Programlama Dilleri Laboratuvar Föyü
Ruby Programlama Dili 3
Metotlar, Sınıflar ve Hata Yakalama

Şevket Umut ÇAKIR

1 Giriş

Bu deneyde Ruby programlama dilindeki metotlar, sınıflar ve hata yakalama işlemleri anlatılmaktadır.

2 Konu Anlatımı ve Deney Hazırlığı

2.1 Metotlar

Ruby metotları başka programlama dillerindeki fonksiyonlara benzerdir. Metotları kullanmanın amacı bir veya daha fazla tekrarlanabilecek kod parçacıklarını tek bir isim altında toplamaktır. Metotlar Ruby dilinde küçük harfle başlamak zorundadır, aksi takdirde bir sabit olarak algılanabilir. Metotlar çağrılmadan önce tanımlanmalıdır. Metotları tanımlamak için `def` anahtar kelimesi kullanılmalıdır.

Bir metot parametresi olmadan veya birden fazla parametre alacak şekilde tanımlanabilir. Örnekleri aşağıda mevcuttur.

```
def merhaba_de
  puts "Merhaba"
end

def selamla(isim)
  puts "Merhaba #{isim}!"
end
```

Metot parametrelerinin varsayılan değerleri belirlenebilir. Metot çağrılırken parametre değeri verilmemişse varsayılan değerler alınır.

```
def test(d1="Perl", d2="Ruby")
  puts "Diller: #{d1} ve #{d2}"
end
test            # Diller: Perl ve Ruby
test "C", "C++" # Diller: C ve C++
```

Her Ruby metodu geriye bir değer döndürür. Ruby metodunun son satırı geri dönecek değerdir. Bunun yanı sıra `return` anahtar kelimesi ile bir veya daha fazla değer metottan geri döndürülebilir.

```
def topla(a,b)
  a+b #son ifade geri dönüş değeridir
end

def toplam_carpim(a,b)
  return a+b, a*b
end

sayi1 = 3
sayi2 = 5
puts "Toplam: #{topla sayi1, sayi2}"
t, c = toplam_carpim sayi1, sayi2
puts "Toplam: #{t}, Çarpım: #{c}"
tc = toplam_carpim sayi1, sayi2
puts tc # tc değişkeni Array tipindedir.
```

Metotlara değişken sayıda parametre verilebilir. Bunun için metot tanımında değişkenin önüne * sembolü getirmek yeterlidir.

```
def yazdir(*args)
  args.each_with_index {|p, i| puts "#{i}. parametre: #{p}"}
end

yazdir 1,2,"merhaba"
yazdir 3.14, "dünya!", "", "!", 5
```

2.2 Sınıflar

Ruby saf bir nesneye yönelik programlama dilidir. Bunun anlamı Ruby dilinde en basit tipler dahil olmak üzere her şey bir nesnedir. Hatta her sınıf `Class` sınıfının birer örneği(instance) olan nesnelerdir.

Sınıf oluşturmak için `class` anahtar kelimesi ile başlanır, metotlar ve değişkenler yazıldıktan sonra `end` anahtar kelimesi ile bitirilir. Sınıf isimleri büyük harfle başlamak zorundadır. `initialize` metodu yapıcı(constructor) görevindedir. Nesneye ait örnek değişkenleri(instance variable) oluşturmak için değişkenin önüne @ sembolü getirilir.

Nesnelerin örnek değişkenlerine dışarıdan erişebilmek için sınıf tanımlamasında üç farklı özellik kullanılabilir. `attr_accessor` okuma ve yazma, `attr_reader` sadece okuma ve `attr_writer` sadece yazma için kullanılan özelliklerdir. Bu özellikler kullanılmadan örnek değişkenlerine isimleriyle erişmek mümkün değildir. Örnek değişkenlerine erişmek için metotlar yazılabilir ama özellikler bunun için kısa bir yol sunar.Aşağıda örnek bir sınıf tanımlaması ve nesne oluşturup kullanılması verilmiştir:

```ruby
class Araba
  def initialize(marka, model, model_yili)
    @marka = marka
    @model = model
    @model_yili = model_yili
  end
  def yazdir
    puts "Araba => marka: #{@marka}, model: #{@model}, model
    ↪  yılı: #{@model_yili}"
  end
  attr_accessor :marka #yazma ve okuma
  attr_reader :model # sadece okuma
  attr_writer :model_yili # sadece yazma
end

araba = Araba.new("Volkswagen", "Passat", 2015)
araba.yazdir
araba.marka = "Renault"
puts araba.marka
puts araba.model
araba.model_yili = 2000
araba.yazdir
```

Sınıf değişkenleri ve metotları örneklere değil de sınıfa aittir. Başka programlama dillerindeki *static* öğeler gibi davranırlar. Sınıf değişkeni oluşturmak için @@ sembolü değişkenin önüne eklenir. Sınıf metodu oluşturmak için ise tanımlama `self.metot_adi` şeklinde yapılır. `to_s` metodu nesnenin metin temsilidir.

```ruby
class Araba
  @@adet = 0 # sınıf değişkeni
  def initialize(marka, model, model_yili)
    @marka, @model, @model_yili = marka, model, model_yili
    @@adet += 1
  end
  def self.adet_yaz() # sınıf metodu
    puts "Araba sayısı: #{@@adet}"
  end
  def to_s
    "Araba => marka: #{@marka}, model: #{@model}, model yılı:
    ↪  #{@model_yili}"
  end
end

a1 = Araba.new "Ferrari", "458", 2010
a2 = Araba.new "Lamborghini", "Gallardo", 2014
puts a1,a2
Araba.adet_yaz
```

Ruby programlama dilinde kalıtım < sembolü ile gerçekleştirilir. Aşağıda `Araba` sınıfından türetilen `ElektrikliAraba` sınıfı vardır. `ElektrikliAraba` sınıfında `to_s` metodu geçersiz kılınmıştır.

```
class ElektrikliAraba < Araba
  attr_accessor :batarya_kapasitesi
  def initialize(marka, model, model_yili, batarya_kapasitesi)
    super marka, model, model_yili
    @batarya_kapasitesi = batarya_kapasitesi
  end
  def to_s
    "#{super.to_s}, batarya kapasitesi: #{@batarya_kapasitesi}"
  end
end

tesla = ElektrikliAraba.new 'Tesla', "Model S", 2018, "60 kWh"
puts tesla
```

2.3 Hata Yakalama

Hata yakalama işlemi `begin` ve `end` blokları içinde gerçekleştirilir. Bu blok içindeki `rescue` ifadeleri ile hatalar yakalanır.

```
begin
  sayi = 1 / 0
rescue ZeroDivisionError => e
  puts e.message
rescue StandardError
  puts "hata"
else
  puts "Hata yok"
ensure
  puts "Hep çalışır"
end
```

3 Deneyin Uygulanması

Bu deneyde hata yakalama uygulaması, metot yazma ve sınıf yazma örnekleri bulunmaktadır.

3.1 Sayıya Böl Topla

Kullanıcı boş metin girene kadar kullanıcıdan tam sayı alacak olan ve 1000 sayısını alınan sayılara bölerek toplayan kodu yazınız. Kullanıcıdan gelen değerler sadece tam sayılar veya geçerli olmayabilir. Eğer kullanıcı sıfır girmişse toplama herhangi bir değer eklenmeyecektir. Buradaki amaç hatalı durumları yakalayan bir uygulama yazmaktır. Bir ifadenin tamsayıya dönüştürülmesi sırasındaki

hatalar yakalanmak isteniyorsa `Integer(ifade)` şeklinde bir kod kullanılabilir. Hatalı dönüşümlerde `ArgumentError` hatası meydana gelecektir. Programın örnek girdi için verdiği sonuç aşağıdadır:

```
sayı girin: 3
sayi girin: merhaba
sayı girin: 2.5
sayı girin: 0
sayı girin:
833
```

3.2 Ters Faktoryel

Ters faktoryel verilen bir sayının kaçın faktoryeli olduğunu bulmaktır. Örneğin 120 sayısı 5!'e eşittir. Ya da 3628800 sayısı 10!'e eşittir. Parametre olarak verilen bir değerin hangi sayının faktoryeli olduğunu geri döndüren `ters_faktoryel` metodunu yazınız. Eğer parametre olarak gönderilen değer bir sayının faktoryeli değilse `raise ArgumentError` ile bir hata oluşturulmalıdır.

3.3 Bak ve Söyle

Bak ve söyle ardışık sayılarında bir sayı kullanılarak bir sonraki sayıyı üretilmektedir. Bunun için sırasıyla art arda gelen rakamların kaç adet olduğu ve bu rakamın değeri okunur. Örneğin 1 sayısı ile başlayıp bu işlemi 5 defa tekrarlarsak aşağıdaki gibi bir sonuç ortaya çıkar.
$1 \rightarrow 11$
$11 \rightarrow 21$
$21 \rightarrow 1211$
$1211 \rightarrow 111221$
$111221 \rightarrow 312211$

Başlangıç değerini ve kaç tekrar yapılacağını parametre olarak alan ve sonucu döndüren `bak_ve_soyle` metodunu yazınız.

3.4 Ev Hesabı Takibi

Bir öğrenci evindeki harcamaların takibini ve ay sonunda kimin kime borçlu olduğunu hesaplamak için bir yazılım geliştirilmek istenmektedir. Bunun için size verilen `Kisi`, `Harcama`, `Borc` ve `Ev` sınıflarını kullanarak ve aşağıdaki yalancı kodu dikkate alarak borçları hesaplayan kodu yazınız. Sizden istenen `Ev` sınıfının `borc_hesapla`, `kisi_ekle` ve `harcama_ekle` metotlarını yazmanızdır.

```
toplamHarcama=tüm kişilerin harcama toplamı//toplam harcamayı hesapla
ortalama=toplamHarcama/Kişi sayısı//ortalama gerekli harcama miktarını hesapla
foreach Kisi k in kisiler//kişilerin alacak değerlerini belirle
        k.alacak=k.harcama-ortalama
foreach Kisi b in borclular//tüm borçluları gez
        foreach Kisi a in alacaklilar//tüm alacaklıları gez
                if -b.alacak>a.alacak//borçlunun borcu alacaklının alacağından fazlaysa
                        b.borcEkle(new Borc(a,a.alacak))
                        b.alacak+=a.alacak
                        a.alacak=0
                else//borçlunun borcu alacaklının alacağından az ise
                        b.borcEkle(new Borc(b,-b.alacak))
                        a.alacak+=b.alacak
                        b.alacak=0
```