangeOffice.stat
#Вывод статистики операций обмена
DirectExchange.stat
ExchangeWithCharge.stat
ExchangeWithChargeAndTax.stat
```