Podrška objektno orijentisanom programiranju u jezicima C++, Objective C, Java, C#, Ada i Ruby

Katarina Popović, Dušan Pantelić, Dejan Bokić, Nikola Stojević

Seminarski rad u okviru kursa
Metodologija stručnog i naučnog rada
Matematički fakultet

Maj 2019

Sadržaj

Uvod

C++

Objective C

Java

Ada

Ruby

Literatura

Uvod

- Programska paradigma
- Princip jedinstvene odgovornosti
- Enkapsulacija
- Nasleđivanje
- Polimorfizam
- Apstrakcija

C++

- C++ je delimično objektno orijentisan jezik
  - Main funkcija izvan klase
  - Koncept globalne promenljive
  - Postojanje friend funkcija
- Enkapsulacija u C++
  - public, protected i private sekcije
- Nasleđivanje u C++
  - public, protected i private nasleđivanje
  - virtuelno nasleđivanje
- Polimorfizam u C++
  - polimorfizam u vreme kompilacije
  - polimorfizam u vreme izvršavanja
- Apstrakcija u C++

Objective C

```
@interface Employee : NSObject {
   double salary; @public int age;}
@property(nonatomic, readwrite) double salary;
- (void)display;
@end # '-' za metode instance, '+' za klasne metode(static)
@implementation Employee
@synthesize salary;
- (void)display { NSLog(@"Employee salary is %f", salary); }
@end # (id) tip koji je kompaktibilan svakom objektu
@interface Driver : Employee { NSString* truck; }
- (id)initWithTruck:(NSString*)model;
@end # self oznacava tekuci objekat
@implementation Driver
- (id)initWithTruck:(NSString*)model {
   truck = model; return self;  }
- (void)display { NSLog(@"Driver salary is %f", salary); } @end
int main(int argc, const char * argv[]) {
   NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
   Employee *empl = [[Driver alloc]initWithTruck:@"Mercedes"];
   empl.salary = 5.0;  empl->age = 33;
   [empl display];
```

Java - primer koda sa enkapsulacijom, nasledjivanjem, polimorfizmom

```
public class Employee {
  private int salary;
  #this je referenca na tekuci objekat
  public Employee(int salary) { this.salary = salary;}
  public int getSalary(){ return salary;}
  public void setSalary(int newSalary) { salary = newSalary;}
  public void display() {
        System.out.println("Hello i'm employee!");
   }
  public static void main(String[] args) {
        Employee Marko = new Driver(600,"Mercedes");
        Marko.display();}
}
class Driver extends Employee {
    String truck = "FAP";
  #super vrsi poziv konstruktora bazne klase
    public Driver(int salary,String truck) {
    super(salary); this.truck = truck;}
    public void display() {
    System.out.println("My truck is "+truck+"!");
  public void display(String x) {
```

Apstrakcija

Apstraktne klase ili interfjesi

```
public abstract class Employee {
  public abstract void display(); ...
interface Employee {
  public void display(); #podrazumevano apstraktna
  default void work(){System.out.println("Working"); }
```

Tabela: Vidljivost različitih modifikatora pristupa.

Modifikator	Klasa	Paket	Podklasa	Svet
public	Da	Da	Da	Da
protected	Da	Da	Da	Ne
podrazumevani	Da	Da	Ne	Ne
private	Da	Ne	Ne	Ne

Ada

- Ada ne sledi model klase zasnovan na jednom konstruktoru
  - Odvojene karakteristike tipova unutar paketa
  - Moguće postojanje funkcija i procedura u paketu
- Enkapsulacija
  - Privatnost se određuje na nivou paketa
  - private, limited private
- Nasleđivanje
  - Implementirano nasleđivanje po nivoima i hijerarhijsko
  - Višestruko nasleđivanje je moguće implementirati
- Polimorfizam
  - Pomoću nasleđivanja, apstrkatnih tipova i podtipova
- Apstrakcija
  - Apstraktni tip se konstruiše dodavanjem abstract u deklaraciji

Ada - primer koda

```
package Employees is
  type Employee is tagged
  record
    Name: String;
  end record;
  procedure Set_Name(Obj: in out Employee; Name: String);
  function Print(E: Employee) return String;
end Employees;

package Drivers
  type Driver is new Employees.Employee with record
    Driver_ID: Integer;
  end record;
  procedure Set_D(Obj: in out Employee; Name: String);
  overriding function Print(D: Driver) return String;
end Drivers;
```

Ruby

- Kreiranje klasa u jeziku Ruby
  - Standardni metod initialize, ponaša se kao konstruktor
- Enkapsulacija
  - public, protected i private
  - attr_accessor(čitanje i izmena), attr_reader(čitanje) i attr_writer(izmena)
- Nasleđivanje
  - Implementirano nasleđivanje po nivoima i hijerarhijsko
  - Višestruko nasleđivanje nije podržano
- Polimorfizam
  - Pomoću nasleđivanja
  - Duck typing
- Apstrakcija
  - Nema direktnu podršku
  - Moguće implementiranje sličnog ponašanja pomoću nasleđivanja

Ruby - primer koda

```ruby
class Employee
  attr_accessor :name
  def initialize(name)
    @name = name
    print()
  end
  def print
    puts "Employee: #{@name}."
  end
end

class Driver < Employee
  def initialize(name)
    @name = name
    print()
  end
  private
  def print
    puts "Driver: #{@name}."
  end
end
```

Literatura

- Introduction to Ada. on-line at: https://learn.adacore.com/courses/intro-to-ada/index.html
- Object C apple documentation. on-line at: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ObjectiveC
- Ruby - Object Oriented. on-line at: https://www.tutorialspoint.com/ruby/ruby_object_oriented.htm
- Gary Bennet, Brad Lees and MItchell Fisher. Objective-C for Absolute Beginners: IPhone, iPad and Mac Programming Made Easy. Apress, Berkely, CA, USA, 3rd edition, 2016
- AdaCore experts. High-Integrity Object-Oriented Programming in Ada. AdaCore(www.adacore.com), 1.2 release edition, 2011. on-line at: http://extranet.eu.adacore.com/articles/HighIntegrityAda.pdf
- Hal Fulton. The Ruby Way. Sams Publishing, 2001.
- Cay S Horstmann. Core Java SE 9 for the Impatient. Addison-Wesley Professional, 2017.
- Aayushi Johari. Object Oriented Programming - Java OOPs Concepts With Examples, 2018. on-line at: https://www.edureka.co/blog/object-oriented-programming/
- Stephen Prata. C++ Primer Plus (5th Edition) (Primer Plus (Sams)). Sams, Indianapolis, IN, USA, 2004