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

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];
    [pool drain];
    return 0;
```

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) {
    System.out.println("My truck is "+truck+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