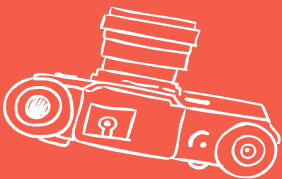
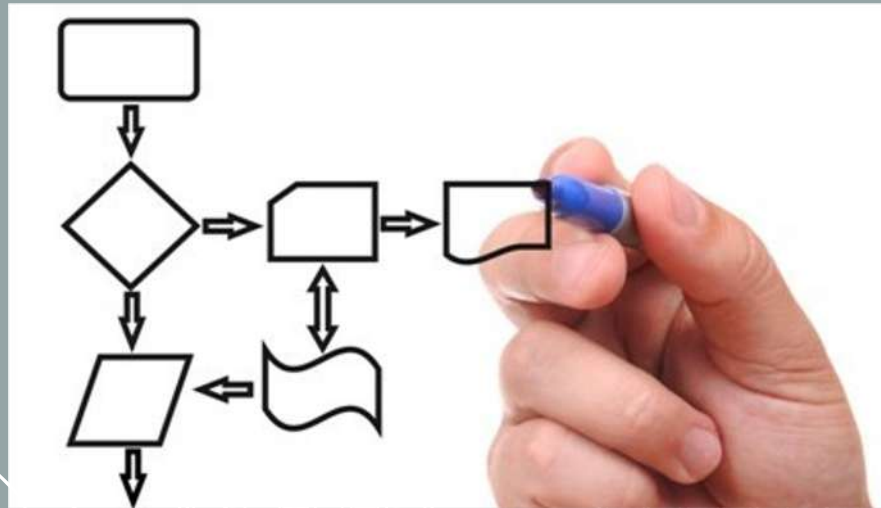


OBJEKTNO ORIJENTIRANO PROGRAMIRANJE



OBLIKOVANJE



ŠTO JE NUŽNO ZA OBLIKOVANJE ?



Definirati zahtjeve korisnika !
prije oblikovanja sustava
Nakon oblikovanja slijedi kodiranje !



OBLIKOVANJE?

✓ Nakon što je **specifikacijom utvrđeno što** softverski sustav treba raditi, **oblikovanje treba utvrditi kako će sustav raditi.**

✓ **Rezultat oblikovanja** je dizajn sustava: **precizni opis građe** sustava, **dijelova** od kojih se on sastoji, **sučelja** između dijelova, **korisničkog sučelja**, te eventualno **struktura podataka** i **algoritama** koji se koriste.

✓ Oblikovanje je **iterativni postupak**, tj. do dizajna se dolazi postupnim usavršavanjem i razradom kroz više iteracija.

GDJE SMO?

1. Najdulji put počinje prvim korakom!
2. Vi ste već trebali odmaknuti prilično daleko od kuće!
3. (Čak i prije samog kolegija Objektno orijentirano programiranje trebali bi znati nešto o objektno orijentiranim programskim jezicima. Većina vas radila je C#.)
4. JavaScript ima neke specifičnosti, ali vi učite principe OOP-a, a ne kodiranje u konkretnom programskom jeziku!



ODAKLE POČETI S OBLIKOVANJEM

Za ilustraciju pojedinih koncepata OOP-a koristit ćemo primjere iz područja računalnih simulacija koje je lako povezati s programiranjem računalnih igara.

Kontekst igara/simulacija izabran je kako bi omogućio povezivanje znanja iz različitih područja poput matematike, fizike, umjetne inteligencije, oblikovanja korisničkih sučelja, računalne grafike ...

ODAKLE POČETI S OBLIKOVANJEM

Odozdo prema gore Bottom-Up

- Fokusira se na uočene podatke
- Odigrava se u realnom vremenu
- Više je upravljani podacima
- Pogodnije za početnike

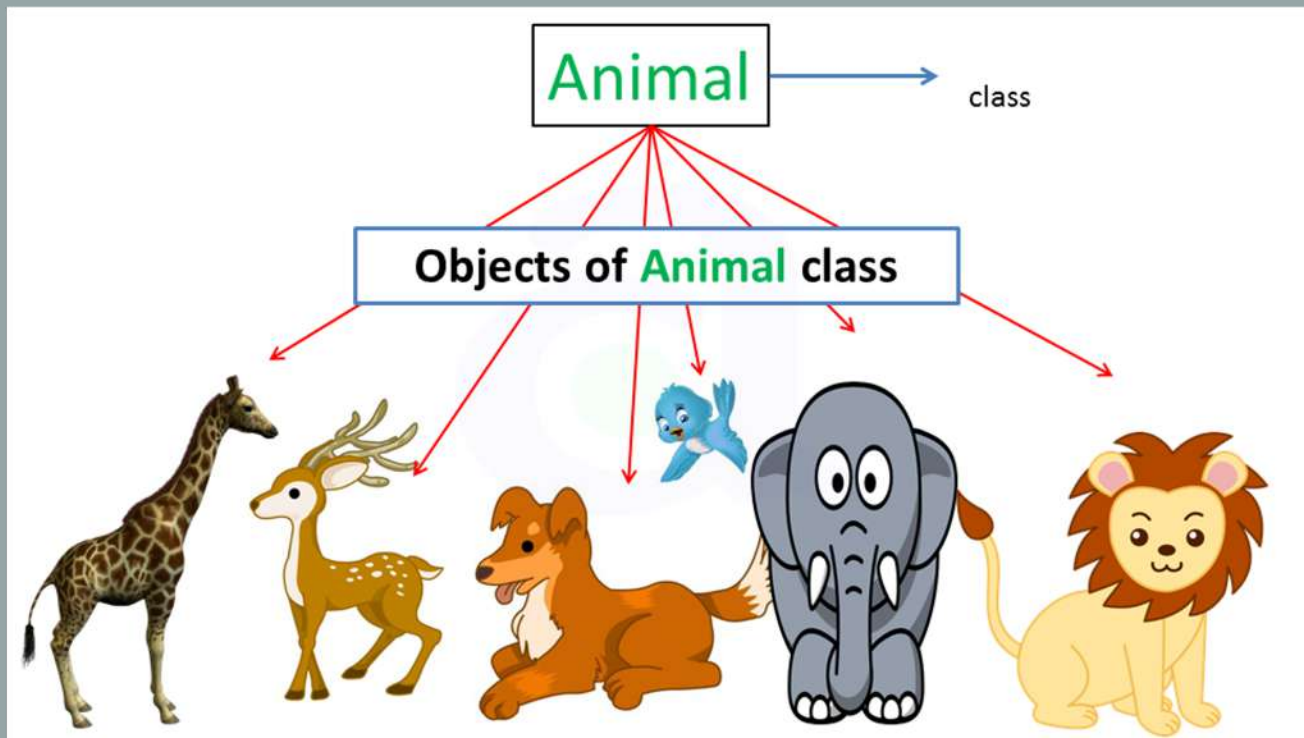


Odozgo prema dolje Top-Down

- Koristi prethodno iskustvo i očekivanja
- Informacije se interpretiraju koristeći kontekst kao vodilju
- Pogodnije za eksperte



KLASE I OBJEKTI



OBJEKT



Objekt ima stanje, ponašanje i identitet



Definicija - Objekt

Objekt ima tri obilježja: identitet, stanje i ponašanje. Identitet objekta služi za jedinstvenu identifikaciju objekta. Objekt je instanca klase čije je postojanje preduvjet za stvaranje objekta. Stanje objekta određeno je vrijednostima njegovih polja, a ponašanje je definirano metodama klase kojoj objekt pripada. □

KLASA



STVARANJE PREDLOŠKA ZA OBJEKT

Klasa predstavlja skup objekata koji dijele zajedničku strukturu i ponašanje



Definicija - Klasa

Klasa predstavlja korisnički definirani tip podatka. Klasa u C#-u započinje ključnom riječi *class* iza kojeg slijedi naziv klase i tijelo klase. Tijelo klase predstavlja blok koda omeđen vitičastim zagradama. □



STVARANJE PREDLOŠKA ZA OBJEKT

C#

```
1 | class NazivKlase
2 | {
3 |     //tijelo klase
4 | }
```

POLJA KLASKE

Definicija - Polje klase

Polje klase je varijabla deklarirana u tijelu klase.
Može biti *public* ili *private*. □


Polja klase služe za pohranu podataka za potrebe objekta.



SVOJSTVO KLASSE

Definicija - Svojstvo klase

Svojstvo klase koristi se za ograničavanje pristupa privatnim poljima klase, a sastoji se od ključne riječi **public**, naziva svojstva i tijela. Tijelo svojstva sadrži pristupne čvorove (eng. accessor nodes), a to su: **get** i **set**. Zadatak **get** dijela je dohvaćanje sadržaja privatnog polja klase, a zadatak **set** dijela je spremanje vrijednosti u privatno polje klase.□



Svojstvo je član klase koji daje pristup nekom podatkovnom polju odnosno elementu klase, te definira kako se elementi klase dohvaćaju i postavljaju.

JEDNOSTAVNA SVOJSTVA

Definicija - Jednostavna svojstva

Jednostavna svojstva su svojstva koja u tijelu sadrže pristupni čvor **get** koji u svojem tijelu ima najviše **return** naredbu i **set** koji u svojem tijelu ima najviše naredbu pridruživanja vrijednosti (ključna riječ **value**). □

C#

```
1 public tip NazivPolja
2 {
3     get {return nazivPolja;}
4     set {nazivPolja = value;}
5 }
```

SLOŽENA SVOJSTVA

Definicija - Složena svojstva

Složena svojstva za razliku od jednostavnih u tijelima pristupnih čvorova **get** i/ili **set** sadrže više naredbi, odnosno u tijelu od get pored return naredbe najmanje još jednu i/ili u tijelu set-a pored naredbe pridruživanja najmanje još jednu. □



KONSTRUKTOR KLASÉ

Definicija - Konstruktor klase

Konstruktor klase je posebna metoda klase koja za razliku od metode klase nema tip. □



KONSTRUKTOR KLASÉ

```
1  using System;
2
3  public class Lik
4  {
5      // Konstruktor koji ne prima argumente:
6      public Lik()
7      {
8          this.Ime = "Bezimeni";
9          this.Zdravlje = 100;
10     }
11
12     // Konstruktor koji prima jedan argument:
13     public Lik(string ime)
14     {
15         this.Ime = ime;
16         this.Zdravlje = 100;
17     }
```

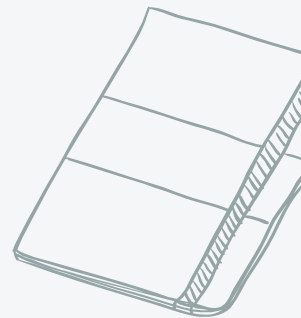




Članovi klase koji opisuju objekt odnose se na podatke, odnosno njihovo spremanje i dohvaćanje (polja i svojstva), a **članovi koji definiraju ponašanje su metode klase**

Definicija - Metoda

Metoda je imenovani blok kôda koji se sastoji od **potpisa** i **tijela** metode. **Potpis metode** sastoji se od modifikatora pristupa, povratnog tipa, naziva metode i popisa parametara u okruglim zagradama. □





private

protected

public

MODIFIKATORI PRISTUPA



MODIFIKATORI PRISTUPA



Definicija - Modifikator pristupa u tijelu klase

Modifikator pristupa kontrolira pristup varijabli ili metodi unutar klase. Može biti *public* (pristup varijabli/metodi dozvoljen je iz svih klasa), *private* (pristup varijabli/metodi dozvoljen samo unutar dane klase) i *protected* (pristup varijabli/metodi dozvoljen je unutar klase te svih klasa koje od nje nasljeđuju). □



MODIFIKATORI PRISTUPA

C#

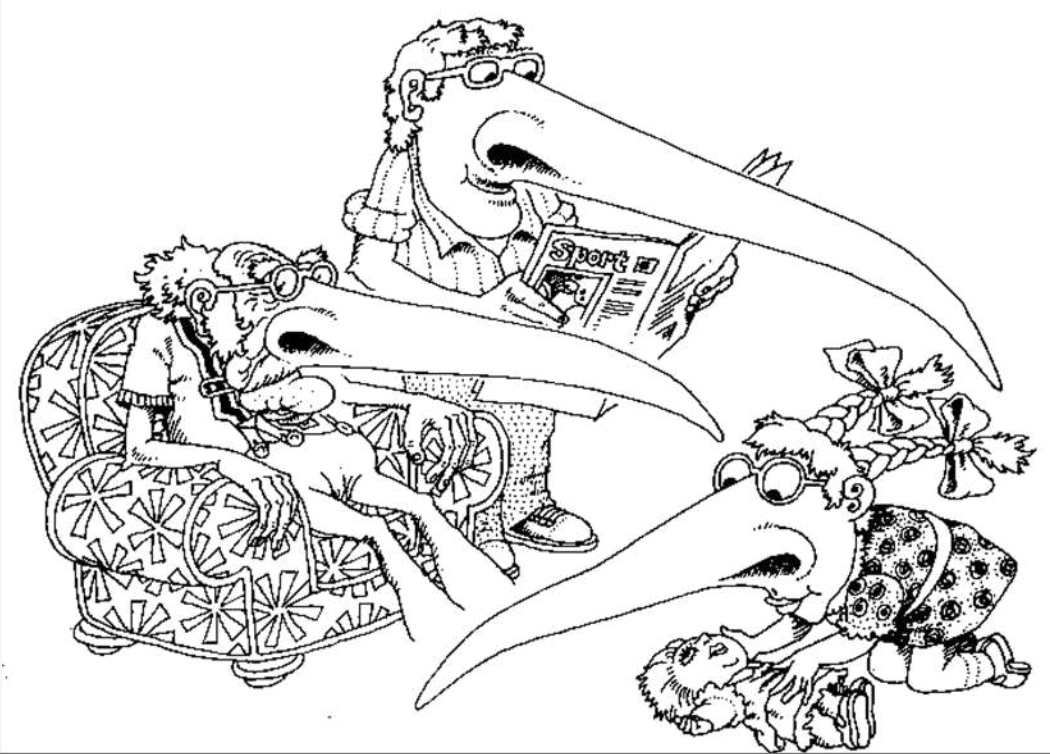
private	Član klase je vidljiv samo metodama koje su članovi iste klase.
protected	Članovi su vidljivi metodama klase kojoj pripadaju i metodama iz nje izvedenih klasa.
public	Članu klase mogu pristupati metode svih klasa.
internal	Dostupnost svim metodama koje pripadaju klasama iz istog asemblja.
protected internal	Dostupnost klasama iz istog asemblja i izvedenim klasama.

MODIFIKATORI PRISTUPA

JAVASCRIPT

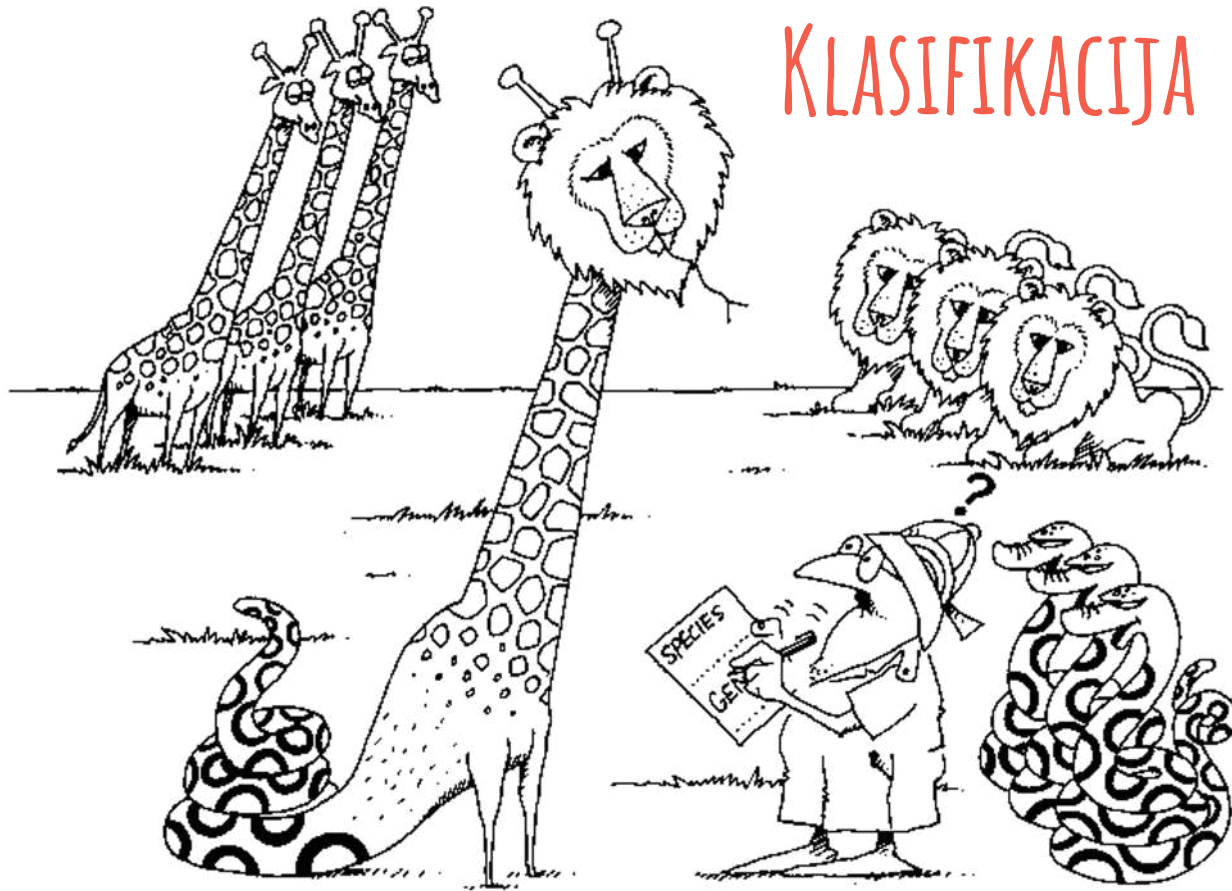
private	Član klase je vidljiv samo metodama koje su članovi iste klase.
public	Članu klase mogu pristupati metode svih klasa.

NASLJEĐIVANJE



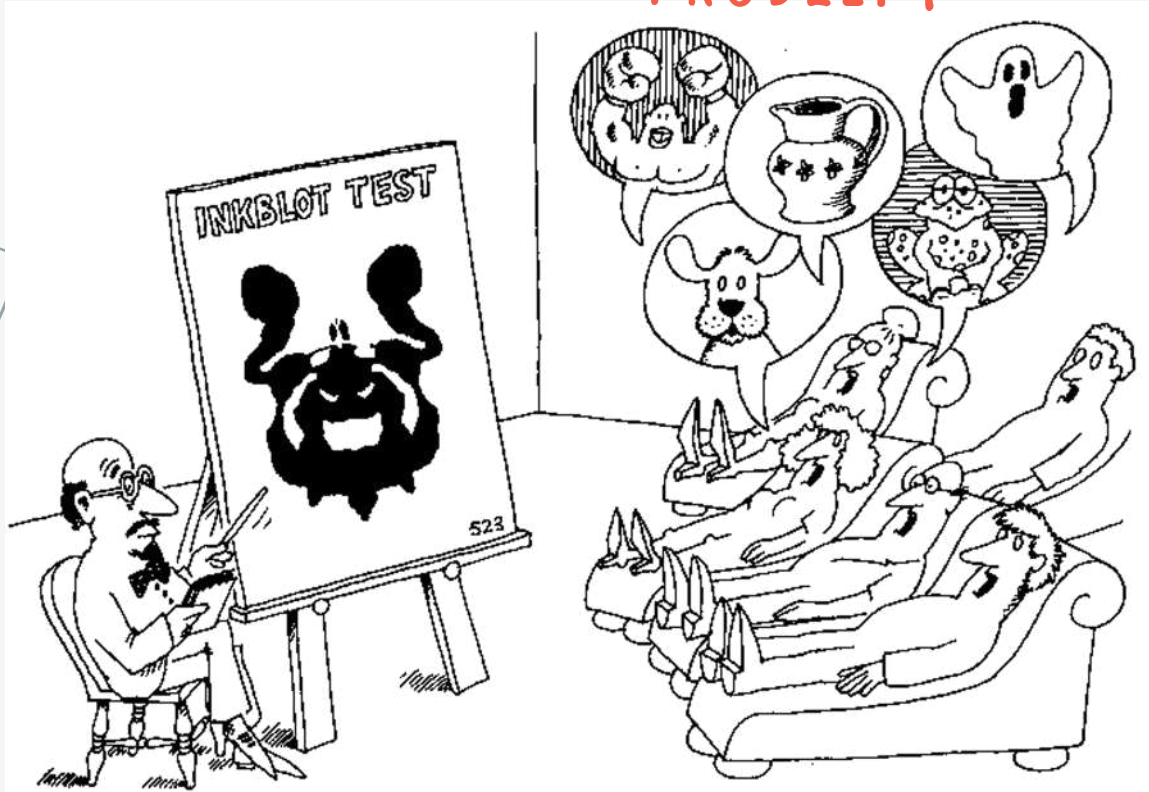
Izvedena klasa obično nasljeđuje strukturu
i ponašanje bazne

KLASIFIKACIJA



Klasifikacija je proces kojim uređujemo znanje

PROBLEM



Veliki problem: različiti promatrači će klasificirati isti objekt na različite načine



RAD U PARU

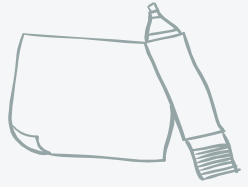
Popularan, ali ipak svatko od vas treba svoje računalo.
Suradujte, ali nemojte samo sjediti i gledati.

ŠTO MORATE ZNATI?

- Programirati (kodirati) u C#
- **Možete pisati užasne programe u objektno orijentiranom jeziku s jednakom lakoćom kao i u proceduralnom jeziku**
- Da bi pisali složene programe morate biti u stanju raditi u timu
- **Alati** koje koriste **loši programeri pomažu im brže pisati loš kod**



ŠTO PRIJE NEGO ZAPOČNETE?

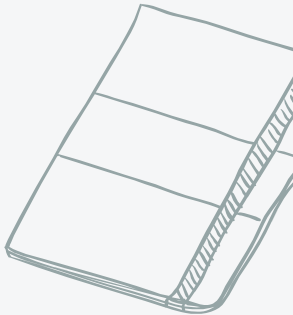


Odaberite temu, programski jezik smo odavrali mi. Koristimo **C#**

Odaberite radno okruženje,

Isplanirajte **vrijeme** kako bi se svaki tjedan mogli posvetiti zadacima

Kreirajte vlastite **bilješke** (rado ćemo ih objaviti na stranicama kolegija)



ČETIRI STUPA OBJEKTNO ORIJENTIRANOG PROGRAMIRANJA

ENKAPSULACIJA

NASLJEĐIVANJE

APSTRAKCIJA

POLIMORFIZAM



IDUĆI PUT!

Prisjetite se što trebate napraviti i koje smo osnovne koncepte spomenuli !