Programowanie obiektowe

Klasy wprowadzenie

Dziedziczene (inheritance)

Szkolenie Rails

- Wzorowane biologią.
- Klasa , która dziedziczy ma takie same cechy oraz metody jak klasa po , której dziedziczy.
- Nie możemy dziedziczyć po kilku klasach.
- Class Player < Person. Używamy znaku < , aby zaznaczyć dziedziczenie.

Wprowadzenie

Szkolenie Rails

Programowanie obiektowe zmieniło sposób myślenia programistów o świecie. Wcześniej używano programowania proceduralnego (funkcjonalnego) , które używało tylko funkcji i metod.

Ta technika programistyczna na trfałe zmieniła podejście do programowania i jest używana do dnia dzisiejszego.

Dzięki programowaniu obiektowemu możemy przenieść to co w świecie rzeczywistym 1 do 1.

Klasa

Szkolenie Rails

Jest to definicja dla obiektów.

Definiujemy zachowanie , cechy , aby móc korzystać później z naszej definicji.

Jest to zbiór obiektów o podobnych właściwościach.

Przykładowa klasa to Osoba, Użytkownik, Telefon, Faktura

Definiujemy klasę jako rzeczownik w liczbie pojedynczej.

Czy zawsze używać Klas???

W większości przypadkach przy komercyjnych rozwiązań tak.

Railsy niejako wymuszają stosowania „programowania obiektowego” , ponieważ Kontroler , Model , Widok są klasami.

Jednak nie popadajmy w przesadę np. do rozwiązania prostego problemu piszemy zwyczajną funkcję.

Jak pisać programy???

Szkolenie Rails

- Tak ,aby inne osoby mogłby z naszego kodu korzystać , nie tylko my.
- Inny programista powinien jak najmniej wiedzieć „co się dzieje w środku.

- **PODSTAWOWA ZASADA DRY**

ZASADA DRY

Szkolenie Rails

- Don't REPEAT YOURSELF.
- Zasada według , której jeśli mamy jakieś powtórzenie w kodzie oznacza ,że możemy poprawić kod , tak aby uniknąć powtórzeń.
- Podstawowe pytanie jakie musimy sobie zadać: „Czy mamy powtórzenie kodu w naszym programie??”

Przykład Klasy Ruby

Szkolenie Rails

- nazwa klasy musi zaczynać się dużą litery
- nazwa klasy kończymy słowem kluczowym end

```
class Person

end
```

Dołączenie pliku klasy

Szkolenie Rails

- Przykład dołączenia pliku klasy.
- W pliku s.rb znajduje się definicja klasy Person.

```
require './s.rb'
puts Person.new
```

Obiekt

Szkolenie Rails

- Jest to struktura , która przechowuje dane oraz metody.
- Definiujemy zachowanie , cechy , aby móc korzystać później z naszej definicji.
- Każdy obiekt ma 3 cechy: tożsamość (odróżnienie od innych obiektów), stan (czyli aktualny stan składowych) , zachowanie
  - czyli zestaw metod wykonujących operacje na tych danych.
- Dla przykładu Person obiektem jest konkretna osoba czyli np. Michał Makaruk, Piotr Kowalski , itd.

Konstruktor

Szkolenie Rails

- Konstruktor to specjalna metoda , która tworzy obiekt.
- Aby w Ruby stworzyć dany obiekt używamy słowa kluczowego new
- Przykładowo Person.new

Initialize – przykład z hashem

Szkolenie Rails

```
class Person
def initialize(hash={})
@name = hash[:name] if hash.has_key?(:name)
@surname = hash[:surname] if hash.has_key?(:surname)
end
def name
@name
end
end
puts Person.new(name:"Adam",surname:"Kowalski").name
```

Initialize -tworzy obiekt -inicjalizuje

Szkolenie Rails

```
Class Person

Def initialize

Puts "Created object"

end

end

Person.new
```

- Gdy robimy new wywoływana jest metoda initalize.

Zmienne obiektu

- Definują stan obiektu.
- Definiujemy je za pomocą @ przykładowo @name = 'Michal'
- „na zewnątrz” nie mamy standardowo do nich dostępu.
- Definiujemy je w metodzie initalize.

Przykład zmiennej obiektu

Szkolenie Rails

- Przykład name dla Person.

```
class Person

def initialize
@name ='Michal'
puts "Created object"
end

end
puts Person.new.name
```

Hermetyzacja

Szkolenie Rails

- hermetyzacja polega na ukrywaniu pewnych danych składowych lub metod obiektów danej klasy , tak aby były one dostępne tylko metodom wewnętrznym danej klasy

Przyczyny stosowania:

- uodparnia tworzone obiekty na błędy
- lepiej oddaje rzeczywistość
- umożliwia rozbiecie na mniejsze kawałki

Źródło: http://pl.wikipedia.org/wiki/Hermetyzacja_(informatyka)

Przykład Ruby -Hermetyzacja

Szkolenie Rails

- Przykład definicji dostępu do zmiennych.

```
class Person
def initialize
end
def name=(a)
@name =a if a.is_a?(String)
end
def name
if @name
   return @name
else
return   "Undefined name"
end
end
end
p = Person.new
puts p.name
p.name = "Adam"
puts p.name
```

Akcesory

Szkolenie Rails

- Akcesory - to specjalne metody dzięki , której możemy odwoływać się do zmiennych obiektu.
- Setter - ustawia wartość
- Getter - pobiera wartość
- W poprzednich ćwiczeniach sami definiowaliśmy te metody.

Ruby -akcesory standardowo.

- attr_accessor - ustawia setter i getter.
- attr_writter - ustawia setter.
- attr_reader - ustawia getter

Przykład accessor

Szkolenie Rails

- Przykład accessor name.

```
class Person
attr_accessor :name
def initialize(name=nil)
@name = name
end
end
p = Person.new
p.name = "Adam"
puts p.name
```

Dziedziczene (inheritance)

Szkolenie Rails

- Wzorowane biologią.
- Klasa , która dziedziczy ma takie same cechy oraz metody jak klasa po , której dziedziczy.
- Nie możemy dziedziczyć po kilku klasach.
- Class Player < Person. Używamy znaku < , aby zaznaczyć dziedziczenie.

Dziedziczene - przykład

Szkolenie Rails

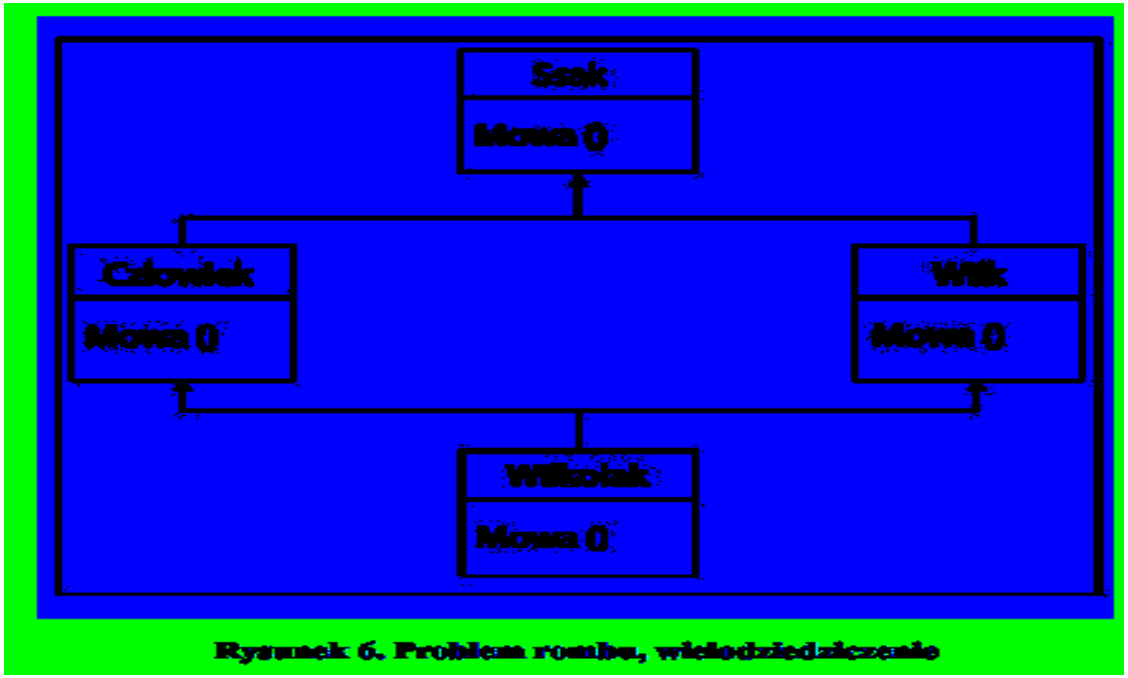
class Person

- end
- class Player < Person
- End
- Znak < - oznacza dziedziczenia, klasa Player dziedziczy po klasie Person.

Problem związany z wielodziedziczeniem

Szkolenie Rails

- Problem rombu



Rysunek 6. Problem rombu, wielodziedziczenie

Overriding

Szkolenie Rails

- Nadpisywanie metody.
- Jeśli mamy klasę dziedziczącą po klasie bazowej I chcemy zmienić zachowanie metody , która była zdefiniowana w klasie nadrzędnej możemy ją nadpisać.

Overriding -nadpisywanie metod -przykład

Szkolenie Rails

```
class Person
attr_accessor :name
def hello
puts "Hello"
end

end
class Adam <Person
def hello
puts "Hello Adam"
end
end
Adam.new.hello
```

Super przykład

- Za pomocą super możemy odwoływać się do metody zdefiniowanej z klasy, po której dziedziczymy.
- Używamy kiedy chcemy zmienić metodę jednocześniej korzystając z wcześniej zdefiniowanej metody w klasie , po której dziedziczymy.

Super przykład

Szkolenie Rails

- Super with initialize.

```
class Person
attr_accessor :name
def initialize(name=nil)
@name = name
end
end
class Adam <Person
def initialize(name,extra)
super(name)
puts extra
end
end
Adam.new("AA",22)
```

Metody dostępność

Szkolenie Rails

- Standardowo metody są public w Ruby. Są one widoczne przez inne obiekty korzystające z klas
- Private mogą korzystać tylko metody obiektu z tych metod.
- Protected mogą korzystać także klasy dziedziczące z danej klasy.

Odwołanie się przez referencje


```
class Person

attr_accessor :name

end

p1 =Person.new

p2 = Person.new

p1.name = "Adam"

p2 =p1

p1.name = "Piotr"

puts p2.name
```

Self

Szkolenie Rails

- Self daje dostęp do obecnego obiektu jakiego używamy.
- Inny sposób na initialize:

```
class Person

puts self

attr_accessor :name

def initialize(name)

self.name = name

Puts self

end
```

Zmienne i metody statyczne

- zmienna statyczna to taka , która jest taka sama dla wszystkich obiektów inaczej nazywana zmienna klasowa.
- Metoda statyczna to taka metoda , która jest taka sama dla wszystkich obiektów. Inaczej nazywana metodą klasową.
- Zmienne klasowe definiujemy za pomocą @@nazwa_zmiennej.

Metoda statyczna Ruby

Szkolenie Rails

- przykładowa metoda statyczna może znajdować wszystkich użytkowników.
- Najczęściej definiujemy ją za pomocą słowa self.
- Do metod statycznych odwołujemy się używając znaku .
- Przykład User.get_all_users.
- User.last - to metoda statyczna ActiveRecord.
- Większość metod z, których korzystaliśmy np. Last,first, where to metody statyczne.

Metoda statyczna Przykład


```
class Post

  def self.print_author

    puts "The author of all posts is Jimmy"

  end

end

Post.print_author

# "The author of all posts is Jimmy"
```

Przeciążenie operatorów

Szkolenie Rails

- Nie we wszystkich je ̨zykach istnieje możliwosść definiowania operatorów dla klas. W języku Ruby programiści mogą korzystacć z tych możiwości. W języku tym operatory traktowane są jak zwykłe metody działające na klasach.

Przeciążenie operatorów -przykład

Szkolenie Rails

```
Class Punkt

attr_reader :x,:y

Def initialize(x,y)

@x,@y = x,y

End

Def +(inny)

End

end
```

Metoda to_s

Szkolenie Rails

- zwraca czytelną wartość naszego obiektu
- nadpisujemy ją , aby móc skorzystać np. z puts obiekt wtedy nie dostejemy numeru obiektu w pamięci tylko czytelną dla nas postać
- tak samo w Railsach możemy z niej korzystać.

STAŁE

Szkolenie Rails

Używamy pisząc wielkimi Literami np.

Class Tax

VAT=23

end

MODUŁ

Szkolenie Rails

Dana klasa może „dołączyć” wiele modułów.

Możemy porównać ten mechanizm do wielodziedziczenia.

W module najczęściej definiujemy zestaw metod, z których mogą korzystać nasze klasy.

Dołączamy moduł do klasy za pomocą słowa kluczowego **include**

MODUŁ PRZYKŁAD

Szkolenie Rails

```
module Brzeszczyk
  def dzwon
    puts "BZZZZ!BZZZZ!BZZZZ!"
  end
end

class Czasomierz
  def podaj_czas
    puts Time.now
  end
end

class Budzik < Czasomierz
  include Brzeczyk
end
b = Budzik.new
b.podaj_czas #=> Sun Aug 05 17:24:08 +0200 2007
b.dzwon      #=> BZZZZ!BZZZZ!BZZZZ!
```