**Introducción a Ruby**

Código

|  |
| --- |
| require 'rails'  require 'csv'  class Producto  def initialize  @codigo = ""  @prod = ""  @color = ""  @tipo = ""  @envase = ""  @cantenv = 0  @precio = 0  @stock = 0  end  def gCod  @codigo  end  def sCod(valor)  @codigo = valor  end  def gProd  @prod  end  def sProd(valor)  @prod = valor  end  def gColor  @color  end  def sColor(valor)  @color = valor  end  def gTipo  @tipo  end  def sTipo(valor)  @tipo = valor  end  def gEnvase  @envase  end  def sEnvase(valor)  @envase = valor  end  def gCantenv  @cantenv  end  def sCantenv(valor)  @cantenv=valor  end  def gPrecio  @precio  end  def sPrecio(valor)  @precio = valor  end  def gStock  @stock  end  def sStock(valor)  @stock = valor  end  end  class Listaprod  def initialize  @list = Array.new  end  def cargarCsv  #arch = File.expand\_path('facturacion/productos.csv')  arch = 'facturacion\productos.csv'  CSV.foreach(arch, :headers => :true) do |row|  pr = Producto.new()  pr.sCod(row[0])  pr.sProd(row[1].truncate(30,:separator => '..+'))  pr.sColor(row[2])  pr.sTipo(row[3])  pr.sEnvase(row[4])  pr.sCantenv(row[5])  pr.sPrecio(row[6])  pr.sStock(row[7])  #puts "#{pr.gCod} #{pr.gProd} #{pr.gPrecio}"  @list.push(pr)  #puts "#{row[0]} #{row[1]} #{row[2]} #{row[3]} #{row[4]} #{row[5]} #{row[6]} #{row[7]} "  end #Cargo el arreglo desde un CSV  end  def listado  @list.each do |p|  puts "#{p.gCod.ljust(15)} #{p.gProd.ljust(50)} $ #{format\_number(p.gPrecio,0).rjust(8)} #{p.gStock.to\_s.rjust(4)}"  end  end  def buscarProdxid(id)  @list.each do |p|  if p.gCod == id  return p  break  end  end  end  end  class Giro  def initialize (id,nomgiro)  @id = id  @nombre = nomgiro  end  def getId  @id  end  def getNombre  @nombre  end  def setId(valor)  @id = valor  end  def setNombre(valor)  @nombre = valor  end  end  class Direccion  def initialize (id,calle,num,dep,villa,comuna,region)  @id = id  @calle = calle  @num = num  @dep = dep  @villa = villa  @comuna = comuna  @region = region  end  def getId  @id  end  def getCalle  @nombre  end  def getNum  @num  end  def getDep  @dep  end  def getVilla  @villa  end  def getComuna  @comuna  end  def getRegion  @region  end  def getDir  return "#{@calle} #{@num} #{@dep} #{@villa} - #{@comuna}"  end  def setId(valor)  @id = valor  end  def setCalle(valor)  @calle = valor  end  end  class Cliente  def initialize (rut,razsoc,dir,giro)  @rut = rut  @razsoc = razsoc  @dir = dir  @giro = giro  end  def getRut  @rut  end  def getNombre  @razsoc  end  def getDir  @dir  end  def getGiro  @giro  end  end  class Listacli  def initialize  @list = Array.new  end  def cargarCsv  require 'csv'  arch = 'D:\PFILES\productos.csv'  CSV.foreach(arch, :headers => :true) do |row|  g = Giro.new(1,row[2])  dir = Direccion.new(0,row[3],row[4],row[5],row[6],row[7],"N/A")  cl = Cliente.new(row[0],row[1],dir,g)  #puts "#{pr.gCod} #{pr.gProd} #{pr.gPrecio}"  @list.push(cl)  #puts "#{row[0]} #{row[1]} #{row[2]} #{row[3]} #{row[4]} #{row[5]} #{row[6]} #{row[7]} "  end  end  def listado  @list.each do |p|  puts "#{p.getRut.ljust(13)} #{p.getNombre.ljust(50)} "  end  end  end  class Estados  def initialize (id,nombre)  @id = id  @nombre = nombre  end  def getId  @id  end  def getNombre  @nombre  end  def setId(valor)  @id = valor  end  def setNombre(valor)  @nombre = valor  end  end  class Detfactura  def initialize  @cant = 0  @codpro = ""  @descr = ""  @valor = 0  end  def clonar(dtf)  @cant = dtf.getCant  @codpro = dtf.getCod  @descr = dtf.getDescr  @valor = dtf.getValor  end  def setProd(cant,prod)  @cant = cant  @codpro = prod.gCod  @descr = prod.gProd  @valor = prod.gPrecio.to\_i  end  def setLibre(cant,descr,vunit)  @cant = cant  @codpro = ""  @descr = descr  @valor = vunit  end  def getCant  @cant  end  def getDescr  @descr  end  def getValor  @valor  end  def getCod  @codpro  end  def getSubt()  return (@cant \* @valor)  end  end  class Factura  def initialize (folio,fecha,emisor,cliente,estado)  @folio = folio  @fecha = fecha  @emisor = emisor  @cliente = cliente  @estado = estado  @neto = 0  @iva = 0  @total = 0  @saldo = 0  @detalle = Array.new()  end  def agregaDetalle(det)  df = Detfactura.new  df.clonar det  @detalle.push (df)  #puts " cant #{df.getCant} descr #{df.getDescr} valor #{df.getValor}"  calculaTotales  end  def calculaTotales  totneto = 0  @detalle.each do |det|  totneto += det.getCant \* det.getValor  end  @neto = totneto  @iva = totneto \* 0.19  @total = totneto + @iva  end  def getNeto  @neto  end  def getIva  @iva  end  def getTotal  @total  end  def imprimeFactura  puts  puts "────────────────────────────────"  puts @emisor.getRut  puts @emisor.getNombre  puts "────────────────────────────────"  puts "FACTURA DE VENTA N° #{@folio.to\_s.rjust(7)}"  puts "────────────────────────────────"  puts "RUT Cliente : #{@cliente.getRut}"  puts "Razón Social: #{@cliente.getNombre}"  puts "Dirección : #{@cliente.getDir.getDir} "  puts "Giro : #{@cliente.getGiro.getNombre}"  puts "─────────────────────────────────────────────────────────"  puts "DETALLE DE FACTURA"  puts "─────────────────────────────────────────────────────────"  @detalle.each do |det|  puts "#{det.getCant.to\_s.rjust(5)} #{det.getDescr.ljust(30)} $#{format\_number(det.getValor,0).rjust(8)} $#{format\_number(det.getSubt,0).rjust(8)}"  end  puts "─────────────────────────────────────────────────────────"  print "Total Neto : $ ".rjust(50)  puts format\_number(@neto.to\_s,0).rjust(7)  print "IVA : $ ".rjust(50)  puts format\_number(@iva.to\_s,0).rjust(7)  print "Total : $ ".rjust(50)  puts format\_number(@total.to\_s,0).rjust(7)  puts "─────────────────────────────────────────────────────────"  end  end  def format\_number(number,dec)  precision = 0  if dec > 0  precision = dec  end  # puts "NUMERO INICIAL #{number}"  if dec > 0 then  number = number.round(precision)  # puts "truncado de numero en #{number}"  end  todo, decimal = number.to\_s.split(".")  strdec = decimal.to\_s  # puts "ENTERO = #{todo} DECIMAL = #{strdec} PRECISION = #{dec}"  for i in(strdec.length..dec-1)  strdec += "0"  end  num\_groups = todo.chars.to\_a.reverse.each\_slice(3)  todo\_y\_coma = num\_groups.map(&:join).join(',').reverse  if precision > 0 then  return [todo\_y\_coma, strdec].compact.join(".")  else  return todo\_y\_coma  end  end  #INICIO DE EJECUCION  #-------------------------------  #Se establecen los datos del emisor de Facturas  system("cls")  lp = Listaprod.new()  lp.cargarCsv  lc = Listacli.new  lc.cargarCsv  estdefault = Estados.new(0,"Pendiente")  estcancelado = Estados.new(1,"Cancelado")  gemi = Giro.new(66,"Servicios Generales")  demi = Direccion.new(1,"Rincón del Diablo","666","","Estrella de la Mañana","Santiago","Metropolitana")  emi = Cliente.new("12884231-4","Javier Salgado",demi,gemi)  #Se establece el dato del cliente  gvta = Giro.new(44,"Servicios Generales")  dvta = Direccion.new(1,"Alameda","666","","Barrio Cívico","Santiago","Metropolitana")  cl = Cliente.new("12884231-4","Javier Salgado",dvta,gvta)  fac = Factura.new(1,"30-08-2021",emi,cl,estdefault)  deta = Detfactura.new()  deta.setLibre(10,"Super 8", 100)  fac.agregaDetalle(deta)  #deta = Detfactura.new()  deta.setLibre(5,"Berlin pequeño",500)  fac.agregaDetalle(deta)  #deta = Detfactura.new()  deta.setLibre(7,"Papas Fritas Lays 170 Gr",1100)  fac.agregaDetalle(deta)  #deta = Detfactura.new()  deta.setLibre(12,"Coca Cola 600cc",800)  fac.agregaDetalle(deta)  #lp.listado  #lc.listado  pr = Producto.new  pr = lp.buscarProdxid('4807204917242')  #puts pr.gProd  deta.setProd(1,pr)  fac.agregaDetalle(deta)  fac.imprimeFactura  #df = Detfactura.new() |

**Descripción**

Esta vez seguimos profundizando en el mismo ejercicio de facturación y productos.

**Reflexión**

Se me está haciendo muy difícil escribir la evidencia, precisamente porque las cosas básicas del lenguaje ya están aprendidas, y el resto es práctica. No se me ocurre cómo expresar en 80 palabras un pequeño descubrimiento que impacta muy fuertemente mi forma de programar.