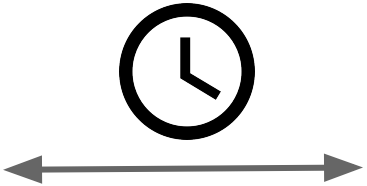
IIC2113 - Diseño Detallado de Software

Presentación Final Grupo 4

1. Introducción

Contexto

Exchange



Banco

Contexto

Código funcional

Refactor

1. Nuestro código inicial

No estamos orgullosos...

```ruby
class InvestmentJob < ApplicationJob
  include HTTParty
  format :json
  base_uri "https://guarded-wildwood-14760.herokuapp.com/api/v1/"
  queue_as :default

  def perform

    api_key = "sQ9ceqzXkyH_2eK6M1Z_"
    all_accounts = Account.all
    total_investment_money = 0

    all_accounts.each do |account|
      #Le quitamos el 5% a cada usuario
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end

    path_get_prices = "/prices?token=#{api_key}"
    sale_price = 0
    purchase_price = 0
    self.class.get(path_get_prices).each do |price|
      sale_price = price["saleprice"]
      purchase_price = price["purchaseprice"]
    end

    path_get_balance = "/balances?token=#{api_key}"
    amount_to_buy = (total_investment_money / purchase_price.to_f).to_f
    path_purchase_btf = "/buy?token=#{api_key}&amount=#{amount_to_buy}"
    message_to_show = self.class.post(path_purchase_btf)["msg"]
    puts message_to_show #compra BTF

    clp_before = self.class.get(path_get_balance)["balances"]["CLP"].to_f
    btf_after_purchase = self.class.get(path_get_balance)["balances"]["BTF"].to_f

    path_sell_btf = "/sell?token=#{api_key}&amount=#{btf_after_purchase}"
    message_to_show_2 = self.class.post(path_sell_btf)["msg"]
    puts message_to_show_2 #venta BTF

    clp_after = self.class.get(path_get_balance)["balances"]["CLP"].to_f
    earning_clp = clp_after -clp_before
    amount_of_total_users = all_accounts.length()
    clp_each_user = (earning_clp / amount_of_total_users)
    all_accounts.each do |account|
      account.saving_balance += clp_each_user
      account.save
    end
  end
end
```

```ruby
class InvestmentJob < ApplicationJob
  include HTTParty
  format :json
  base_uri "https://guarded-wildwood-14760.herokuapp.com/api/v1/"
  queue_as :default

  def perform

    api_key = "sQ9ceqzXkyH_2eK6M1Z_"
    all_accounts = Account.all
    total_investment_money = 0

    all_accounts.each do |account|
      #Le quitamos el 5% a cada usuario
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end

    path_get_prices = "/prices?token=#{api_key}"
    sale_price = 0
    purchase_price = 0
    self.class.get(path_get_prices).each do |price|
      sale_price = price["saleprice"]
      purchase_price = price["purchaseprice"]
    end

    path_get_balance = "/balances?token=#{api_key}"
    amount_to_buy = (total_investment_money / purchase_price.to_f).to_f
    path_purchase_btf = "/buy?token=#{api_key}&amount=#{amount_to_buy}"
    message_to_show = self.class.post(path_purchase_btf)["msg"]
    puts message_to_show #compra BTF

    clp_before = self.class.get(path_get_balance)["balances"]["CLP"].to_f
    btf_after_purchase = self.class.get(path_get_balance)["balances"]["BTF"].to_f

    path_sell_btf = "/sell?token=#{api_key}&amount=#{btf_after_purchase}"
    message_to_show_2 = self.class.post(path_sell_btf)["msg"]
    puts message_to_show_2 #venta BTF

    clp_after = self.class.get(path_get_balance)["balances"]["CLP"].to_f
    earning_clp = clp_after -clp_before
    amount_of_total_users = all_accounts.length()
    clp_each_user = (earning_clp / amount_of_total_users)
    all_accounts.each do |account|
      account.saving_balance += clp_each_user
      account.save
    end
  end
end
```

```ruby
class InvestmentJob < ApplicationJob
  include HTTParty
  format :json
  base_uri "https://guarded-wildwood-14760.herokuapp.com/api/v1/"
  queue_as :default

  def perform

    api_key = "sQ9ceqzXkyH_2eK6M1Z_"
    all_accounts = Account.all
    total_investment_money = 0

    all_accounts.each do |account|
      #Le quitamos el 5% a cada usuario
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end

    path_get_prices = "/prices?token=#{api_key}"
    sale_price = 0
    purchase_price = 0
    self.class.get(path_get_prices).each do |price|
      sale_price = price["saleprice"]
      purchase_price = price["purchaseprice"]
    end

    path_get_balance = "/balances?token=#{api_key}"
    amount_to_buy = (total_investment_money / purchase_price.to_f).to_f
    path_purchase_btf = "/buy?token=#{api_key}&amount=#{amount_to_buy}"
    message_to_show = self.class.post(path_purchase_btf)["msg"]
    puts message_to_show #compra BTF

    clp_before = self.class.get(path_get_balance)["balances"]["CLP"].to_f
    btf_after_purchase = self.class.get(path_get_balance)["balances"]["BTF"].to_f

    path_sell_btf = "/sell?token=#{api_key}&amount=#{btf_after_purchase}"
    message_to_show_2 = self.class.post(path_sell_btf)["msg"]
    puts message_to_show_2 #venta BTF

    clp_after = self.class.get(path_get_balance)["balances"]["CLP"].to_f
    earning_clp = clp_after -clp_before
    amount_of_total_users = all_accounts.length()
    clp_each_user = (earning_clp / amount_of_total_users)
    all_accounts.each do |account|
      account.saving_balance += clp_each_user
      account.save
    end
  end
end
```

```ruby
class InvestmentJob < ApplicationJob
  ...

  def perform

    api_key = "sQ9ceqzXkyH_2eK6M1Z_"
    all_accounts = Account.all
    total_investment_money = 0

    all_accounts.each do |account|
      #Le quitamos el 5% a cada usuario
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end

    path_get_prices = "/prices?token=#{api_key}"
    sale_price = 0
    purchase_price = 0
    self.class.get(path_get_prices).each do |price|
      sale_price = price["saleprice"]
      purchase_price = price["purchaseprice"]
    end

    ...
  end
end
```

```ruby
class InvestmentJob < ApplicationJob
  ...

  def perform

    api_key = "sQ9ceqzXkyH_2eK6M1Z_"
    all_accounts = Account.all
    total_investment_money = 0

    all_accounts.each do |account|
      #Le quitamos el 5% a cada usuario
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end

    path_get_prices = "/prices?token=#{api_key}"
    sale_price = 0
    purchase_price = 0
    self.class.get(path_get_prices).each do |price|
      sale_price = price["saleprice"]
      purchase_price = price["purchaseprice"]
    end

    ...
  end
end
```

```ruby
class InvestmentJob < ApplicationJob
  ...

  def perform

    api_key = "sQ9ceqzXkyH_2eK6M1Z_"
    all_accounts = Account.all
    total_investment_money = 0

    all_accounts.each do |account|
      #Le quitamos el 5% a cada usuario
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end

    path_get_prices = "/prices?token=#{api_key}"
    sale_price = 0
    purchase_price = 0
    self.class.get(path_get_prices).each do |price|
      sale_price = price["saleprice"]
      purchase_price = price["purchaseprice"]
    end

    ...
  end
end
```

```ruby
class InvestmentJob < ApplicationJob
  ...

  def perform

    api_key = "sQ9ceqzXkyH_2eK6M1Z_"
    all_accounts = Account.all
    total_investment_money = 0

    all_accounts.each do |account|
      #Le quitamos el 5% a cada usuario
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end

    path_get_prices = "/prices?token=#{api_key}"
    sale_price = 0
    purchase_price = 0
    self.class.get(path_get_prices).each do |price|
      sale_price = price["saleprice"]
      purchase_price = price["purchaseprice"]
    end

    ...
  end
end
```

```ruby
class InvestmentJob < ApplicationJob
  ...

  def perform

    ...
    path_get_balance = "/balances?token=#{api_key}"
    amount_to_buy = (total_investment_money / purchase_price.to_f).to_f
    path_purchase_btf = "/buy?token=#{api_key}&amount=#{amount_to_buy}"
    message_to_show = self.class.post(path_purchase_btf)["msg"]
    puts message_to_show #compra BTF

    clp_before = self.class.get(path_get_balance)["balances"]["CLP"].to_f
    btf_after_purchase = self.class.get(path_get_balance)["balances"]["BTF"].to_f

    path_sell_btf = "/sell?token=#{api_key}&amount=#{btf_after_purchase}"
    message_to_show_2 = self.class.post(path_sell_btf)["msg"]
    puts message_to_show_2 #venta BTF

    clp_after = self.class.get(path_get_balance)["balances"]["CLP"].to_f
    earning_clp = clp_after -clp_before
    amount_of_total_users = all_accounts.length()
    clp_each_user = (earning_clp / amount_of_total_users)
    all_accounts.each do |account|
      account.saving_balance += clp_each_user
      account.save
    end
  end
end
```

```ruby
class InvestmentJob < ApplicationJob
  ...

  def perform

    ...
    path_get_balance = "/balances?token=#{api_key}"
    amount_to_buy = (total_investment_money / purchase_price.to_f).to_f
    path_purchase_btf = "/buy?token=#{api_key}&amount=#{amount_to_buy}"
    message_to_show = self.class.post(path_purchase_btf)["msg"]
    puts message_to_show #compra BTF

    clp_before = self.class.get(path_get_balance)["balances"]["CLP"].to_f
    btf_after_purchase = self.class.get(path_get_balance)["balances"]["BTF"].to_f

    path_sell_btf = "/sell?token=#{api_key}&amount=#{btf_after_purchase}"
    message_to_show_2 = self.class.post(path_sell_btf)["msg"]
    puts message_to_show_2 #venta BTF

    clp_after = self.class.get(path_get_balance)["balances"]["CLP"].to_f
    earning_clp = clp_after -clp_before
    amount_of_total_users = all_accounts.length()
    clp_each_user = (earning_clp / amount_of_total_users)
    all_accounts.each do |account|
      account.saving_balance += clp_each_user
      account.save
    end
  end
end
```

```ruby
class InvestmentJob < ApplicationJob
  ...

  def perform

    ...
    path_get_balance = "/balances?token=#{api_key}"
    amount_to_buy = (total_investment_money / purchase_price.to_f).to_f
    path_purchase_btf = "/buy?token=#{api_key}&amount=#{amount_to_buy}"
    message_to_show = self.class.post(path_purchase_btf)["msg"]
    puts message_to_show #compra BTF

    clp_before = self.class.get(path_get_balance)["balances"]["CLP"].to_f
    btf_after_purchase = self.class.get(path_get_balance)["balances"]["BTF"].to_f

    path_sell_btf = "/sell?token=#{api_key}&amount=#{btf_after_purchase}"
    message_to_show_2 = self.class.post(path_sell_btf)["msg"]
    puts message_to_show_2 #venta BTF

    clp_after = self.class.get(path_get_balance)["balances"]["CLP"].to_f
    earning_clp = clp_after -clp_before
    amount_of_total_users = all_accounts.length()
    clp_each_user = (earning_clp / amount_of_total_users)
    all_accounts.each do |account|
      account.saving_balance += clp_each_user
      account.save
    end
  end
end
```

Code Smells

1. **Large class**
   - `InvestmentJob` tiene excesivas líneas de código
   - Tiene toda la responsabilidad → No cumple SOLID

Code Smells

1. **Large class**
   - `InvestmentJob` tiene excesivas líneas de código
   - Tiene toda la responsabilidad → No cumple SOLID
2. **Long Method**
   - Método `perform` tiene excesivas líneas de código

2. Nuestro código refactorizado

Este si es nuestro orgullo

Soluciones

1. **Large class**
    - Crear otra clase para manejo de API
2. **Long Method**
    - Descomponer en funciones
    - Mover el código correspondiente a cada clase

```ruby
class ApiJob
  include HTTParty
  format :json
  base_uri "https://guarded-wildwood-14760.herokuapp.com/api/v1/"

  def initialize
    @api_key = "sQ9ceqzXkyH_2eK6M1Z_"
  end

  def get_purchase_price
    path_get_prices = "/prices?token=#{@api_key}"
    purchase_price = self.class.get(path_get_prices)[0]["purchaseprice"]
    purchase_price
  rescue StandardError => e
    puts "Error: #{e}"
  end

  def get_balance(currency)
    path_get_balance = "/balances?token=#{@api_key}"
    balance = self.class.get(path_get_balance)["balances"][currency].to_f
    balance
  rescue StandardError => e
    puts "Error: #{e}"
  end

  def buy_btf(amount)
    path_purchase_btf = "/buy?token=#{@api_key}&amount=#{amount}"
    message_to_show = self.class.post(path_purchase_btf)["msg"]
    puts message_to_show    # compra BTF
  rescue StandardError => e
    puts "Error: #{e}"
  end

  def sell_btf(amount)
    path_sell_btf = "/sell?token=#{@api_key}&amount=#{amount}"
    message_to_show = self.class.post(path_sell_btf)["msg"]
    puts message_to_show   # venta BTF
  rescue StandardError => e
    puts "Error: #{e}"
  end
end
```

```ruby
class ApiJob
  include HTTParty
  format :json
  base_uri "https://guarded-wildwood-14760.herokuapp.com/api/v1/"

  def initialize
    @api_key = "sQ9ceqzXkyH_2eK6M1Z_"
  end

  def get_purchase_price
    path_get_prices = "/prices?token=#{@api_key}"
    purchase_price = self.class.get(path_get_prices)[0]["purchaseprice"]
    purchase_price
  rescue StandardError => e
    puts "Error: #{e}"
  end

  def get_balance(currency)
    path_get_balance = "/balances?token=#{@api_key}"
    balance = self.class.get(path_get_balance)["balances"][currency].to_f
    balance
  rescue StandardError => e
    puts "Error: #{e}"
  end

  def buy_btf(amount)
    path_purchase_btf = "/buy?token=#{@api_key}&amount=#{amount}"
    message_to_show = self.class.post(path_purchase_btf)["msg"]
    puts message_to_show    # compra BTF
  rescue StandardError => e
    puts "Error: #{e}"
  end

  def sell_btf(amount)
    path_sell_btf = "/sell?token=#{@api_key}&amount=#{amount}"
    message_to_show = self.class.post(path_sell_btf)["msg"]
    puts message_to_show   # venta BTF
  rescue StandardError => e
    puts "Error: #{e}"
  end
end
```

```ruby
class ApiJob
  include HTTParty
  format :json
  base_uri "https://guarded-wildwood-14760.herokuapp.com/api/v1/"

  def initialize
    @api_key = "sQ9ceqzXkyH_2eK6M1Z_"
  end

  def get_purchase_price
    path_get_prices = "/prices?token=#{@api_key}"
    purchase_price = self.class.get(path_get_prices)[0]["purchaseprice"]
    purchase_price
  rescue StandardError => e
    puts "Error: #{e}"
  end

  def get_balance(currency)
    path_get_balance = "/balances?token=#{@api_key}"
    balance = self.class.get(path_get_balance)["balances"][currency].to_f
    balance
  rescue StandardError => e
    puts "Error: #{e}"
  end

  def buy_btf(amount)
    path_purchase_btf = "/buy?token=#{@api_key}&amount=#{amount}"
    message_to_show = self.class.post(path_purchase_btf)["msg"]
    puts message_to_show    # compra BTF
  rescue StandardError => e
    puts "Error: #{e}"
  end

  def sell_btf(amount)
    path_sell_btf = "/sell?token=#{@api_key}&amount=#{amount}"
    message_to_show = self.class.post(path_sell_btf)["msg"]
    puts message_to_show   # venta BTF
  rescue StandardError => e
    puts "Error: #{e}"
  end
end
```

```ruby
class ApiJob
  include HTTParty
  format :json
  base_uri "https://guarded-wildwood-14760.herokuapp.com/api/v1/"

  def initialize
    @api_key = "sQ9ceqzXkyH_2eK6M1Z_"
  end

  def get_purchase_price
    path_get_prices = "/prices?token=#{@api_key}"
    purchase_price = self.class.get(path_get_prices)[0]["purchaseprice"]
    purchase_price
  rescue StandardError => e
    puts "Error: #{e}"
  end

  def get_balance(currency)
    path_get_balance = "/balances?token=#{@api_key}"
    balance = self.class.get(path_get_balance)["balances"][currency].to_f
    balance
  rescue StandardError => e
    puts "Error: #{e}"
  end

  def buy_btf(amount)
    path_purchase_btf = "/buy?token=#{@api_key}&amount=#{amount}"
    message_to_show = self.class.post(path_purchase_btf)["msg"]
    puts message_to_show    # compra BTF
  rescue StandardError => e
    puts "Error: #{e}"
  end

  def sell_btf(amount)
    path_sell_btf = "/sell?token=#{@api_key}&amount=#{amount}"
    message_to_show = self.class.post(path_sell_btf)["msg"]
    puts message_to_show   # venta BTF
  rescue StandardError => e
    puts "Error: #{e}"
  end
end
```

```ruby
class ApiJob
  include HTTParty
  format :json
  base_uri "https://guarded-wildwood-14760.herokuapp.com/api/v1/"

  def initialize
    @api_key = "sQ9ceqzXkyH_2eK6M1Z_"
  end

  def get_purchase_price
    path_get_prices = "/prices?token=#{@api_key}"
    purchase_price = self.class.get(path_get_prices)[0]["purchaseprice"]
    purchase_price
  rescue StandardError => e
    puts "Error: #{e}"
  end

  def get_balance(currency)
    path_get_balance = "/balances?token=#{@api_key}"
    balance = self.class.get(path_get_balance)["balances"][currency].to_f
    balance
  rescue StandardError => e
    puts "Error: #{e}"
  end

  def buy_btf(amount)
    path_purchase_btf = "/buy?token=#{@api_key}&amount=#{amount}"
    message_to_show = self.class.post(path_purchase_btf)["msg"]
    puts message_to_show    # compra BTF
  rescue StandardError => e
    puts "Error: #{e}"
  end

  def sell_btf(amount)
    path_sell_btf = "/sell?token=#{@api_key}&amount=#{amount}"
    message_to_show = self.class.post(path_sell_btf)["msg"]
    puts message_to_show   # venta BTF
  rescue StandardError => e
    puts "Error: #{e}"
  end
end
```

```ruby
class InvestmentJob < ApplicationJob
  queue_as :default

  @@api = ApiJob.new
  @@all_accounts = Account.all

  def perform
    total_investment_money = get_investment_money
    purchase_price = @@api.get_purchase_price
    amount_to_buy = (total_investment_money / purchase_price.to_f).to_f
    @@api.buy_btf(amount_to_buy)
    clp_before_sell = @@api.get_balance("CLP")
    amount_to_sell = @@api.get_balance("BTF")
    @@api.sell_btf(amount_to_sell)
    clp_after_sell = @@api.get_balance("CLP")
    earning_clp = clp_after_sell - clp_before_sell
    divide_earnings_between_users(earning_clp)
  end

  private
  def divide_earnings_between_users(earnings)
    amount_of_total_users = @@all_accounts.length
    clp_each_user = (earnings / amount_of_total_users)
    @@all_accounts.each do |account|
      account.saving_balance += clp_each_user
      account.save
    end
  end

  def get_investment_money
    total_investment_money = 0
    @@all_accounts.each do |account|
      # Le quitamos el 5% a cada usuario para invertir
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end
    total_investment_money
  end
end
```

```ruby
class InvestmentJob < ApplicationJob
  queue_as :default

  @@api = ApiJob.new
  @@all_accounts = Account.all

  def perform
    total_investment_money = get_investment_money
    purchase_price = @@api.get_purchase_price
    amount_to_buy = (total_investment_money / purchase_price.to_f).to_f
    @@api.buy_btf(amount_to_buy)
    clp_before_sell = @@api.get_balance("CLP")
    amount_to_sell = @@api.get_balance("BTF")
    @@api.sell_btf(amount_to_sell)
    clp_after_sell = @@api.get_balance("CLP")
    earning_clp = clp_after_sell - clp_before_sell
    divide_earnings_between_users(earning_clp)
  end

  private
  def divide_earnings_between_users(earnings)
    amount_of_total_users = @@all_accounts.length
    clp_each_user = (earnings / amount_of_total_users)
    @@all_accounts.each do |account|
      account.saving_balance += clp_each_user
      account.save
    end
  end

  def get_investment_money
    total_investment_money = 0
    @@all_accounts.each do |account|
      # Le quitamos el 5% a cada usuario para invertir
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end
    total_investment_money
  end
end
```

```ruby
class InvestmentJob < ApplicationJob
  queue_as :default

  @@api = ApiJob.new
  @@all_accounts = Account.all

  def perform
    total_investment_money = get_investment_money
    purchase_price = @@api.get_purchase_price
    amount_to_buy = (total_investment_money / purchase_price.to_f).to_f
    @@api.buy_btf(amount_to_buy)
    clp_before_sell = @@api.get_balance("CLP")
    amount_to_sell = @@api.get_balance("BTF")
    @@api.sell_btf(amount_to_sell)
    clp_after_sell = @@api.get_balance("CLP")
    earning_clp = clp_after_sell - clp_before_sell
    divide_earnings_between_users(earning_clp)
  end

  private
  def divide_earnings_between_users(earnings)
    amount_of_total_users = @@all_accounts.length
    clp_each_user = (earnings / amount_of_total_users)
    @@all_accounts.each do |account|
      account.saving_balance += clp_each_user
      account.save
    end
  end

  def get_investment_money
    total_investment_money = 0
    @@all_accounts.each do |account|
      # Le quitamos el 5% a cada usuario para invertir
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end
    total_investment_money
  end
end
```

```ruby
class InvestmentJob < ApplicationJob
  queue_as :default

  @@api = ApiJob.new
  @@all_accounts = Account.all

  def perform
    total_investment_money = get_investment_money
    purchase_price = @@api.get_purchase_price
    amount_to_buy = (total_investment_money / purchase_price.to_f).to_f
    @@api.buy_btf(amount_to_buy)
    clp_before_sell = @@api.get_balance("CLP")
    amount_to_sell = @@api.get_balance("BTF")
    @@api.sell_btf(amount_to_sell)
    clp_after_sell = @@api.get_balance("CLP")
    earning_clp = clp_after_sell - clp_before_sell
    divide_earnings_between_users(earning_clp)
  end

  private
  def divide_earnings_between_users(earnings)
    amount_of_total_users = @@all_accounts.length
    clp_each_user = (earnings / amount_of_total_users)
    @@all_accounts.each do |account|
      account.saving_balance += clp_each_user
      account.save
    end
  end

  def get_investment_money
    total_investment_money = 0
    @@all_accounts.each do |account|
      # Le quitamos el 5% a cada usuario para invertir
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end
    total_investment_money
  end
end
```

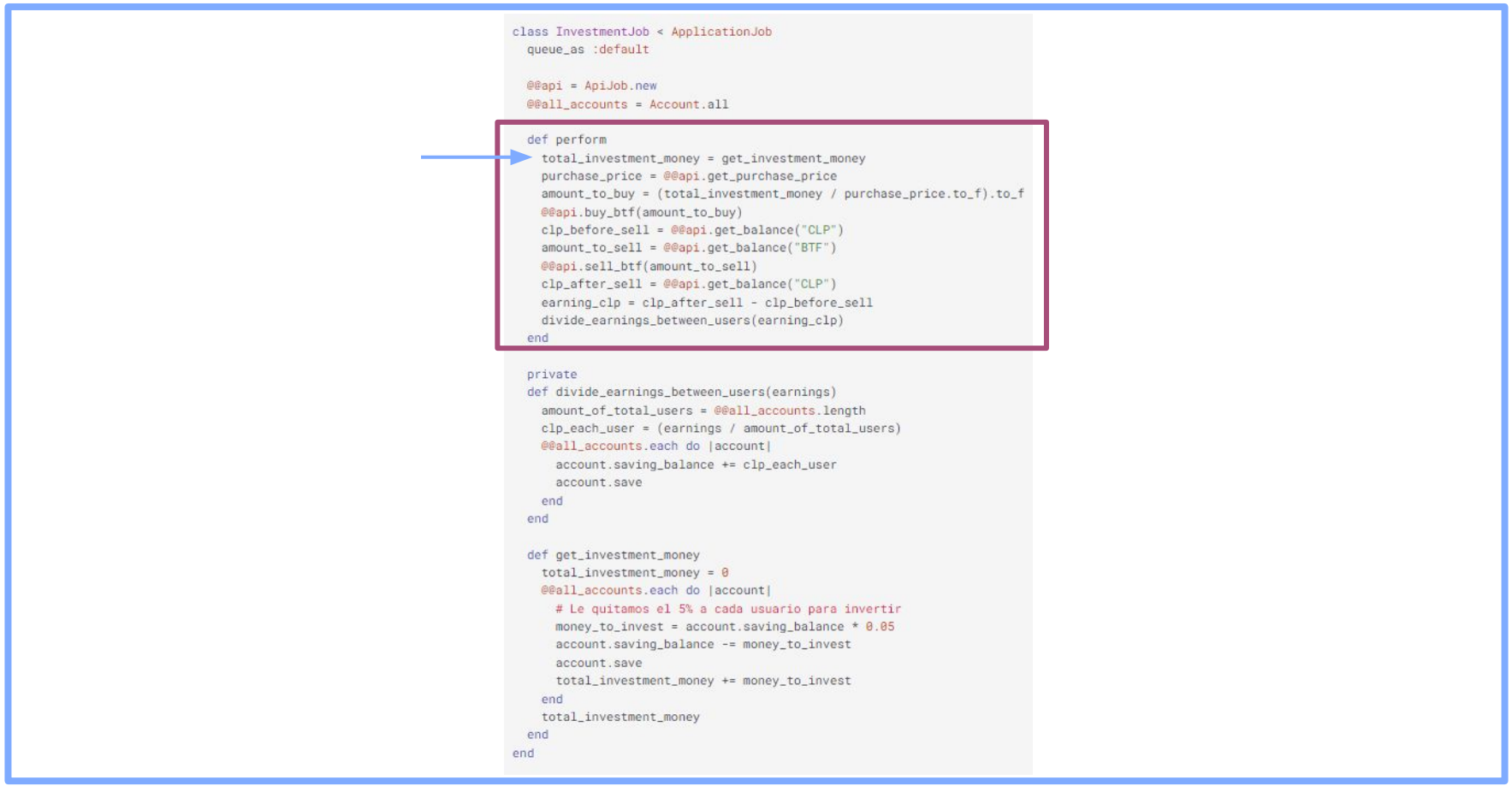
```ruby
class InvestmentJob < ApplicationJob
  queue_as :default

  @@api = ApiJob.new
  @@all_accounts = Account.all

  def perform
    total_investment_money = get_investment_money
    purchase_price = @@api.get_purchase_price
    amount_to_buy = (total_investment_money / purchase_price.to_f).to_f
    @@api.buy_btf(amount_to_buy)
    clp_before_sell = @@api.get_balance("CLP")
    amount_to_sell = @@api.get_balance("BTF")
    @@api.sell_btf(amount_to_sell)
    clp_after_sell = @@api.get_balance("CLP")
    earning_clp = clp_after_sell - clp_before_sell
    divide_earnings_between_users(earning_clp)
  end

  private
  def divide_earnings_between_users(earnings)
    amount_of_total_users = @@all_accounts.length
    clp_each_user = (earnings / amount_of_total_users)
    @@all_accounts.each do |account|
      account.saving_balance += clp_each_user
      account.save
    end
  end

  def get_investment_money
    total_investment_money = 0
    @@all_accounts.each do |account|
      # Le quitamos el 5% a cada usuario para invertir
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end
    total_investment_money
  end
end
```



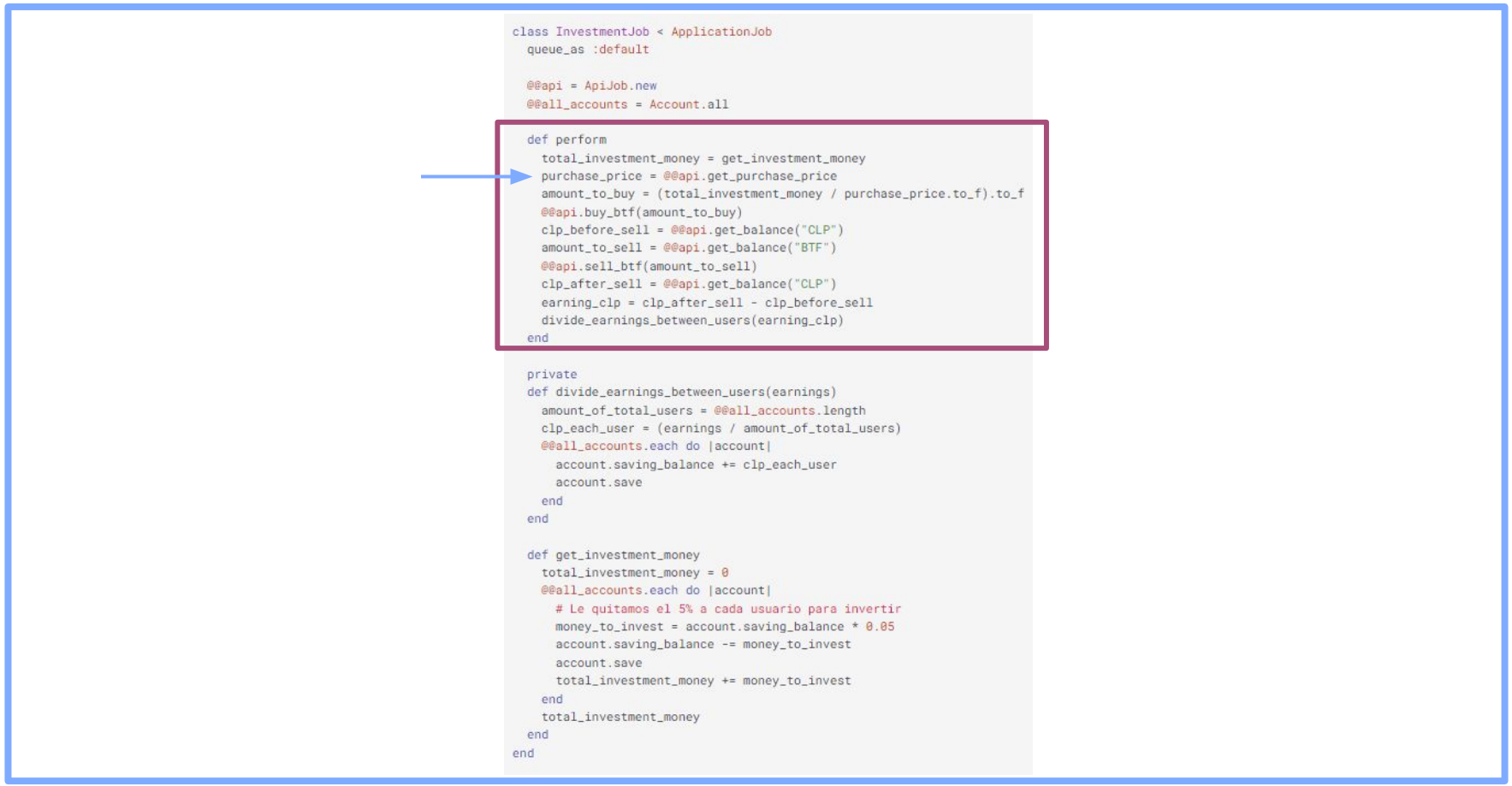
```ruby
class InvestmentJob < ApplicationJob
  queue_as :default

  @@api = ApiJob.new
  @@all_accounts = Account.all

  def perform
    total_investment_money = get_investment_money
    purchase_price = @@api.get_purchase_price
    amount_to_buy = (total_investment_money / purchase_price.to_f).to_f
    @@api.buy_btf(amount_to_buy)
    clp_before_sell = @@api.get_balance("CLP")
    amount_to_sell = @@api.get_balance("BTF")
    @@api.sell_btf(amount_to_sell)
    clp_after_sell = @@api.get_balance("CLP")
    earning_clp = clp_after_sell - clp_before_sell
    divide_earnings_between_users(earning_clp)
  end

  private
  def divide_earnings_between_users(earnings)
    amount_of_total_users = @@all_accounts.length
    clp_each_user = (earnings / amount_of_total_users)
    @@all_accounts.each do |account|
      account.saving_balance += clp_each_user
      account.save
    end
  end

  def get_investment_money
    total_investment_money = 0
    @@all_accounts.each do |account|
      # Le quitamos el 5% a cada usuario para invertir
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end
    total_investment_money
  end
end
```



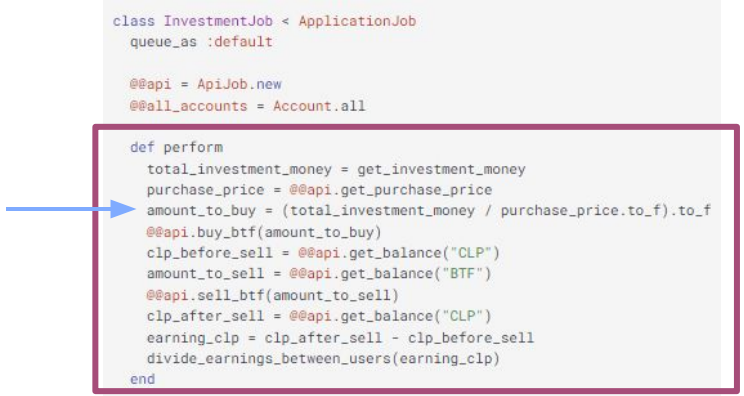
```ruby
class InvestmentJob < ApplicationJob
  queue_as :default

  @@api = ApiJob.new
  @@all_accounts = Account.all

  def perform
    total_investment_money = get_investment_money
    purchase_price = @@api.get_purchase_price
    amount_to_buy = (total_investment_money / purchase_price.to_f).to_f
    @@api.buy_btf(amount_to_buy)
    clp_before_sell = @@api.get_balance("CLP")
    amount_to_sell = @@api.get_balance("BTF")
    @@api.sell_btf(amount_to_sell)
    clp_after_sell = @@api.get_balance("CLP")
    earning_clp = clp_after_sell - clp_before_sell
    divide_earnings_between_users(earning_clp)
  end

  private
  def divide_earnings_between_users(earnings)
    amount_of_total_users = @@all_accounts.length
    clp_each_user = (earnings / amount_of_total_users)
    @@all_accounts.each do |account|
      account.saving_balance += clp_each_user
      account.save
    end
  end

  def get_investment_money
    total_investment_money = 0
    @@all_accounts.each do |account|
      # Le quitamos el 5% a cada usuario para invertir
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end
    total_investment_money
  end
end
```



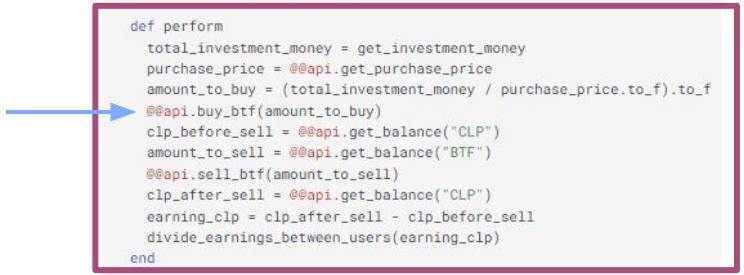
```ruby
class InvestmentJob < ApplicationJob
  queue_as :default

  @@api = ApiJob.new
  @@all_accounts = Account.all

  def perform
    total_investment_money = get_investment_money
    purchase_price = @@api.get_purchase_price
    amount_to_buy = (total_investment_money / purchase_price.to_f).to_f
    @@api.buy_btf(amount_to_buy)
    clp_before_sell = @@api.get_balance("CLP")
    amount_to_sell = @@api.get_balance("BTF")
    @@api.sell_btf(amount_to_sell)
    clp_after_sell = @@api.get_balance("CLP")
    earning_clp = clp_after_sell - clp_before_sell
    divide_earnings_between_users(earning_clp)
  end

  private
  def divide_earnings_between_users(earnings)
    amount_of_total_users = @@all_accounts.length
    clp_each_user = (earnings / amount_of_total_users)
    @@all_accounts.each do |account|
      account.saving_balance += clp_each_user
      account.save
    end
  end

  def get_investment_money
    total_investment_money = 0
    @@all_accounts.each do |account|
      # Le quitamos el 5% a cada usuario para invertir
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end
    total_investment_money
  end
end
```



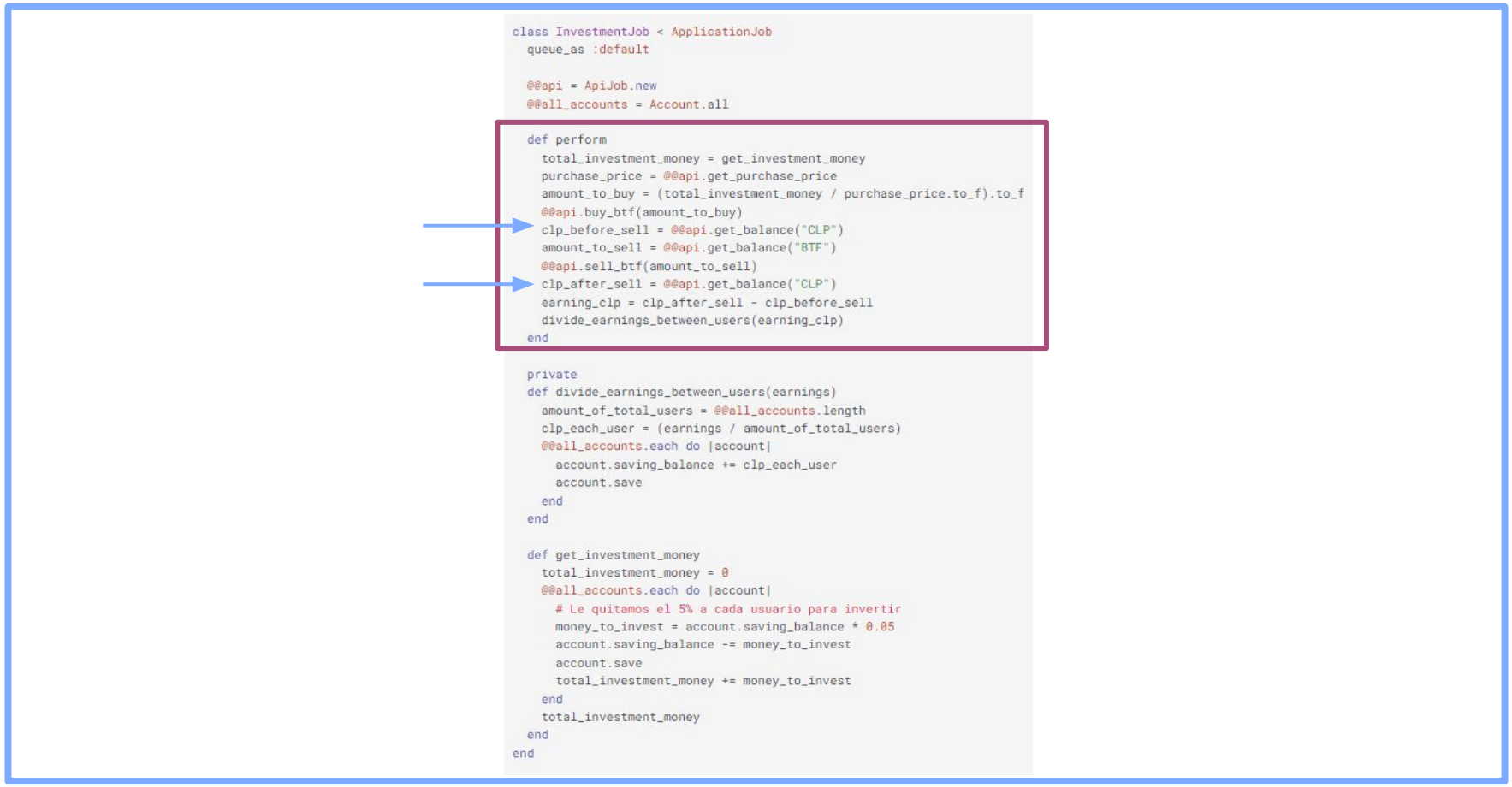
```ruby
class InvestmentJob < ApplicationJob
  queue_as :default

  @@api = ApiJob.new
  @@all_accounts = Account.all

  def perform
    total_investment_money = get_investment_money
    purchase_price = @@api.get_purchase_price
    amount_to_buy = (total_investment_money / purchase_price.to_f).to_f
    @@api.buy_btf(amount_to_buy)
    clp_before_sell = @@api.get_balance("CLP")
    amount_to_sell = @@api.get_balance("BTF")
    @@api.sell_btf(amount_to_sell)
    clp_after_sell = @@api.get_balance("CLP")
    earning_clp = clp_after_sell - clp_before_sell
    divide_earnings_between_users(earning_clp)
  end

  private
  def divide_earnings_between_users(earnings)
    amount_of_total_users = @@all_accounts.length
    clp_each_user = (earnings / amount_of_total_users)
    @@all_accounts.each do |account|
      account.saving_balance += clp_each_user
      account.save
    end
  end

  def get_investment_money
    total_investment_money = 0
    @@all_accounts.each do |account|
      # Le quitamos el 5% a cada usuario para invertir
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end
    total_investment_money
  end
end
```



```ruby
class InvestmentJob < ApplicationJob
  queue_as :default

  @@api = ApiJob.new
  @@all_accounts = Account.all

  def perform
    total_investment_money = get_investment_money
    purchase_price = @@api.get_purchase_price
    amount_to_buy = (total_investment_money / purchase_price.to_f).to_f
    @@api.buy_btf(amount_to_buy)
    clp_before_sell = @@api.get_balance("CLP")
    amount_to_sell = @@api.get_balance("BTF")
    @@api.sell_btf(amount_to_sell)
    clp_after_sell = @@api.get_balance("CLP")
    earning_clp = clp_after_sell - clp_before_sell
    divide_earnings_between_users(earning_clp)
  end

  private
  def divide_earnings_between_users(earnings)
    amount_of_total_users = @@all_accounts.length
    clp_each_user = (earnings / amount_of_total_users)
    @@all_accounts.each do |account|
      account.saving_balance += clp_each_user
      account.save
    end
  end

  def get_investment_money
    total_investment_money = 0
    @@all_accounts.each do |account|
      # Le quitamos el 5% a cada usuario para invertir
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end
    total_investment_money
  end
end
```

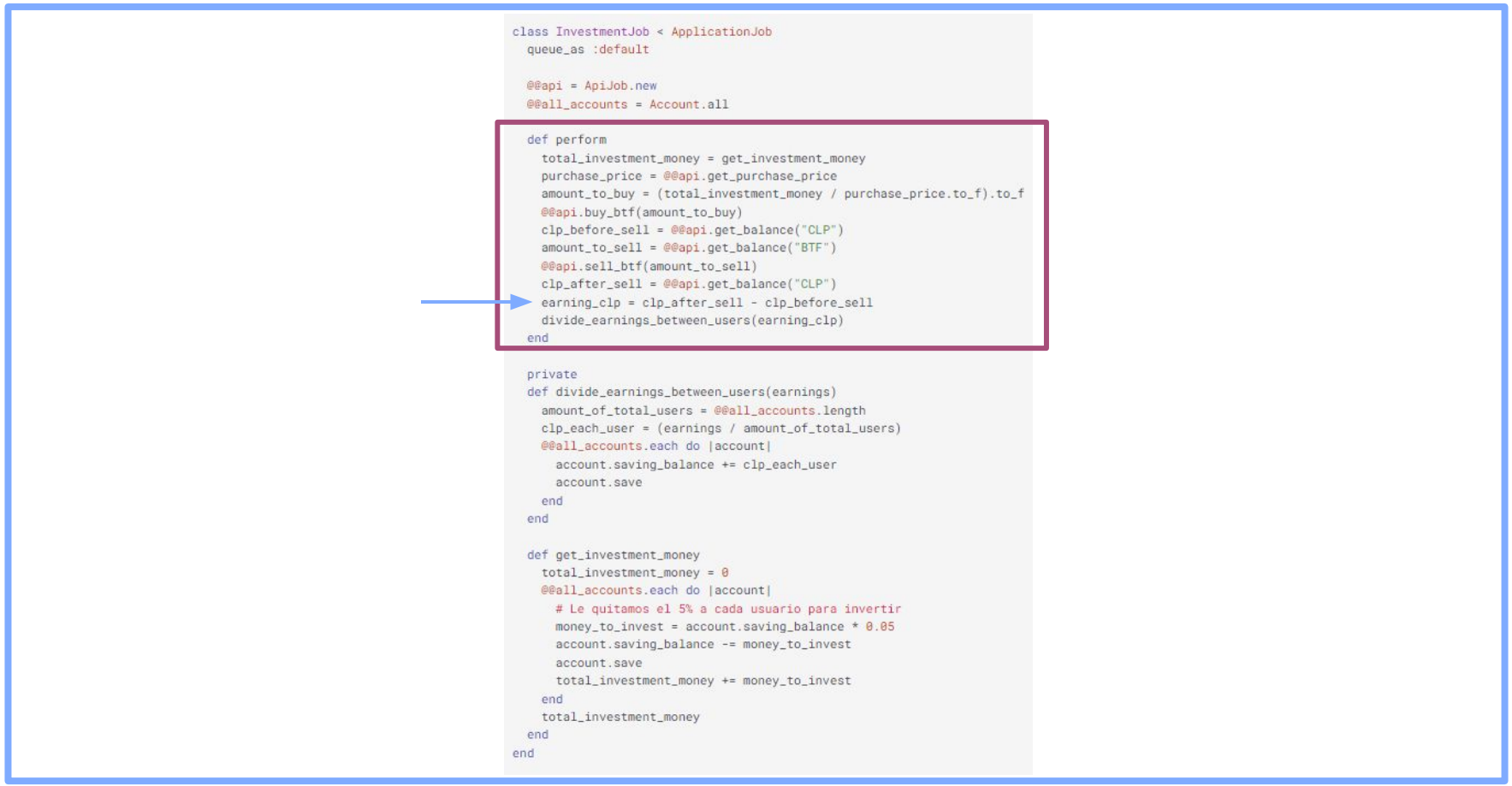
```ruby
class InvestmentJob < ApplicationJob
  queue_as :default

  @@api = ApiJob.new
  @@all_accounts = Account.all

  def perform
    total_investment_money = get_investment_money
    purchase_price = @@api.get_purchase_price
    amount_to_buy = (total_investment_money / purchase_price.to_f).to_f
    @@api.buy_btf(amount_to_buy)
    clp_before_sell = @@api.get_balance("CLP")
    amount_to_sell = @@api.get_balance("BTF")
    @@api.sell_btf(amount_to_sell)
    clp_after_sell = @@api.get_balance("CLP")
    earning_clp = clp_after_sell - clp_before_sell
    divide_earnings_between_users(earning_clp)
  end

  private
  def divide_earnings_between_users(earnings)
    amount_of_total_users = @@all_accounts.length
    clp_each_user = (earnings / amount_of_total_users)
    @@all_accounts.each do |account|
      account.saving_balance += clp_each_user
      account.save
    end
  end

  def get_investment_money
    total_investment_money = 0
    @@all_accounts.each do |account|
      # Le quitamos el 5% a cada usuario para invertir
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end
    total_investment_money
  end
end
```



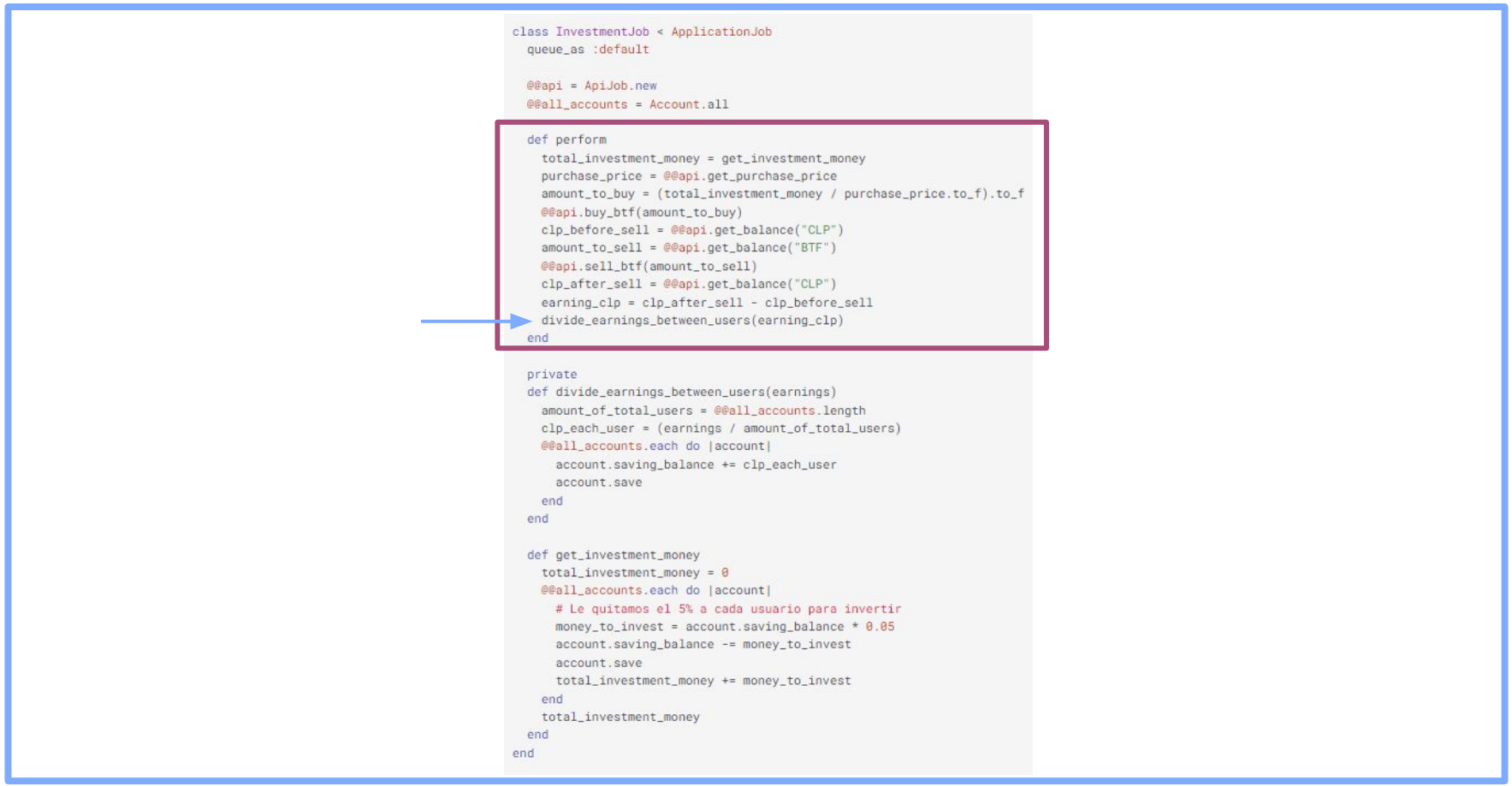
```ruby
class InvestmentJob < ApplicationJob
  queue_as :default

  @@api = ApiJob.new
  @@all_accounts = Account.all

  def perform
    total_investment_money = get_investment_money
    purchase_price = @@api.get_purchase_price
    amount_to_buy = (total_investment_money / purchase_price.to_f).to_f
    @@api.buy_btf(amount_to_buy)
    clp_before_sell = @@api.get_balance("CLP")
    amount_to_sell = @@api.get_balance("BTF")
    @@api.sell_btf(amount_to_sell)
    clp_after_sell = @@api.get_balance("CLP")
    earning_clp = clp_after_sell - clp_before_sell
    divide_earnings_between_users(earning_clp)
  end

  private
  def divide_earnings_between_users(earnings)
    amount_of_total_users = @@all_accounts.length
    clp_each_user = (earnings / amount_of_total_users)
    @@all_accounts.each do |account|
      account.saving_balance += clp_each_user
      account.save
    end
  end

  def get_investment_money
    total_investment_money = 0
    @@all_accounts.each do |account|
      # Le quitamos el 5% a cada usuario para invertir
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end
    total_investment_money
  end
end
```



```ruby
class InvestmentJob < ApplicationJob
  queue_as :default

  @@api = ApiJob.new
  @@all_accounts = Account.all

  def perform
    total_investment_money = get_investment_money
    purchase_price = @@api.get_purchase_price
    amount_to_buy = (total_investment_money / purchase_price.to_f).to_f
    @@api.buy_btf(amount_to_buy)
    clp_before_sell = @@api.get_balance("CLP")
    amount_to_sell = @@api.get_balance("BTF")
    @@api.sell_btf(amount_to_sell)
    clp_after_sell = @@api.get_balance("CLP")
    earning_clp = clp_after_sell - clp_before_sell
    divide_earnings_between_users(earning_clp)
  end

  private
  def divide_earnings_between_users(earnings)
    amount_of_total_users = @@all_accounts.length
    clp_each_user = (earnings / amount_of_total_users)
    @@all_accounts.each do |account|
      account.saving_balance += clp_each_user
      account.save
    end
  end

  def get_investment_money
    total_investment_money = 0
    @@all_accounts.each do |account|
      # Le quitamos el 5% a cada usuario para invertir
      money_to_invest = account.saving_balance * 0.05
      account.saving_balance -= money_to_invest
      account.save
      total_investment_money += money_to_invest
    end
    total_investment_money
  end
end
```

Desafíos

- No introducir más code smells
  - Era difícil introducir object-orientation-abusers
  - Mucho cuidado con dispensables
  - Cuidado con excesivos comentarios
  - Cuidado con código duplicado
  - Cuidado con lazy class
  - Cuidado con **dead code y speculative generality**

Desafíos

- No introducir más code smells
- Testeo manual para comprobar el funcionamiento del código

Gracias!

Preguntas?

Credits

This is where you give credit to the ones who are part of this project.

Did you like the resources on this template? Get them for **free** at our other websites.

Presentation template by Slidesgo

Icons by Flaticon

Images & infographics by Freepik

Author introduction slide photo created by **katemangostar** - Freepik.com

Big image slide photo created by **jcomp** - Freepik.com

Text & Image slide photo created by **rawpixel.com** - Freepik.com

Text & Image slide photo created by **Freepik**