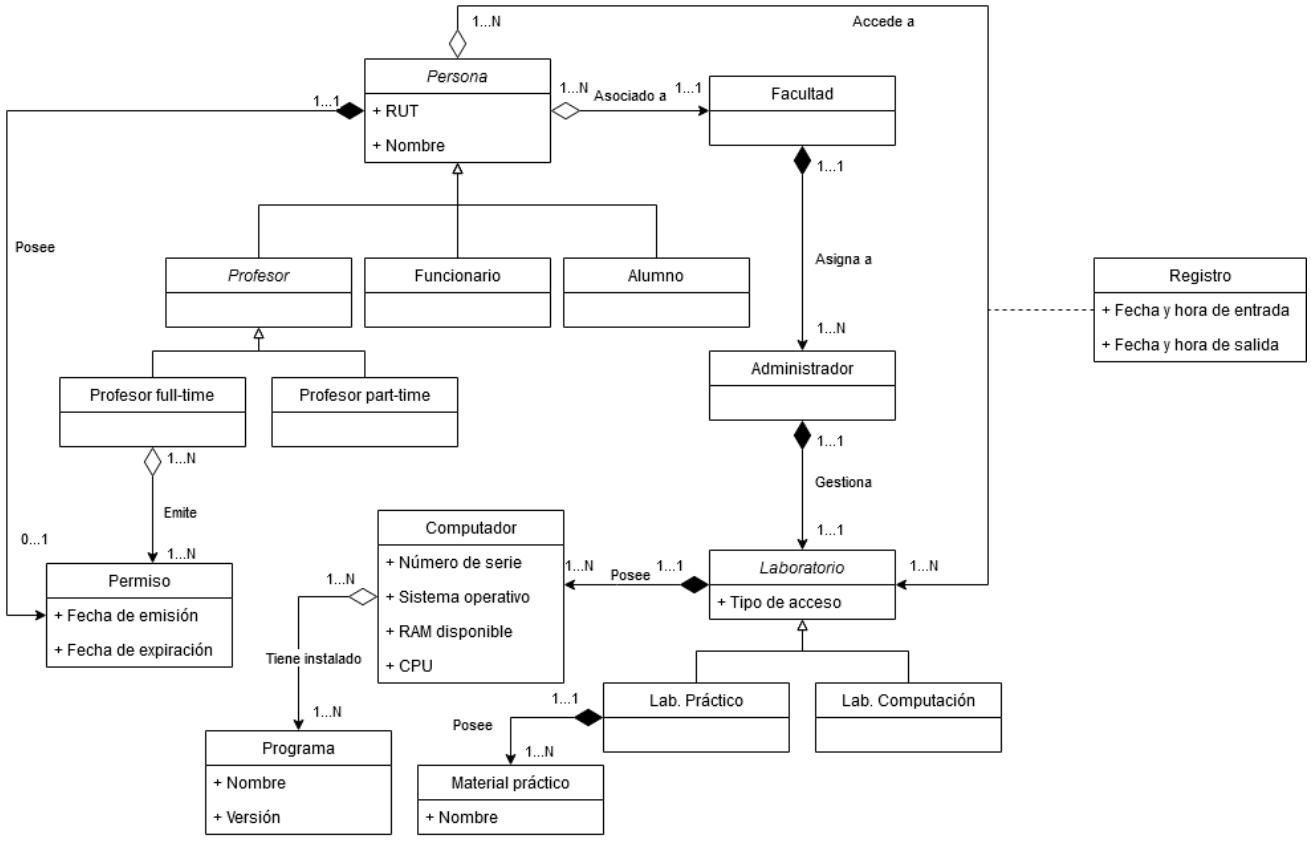
Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2143— Ingeniería de Software 2020-1

17 de Junio de 2020
Matías Duhalde

Interrogación 2

Pregunta 1



Supuestos realizados

- Un **administrador** corresponde a un funcionario de la universidad designado por la facultad para gestionar el laboratorio.
- La política de uso de los laboratorios (libre o restringido) se indica por el atributo **tipo de acceso** dentro de **Laboratorio**. Lo hice así para no complicar excesivamente el árbol de generalización de los **Laboratorios**.
- Definí las características del procesador como **CPU** (atributo de **Computador**), para no tener que crear una entidad/tabla extra que a su vez defina las características (núcleos y frecuencia) de cada procesador.

Pregunta 2

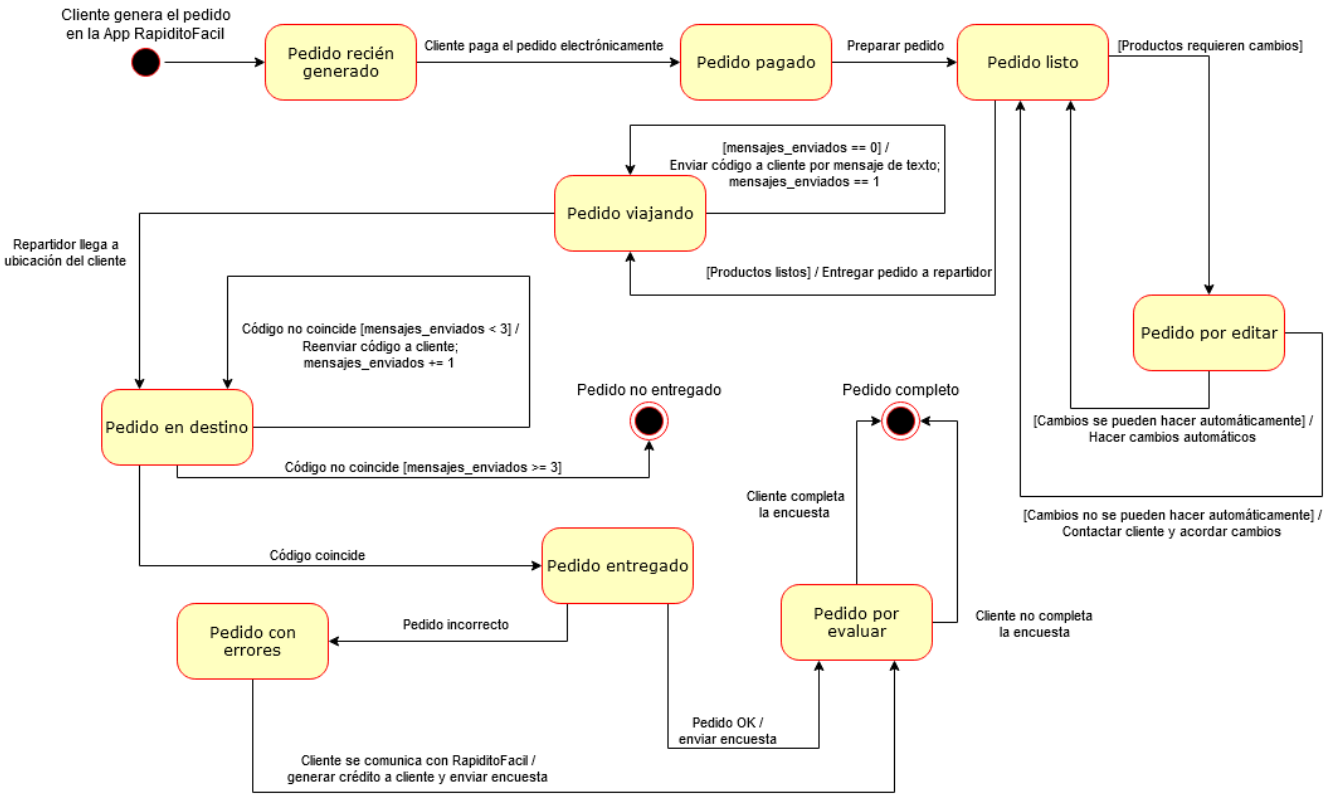
```ruby
class EncriptionObject
  def encrypt(string)
    string
  end

  def decrypt(string)
    string
  end
end

class RotDecorator
  def initialize(decorator, n)
    @decorator = decorator
    @n = n
  end

  def encrypt(string)
    string = @decorator.encrypt(string)
    list = []
    string.split("").each do |char|
      if 96 < char.ord && char.ord < 97 + 26
        list << ((char.ord - 97 + @n) % 26 + 97).chr
      else
        list << char
      end
    end
    string = list.join("")
  end

  def decrypt(string)
    list = []
    string.split("").each do |char|
      if 96 < char.ord && char.ord < 97 + 26
        list << ((char.ord - 97 - @n) % 26 + 97).chr
      else
        list << char
      end
    end
    string = list.join("")
    string = @decorator.decrypt(string)
  end
end

class SimpleTransposeDecorator
  def initialize(decorator, n)
    @decorator = decorator
    @n = n
  end
```

```ruby

  def encrypt(string)
    string = @decorator.encrypt(string)
    offset = -(@n) % string.length
    string = string[offset, string.length] + string[0, offset]
  end

  def decrypt(string)
    offset = (@n) % string.length
    string = string[offset, string.length] + string[0, offset]
    string = @decorator.decrypt(string)
  end
end

def get_random_cipher
  object = EncriptionObject.new
  recursive_part(object)
end

def recursive_part(object)
  n = (rand() * 10 ).ceil()
  if rand() < 0.5
    object = SimpleTransposeDecorator.new(object, n)
  else
    object = RotDecorator.new(object, n)
  end
  if rand() < 0.8
    recursive_part(object)
  else
    object
  end
end

# Test
cipher = get_random_cipher
puts encrypted_string = cipher.encrypt("hola mundo")
puts cipher.decrypt(encrypted_string)
```

Supuestos realizados

- Para elegir entre `Rot` y `SimpleTranspose`, se usó una probabilidad 50-50.
- Cuando se elige el valor de `n` entre 1 y 10, esto último lo consideré **inclusive**.
- Para `text_random_cipher`, entendí del enunciado (del paso 3, específicamente) que se podían repetir los pasos infinitamente (respetando la probabilidad del 80 %). En otras palabras, entendí que el cifrado se podía componer indefinidamente. Por esta razón, la definí recursivamente, y la parte recursiva está en `recursive_part`.
- El código será incluido dentro del *zip* de entrega en el archivo `p2.rb`, para que lo puedan corregir más fácil.

Pregunta 3

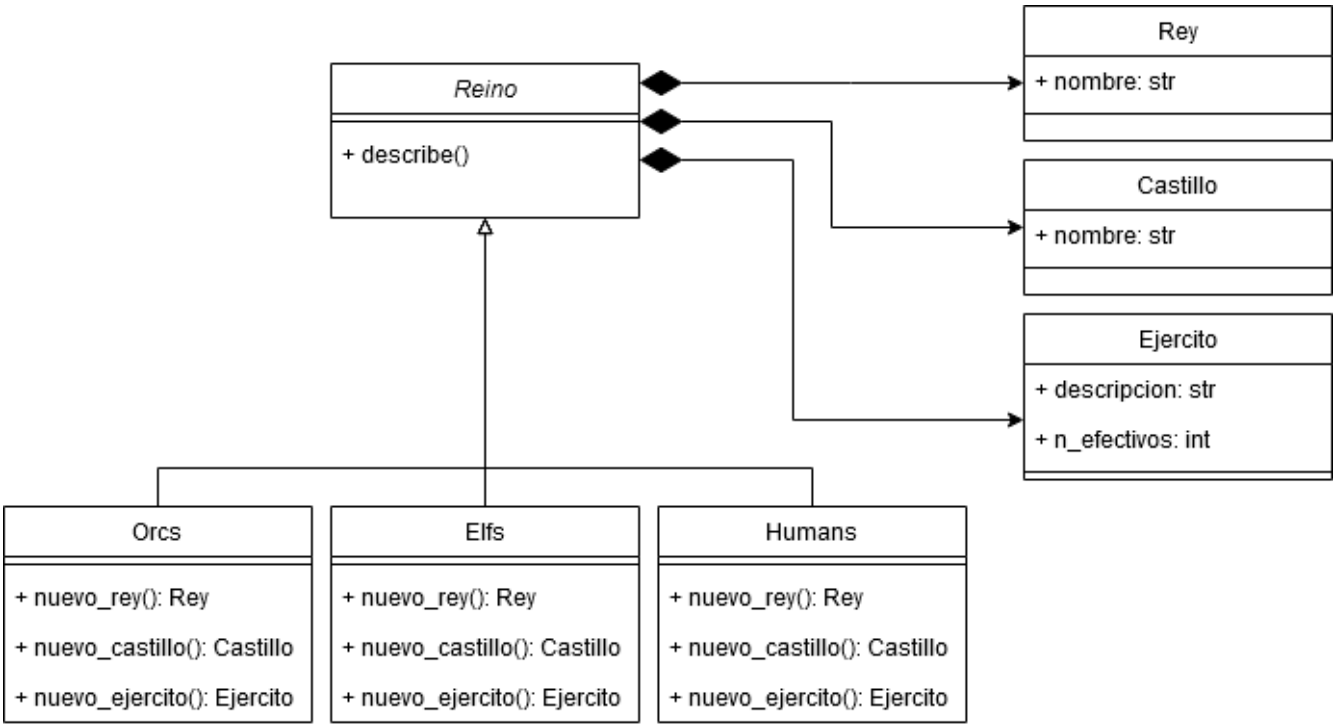


Estados

- **Pedido recién generado**: Primer estado, ocurre luego que el cliente genere el pedido mediante la App RapiditoFacil.
- **Pedido pagado**: Ocurre desde **Pedido recién generado**, luego de que el cliente complete exitosamente el pago electrónico. A espera de ser enviado al local.
- **Pedido listo**: Pedido ya preparado por el local. Deben revisarse sus productos y si es necesario hacer cambios.
- **Pedido por editar**: Pedido tiene productos que requieren cambios.
- **Pedido viajando**: Pedido en poder del repartidor, viajando en camino al cliente final. Notar variable `mensajes_enviados`, representa las veces que se ha enviado un mensaje de texto con el código del pedido al cliente.
- **Pedido en destino**: Pedido aún en poder del repartidor, el cual se reúne con el cliente para entregarlo.
- **Pedido entregado**: Pedido en poder del cliente.
- **Pedido con errores**: Pedido en poder del cliente, posee errores (productos faltantes/incorrectos).
- **Pedido por evaluar**: Pedido ya entregado, esperando a que el cliente conteste la encuesta.
- **Pedido no entregado**: Estado final. El pedido no fue entregado al cliente final.
- **Pedido completo**: Estado final. El pedido ya se entregó y ya se envió la encuesta al cliente.

Pregunta 4

Parte a



Parte b y c

```ruby
class Rey
  attr_accessor :nombre

  def initialize(nombre)
    @nombre = nombre
  end
end

class Castillo
  attr_accessor :nombre

  def initialize(nombre)
    @nombre = nombre
  end
end

class Ejercito
  attr_accessor :descripcion, :n_efectivos

  def initialize(descripcion, n_efectivos)
    @descripcion = descripcion
    @n_efectivos = n_efectivos
  end
```

```ruby
end

class Realm
  def initialize
    @rey = nuevo_rey
    @castillo = nuevo_castillo
    @ejercito = nuevo_ejercito
  end

  def describe
    puts "I am #{@rey.nombre}, king of the #{self.class.name}"
    puts "This is #{@castillo.nombre}, home of the #{self.class.name}"
    puts "This is the powerful #{@ejercito.descripcion} army " +
         "composed of #{@ejercito.n_efectivos} #{self.class.name}"
  end
end

class Orcs < Realm
  def nuevo_rey
    Rey.new("Melkor")
  end

  def nuevo_castillo
    Castillo.new("Isengard")
  end

  def nuevo_ejercito
    Ejercito.new("Orcs", 20000)
  end
end

class Elfs < Realm # Elves?
  def nuevo_rey
    Rey.new("Thranduil")
  end

  def nuevo_castillo
    Castillo.new("Woodlands")
  end

  def nuevo_ejercito
    Ejercito.new("Elves", 2500)
  end
end

class Humans < Realm
  def nuevo_rey
    Rey.new("Aragorn")
  end

```

```ruby
  def nuevo_castillo
    Castillo.new("Silmarillion")
  end

  def nuevo_ejercito
    Ejercito.new("Soldiers", 4500)
  end
end


orcs = Orcs.new
elfs = Elfs.new
humans = Humans.new

orcs.describe
puts "-"*60
elfs.describe
puts "-"*60
humans.describe
```

Supuestos realizados

- Implementé las clases `Orcs`, `Elfs`, y `Humans` como *abstract factories* de una clase abstracta `Reino`.
- `Rey` y `Castillo` pudieron haber sido definidos como atributos de `Reino` (ya que solo poseen nombre), pero para que puedan ser escalados de mejor manera a un futuro, se definieron como clases aparte.
- Los valores de los atributos quedaron *hardcodeados* dentro de las clases, `Orcs` `Elfs`, y `Humans`, con el objetivo de simplificar los atributos de los constructores.