



# Single Responsibility Principle

# How much behaviour must a class implement?



# Usefulness

Classes need to do the smallest possible useful thing.

```
class Printer
  def print(string)
    puts string
  end
end
```

# Do not think ahead

```
class User
  # [...]

  def authenticate(username, password)
    # [...]
    @credentials.first.verify(username, password)
    # [...]
  end
end
```

# Easy to extend?

- No side effects
- Small changes in requirements require small changes in code
- Existing code is easy to reuse
- The easiest way to make a change is to add code that is also easy to change itself

# Describe it!

One way to know that a class is doing too much is by describing it.

If you are using a lot of *ands* and *ors*, it's probably doing too much.

Let's see an example.

# Describe it!

```
require 'digest/sha1'

class User
  # [...]

  def check_password(password)
    Digest::SHA1.hexdigest password == @password
  end
end
```

# Describe it!

```
require 'digest/sha1'

class User
  # [...]

  def login_in_facebook(username, password)
    HTTP::Post('http://facebook.com/api/auth', {user: username, password: password})
  end

  def check_password(password)
    return true if Digest::SHA1.hexdigest password == @password
    login_in_facebook(@facebook_username, password)
  end
end
```



# Describe it!

```
require 'digest/sha1'

class User
  # [...]

  def login_in_facebook(username, password)
    http_response = HTTP::Post('http://facebook.com/api/auth', {user: username, password:
password})
    http_response == 200
  end

  def login_in_twitter(username, password)
    http_response = HTTP::Get('http://twitter.com/auth', {user: username, password:
password})
    http_response.body == 'OK'
  end

  def check_password(password)
    return true if Digest::SHA1.hexdigest password == @password
    return true if login_in_facebook(@facebook_username, password)
    login_in_twitter(@twitter_username)
  end
end
```

If your description is too long  
or contains some 'or' or 'and'  
maybe you need to think  
twice

# Exercise

Create a command line tool. This command line tool will ask you for a user and a password. If it's correct it will ask to enter a text and the program will count the number of words.

Passwords can be hardcoded into the source code.

# Anti-patterns / Smells



# God objects

A class that seems to know everything about your application.





# Shotgun surgery

A change is spread among different small changes in different objects.

Often, this means duplication that should be extracted into its own class



# Shotgun surgery

```
# [...]
```

```
def buy  
  HTTP:  
  Stock  
  Cash.  
  # [...]  
end
```

```
def sell  
  HTTP:  
  Stock  
  Cash.  
  # [...]  
end
```

```
def char  
  HTTP:  
  Price.  
  # [...]  
end
```

```
# [...]
```



```
price}")
```



```
# [...]  
  
def buy(item)  
  HTTP::Post("http://twitter.com/api/new_tweet", "I just bought a #{item}")  
  Stock.for(item).increment  
  Cash.decrease  
  # [...]  
end  
  
def sell(item)  
  HTTP::Post("http://twitter.com/api/new_tweet", "I just sold a #{item}")  
  Stock.for(item).decrement  
  Cash.increase  
  # [...]  
end  
  
def change_price(item, price)  
  HTTP::Post("http://twitter.com/api/new_tweet", "The price for #{item} is #{price}")  
  Price.for(item) = price  
  # [...]  
end  
  
# [...]
```

```

# [...]

def buy(item)
  Tweet.new("I just bought a #{item}") .publish
  Stock.for(item) .increment
  Cash.decrease
  # [...]
end

def sell(item)
  Tweet.new("I just sold a #{item}") .publish
  Stock.for(item) .decrement
  Cash.increase
  # [...]
end

def change_price(item, price)
  Tweet.new("The price for #{item} is #{price}") .publish
  Price.for(item) = price
  # [...]
end

# [...]

class Tweet
  def initialize(text)
    @text = text
  end

  def publish
    HTTP::Post("http://twitter.com/api/new_tweet", text)
  end
end

```

# Feature envy

Feature envy reveals a method (or method-to-be) that would work better on a different class.

We often find it with repeated calls to a different class.

# Feature envy

```
class SellableItem
  # [...]

  def price
    @price
  end
end

class Cost
  # [...]

  def price
    @sellable_item.price
  end

  def summer_price
    @sellable_item.price * 0.9
  end

  def christmas_price
    @sellable_item.price * 1.1
  end
end
```

# CAPTAIN OBVIOUS



```
class SellableItem
  #[...]

  def price
    @price
  end

  def summer_price
    @price * 0.9
  end

  def christmas_price
    @price * 1.1
  end
end

class Cost
  #[...]
end
```

# Duplicated code

It is often the root of all evil



Your code should be  
DRY

# Weapons of our warfare





# Extract a method

```
class Passenger
  def enter_country
    control_list_hit = ConstrolList.find(@document_number)
    if control_list_hit
      if control_list_hit.most_wanted
        NotificationForAgent.new("Most Wanted criminal found!", self).save
      else
        NotificationForAgent.new("Person found!", self).save
      end
    end
  end

  verification = VerificationResult.new
  verification.document_number = passenger.document_number
  if @documentation.verify
    verification.correct = true
  else
    verification.correct = false
  end
  verification.save

  Transit.for_passenger(self).save
end
end
```



# Extract a method

```
class Passenger
  def enter_country
    control_list_hit = ControlList.find(@document_number)
    if control_list_hit
      if control_list_hit.most_wanted
        NotificationForAgent.new("Most Wanted criminal found!", self).save
      else
        NotificationForAgent.new("Person found!", self).save
      end
    end
  end

  verification = VerificationResult.new
  verification.document_number = passenger.document_number
  if @documentation.verify
    verification.correct = true
  else
    verification.correct = false
  end
  verification.save

  Transit.for_passenger(self).save
end
end
```



```

class Passenger
  def enter_country
    control_list_check

    verification = VerificationResult.new
    verification.document_number = passenger.document_number
    if @documentation.verify
      verification.correct = true
    else
      verification.correct = false
    end
    verification.save

    Transit.for_passenger(self).save
  end

  private
  def control_list_check
    control_list_hit = ConstrolList.find(@document_number)
    if control_list_hit
      if control_list_hit.most_wanted
        NotificationForAgent.new("Most Wanted criminal found!", self).save
      else
        NotificationForAgent.new("Person found!", self).save
      end
    end
  end
end
end

```



```

class Passenger
  def enter_country
    control_list_check

    verification = VerificationResult.new
    verification.document_number = passenger.document_number
    if @documentation.verify
      verification.correct = true
    else
      verification.correct = false
    end
    verification.save

    Transit.for_passenger(self).save
  end

  private
  def control_list_check
    control_list_hit = ConstrollList.find(@document_number)
    if control_list_hit
      if control_list_hit.most_wanted
        NotificationForAgent.new("Most Wanted criminal found!", self).save
      else
        NotificationForAgent.new("Person found!", self).save
      end
    end
  end
end
end

```



```

class Passenger
  def enter_country
    control_list_check()
    verify_document()
    Transit.for_passenger(self).save
  end

  private
  def control_list_check
    control_list_hit = ConstrolList.find(@document_number)
    if control_list_hit
      if control_list_hit.most_wanted
        NotificationForAgent.new("Most Wanted criminal found!", self).save
      else
        NotificationForAgent.new("Person found!", self).save
      end
    end
  end
end

def verify_document
  verification = VerificationResult.new
  verification.document_number = passenger.document_number
  if @documentation.verify
    verification.correct = true
  else
    verification.correct = false
  end
  verification.save
end
end

```

```

class Passenger
  def enter_country
    control_list_check()
    verify_document()
    Transit.for_passenger(self).save
  end

  private
  def control_list_check
    control_list_hit = ConstrolList.find(@document_number)
    if control_list_hit
      if control_list_hit.most_wanted
        NotificationForAgent.new("Most Wanted criminal found!", self).save
      else
        NotificationForAgent.new("Person found!", self).save
      end
    end
  end
end

def verify_document
  verification = VerificationResult.new
  verification.document_number = passenger.document_number
  if @documentation.verify
    verification.correct = true
  else
    verification.correct = false
  end
  verification.save
end
end

```



# Extract a class

```
class Passenger
  def enter_country
    AgentNotificator.new.create_for_passenger(self)
    DocumentVerificator.new.verify(@documentation)

    Transit.for_passenger(self).save
  end
end
```

# Extract 'til you drop

- Extract until you can't extract any more
- Then extract a little bit more
- Then you are done...
- Then extract a little bit more
- Now you are really done



# Replace conditional with polymorphism

```
class Passenger
  def enter_country
    # [...]
    case @type
    when :minor
      check_documentation tutor
    when :dependant
      check_documentation(self)
      check_can_be_dependant_of tutor
      save_biometrics
      control_list_check
    when :regular
      check_documentation(self)
      save_biometrics
      control_list_check
    end
    save_transit
  end
end
```

# Replace conditional with polymorphism

```
class Passenger
  def enter_country
    save_transit
  end
end
```

```
class Minor < Passenger
  def enter_country
    check_documentation tutor
    super
  end
end
```

```
class Dependant < Passenger
  def enter_country
    check_documentation(self)
    check_can_be_dependant_of tutor
    save_biometrics
    control_list_check
    super
  end
end
```

```
class RegularPassenger < Passenger
  def enter_country
    check_documentation(self)
    save_biometrics
    control_list_check
    super
  end
end
```

# Refactor

Refactoring is the process of changing a software system in such a way that it does not alter the external behaviour of the code yet improves its internal structure.

# Example code

```
if output == :screen
  puts text
elsif output == :logfile
  IO.write("log.log", text)
elsif output == :message_queue
  Queue.put(text)
end
```



The next programmer will often be a lazy dog

The next programmer will often be you



# Example code

```
if output == :screen
  puts text
elsif output == :logfile
  IO.write("log.log", text)
elsif output == :message_queue
  Queue.put(text)
elsif output == :json_endpoint
  HTTP.post(text)
elsif output == :logstash
  Logstash.add(text)
elsif output == :logwatcher
  LogWatch.new.post(text)
elsif output == :logfile
  IO.write("log.log", text)
elsif output == :message_queue
  Queue.put(text)
elsif output == :json_endpoint
  HTTP.post(text)
elsif output == :logstash
```



# Refactor this!

```
if output == :screen
  puts text
elsif output == :logfile
  IO.write("log.log", text)
elsif output == :message_queue
  Queue.put(text)
elsif output == :json_endpoint
  HTTP.post(text)
elsif output == :logstash
  Logstash.add(text)
elsif output == :logwatcher
  LogWatch.new.post(text)
elsif output == :logfile
  IO.write("log.log", text)
elsif output == :message_queue
  Queue.put(text)
elsif output == :json_endpoint
  HTTP.post(text)
elsif output == :logstash
  Logstash.add(text)
elsif output == :logwatcher
```