CENTAR ZA
mlade talente

Primena OOP u izradi složenih aplikacija (programski jezik C#)

Uvod u C# (3)

Strukture

'Value' tipovi

Preporuka za tipove manjih veličina (do 16 bajta)

```
struct ImeStrukture { ... }
```

Polja moraju da se inicijalizuju ili:

- u telu konstruktora
- u inicijalizator objekata, podrazumevani konstruktor se implicitno poziva,
- ručno nakon što se objekat strukture kreira

Samo se konstantna i statička polja mogu inicijalizovati u deklaraciji

Ograničenja:

- Podrazumevani konstruktor se ne može eksplicitno definisati
- Ne može da nasledi baznu klasu niti da bude nasleđena, ali može da implementira proizvoljan broj interfejsa

Klase

Referentni tipovi

```
class Imeklase { ... }
```

Polja se mogu inicijalizovati na sledeće načine:

- U telu konstruktora, ako konstruktor nije definisan, kompajler generiše podrazumevani konstruktor.
- U inicijalizator objekata, podrazumevani konstruktor se implicitno poziva, pa mora biti dostupan ako je definisan.
- U deklaraciji klase.

Ključna reč **this** se koristi za pristupanje trenutnoj klasi (strukturi), takođe se koristi za pristupanje njihovim članovima (konstruktori, metode, svojstva)

Može da nasledi jednu baznu klasu i da implementira više interfejsa

Modifikatori pristupa

- **Public** – označava dostupnost unutar i van tipa ili biblioteke (eng. *Assembly*) gde je definisano, može se primeniti na klase, strukture i njihove članove.
- **Internal** – označava dostupnost unutar iste biblioteke, može se primeniti na klase, strukture i njihove članove, podrazumevana vrednost za klase i strukture ako se ne navede.
- **Protected** – označava dostupnost člana klase unutar same klase ili iz nasleđenih klasa.
- **Protected Internal** – označava dostupnost člana klase unutar iste biblioteke ili iz nasleđenih klasa.
- **Private** – označava dostupnost unutar tipa gde je definisan, primenjuje se na članove klasa/struktura i ugnježdenih klasa/struktura, podrazumevana vrednost za članove klasa i struktura ako se ne navede.

Polja

Polja su promenljive unutar klase ili strukture, tj. članovi klase ili strukture

Vrednosti **polja** deklarisanih kao **const** ili **readonly** se ne mogu promeniti nakon inicijalizacije

const članovi

- inicijalizuju se jednom kada se deklarišu
- ne mogu biti s leve strane operatora dodele nakon inicijalizacije
- implicitno statički
- samo se primitivni tipovi mogu deklarisati kao const
- lokalne promenljive se mogu deklarisati kao const

readonly članovi

- mogu se inicijalizovati kada se deklarišu ili najkasnije u konstruktoru
- primitivni i korisnički definisani tipovi se mogu deklarisati kao readonly
- lokalne promenljive se ne mogu deklarisati kao readonly

Svojstva

- **Svojstva** predstavljaju šablon gde se metode (*getter* i *setter*) koriste za pristupanje članovima podataka klasa, struktura ili interfejsa
- Simuliraju *get* (čitanje vrednost) i *set* (postavljanje vrednost) metode
- Mogu biti *read/write* (*getter* i *setter*), *read-only* (*getter*) ili *write-only* (*setter*)
- Preko ključne reči **value** se u telu *setter*-a pristupa prosleđenoj vrednost za postavljanje vrednosti odgovorajućeg polja
- Autoimplementirana svojstva se mogu koristiti ako su *getter*-i i *setter*-i trivijalni

```
public int ImeSvojstva
{
   get { ... }
   set { ... = value }
}
```

Metode

Metode se moraju deklarisati unutar klase, strukture i interfejsa

Metoda se deklariše na sledeći način:

- prvo se deklarišu **atributi** (opciono)
- zatim slede njeni **modifikatori** (opciono)
- posle se obavezno navodi tip povratne vrednost, ako metoda ne vraća vrednost, **void** se navodi kao tip povratne vrednost
- **ime** metode se nakon toga specificira
- parametri se na kraju mogu navesti u zagradi tako što se deklariše **modifikator**, **tip** i **ime** za svaki parametar, parametri se razdvajaju zarezima, ako metoda nema parametara, zagrade su prazne

return se koristi za izlazak iz metode i za vraćanje rezultata, ako je povratna vrednost metode u deklaraciji **void**, **return** može da se izostavi

Metode - parametri

Ulazni parametar

- nijedan modifikator ispred formalnog i stvarnog parametra
- prosleđuje se po vrednosti pa se kopija parametra prosleđuje u metodu, stvaran parametar mora da se inicijalizuje pre prosleđivanja u metodu
- ako je parametar ’value’ tip, izmene u metodi ne utiču na vrednost stvarnog parametra koji je prosleđen tj. izmene u metodi nisu vidljive van metode
- ako je parametar referentni tip, izmene unutar metode utiču na vrednost stvarnog parametra koji je prosleđen tj. izmene u metodi su vidljive van metode

Ulazno-izlazni parametar (ref)

- **ref** modifikator ispred formalnog i stvarnog parametra
- **referenca** stvarnog parametra se zapravo prosleđuje u metodu, stvaran parametar mora da se inicijalizuje pre prosleđivanja u metodu
- izmene unutar metode utiču na vrednost stvarnog parametra koji je prosleđen tj. izmene u metodi su vidljive van metode

Metode - parametri

Izlazni parametar (out)

- **out** modifikator ispred formalnog i stvarnog parametra
- **referenca** stvarnog parametra se zapravo prosleđuje u metodu
- stvarnom parametru se mora dodeliti vrednost u metodi pre izlaska iz nje

Proizvoljan broj ulaznih parametara (params)

- omogućava deklaraciju metode koja ne prima nijedan ili prima više ulaznih parametara istog tipa
- **params** modifikator ispred formalnih i stvarnih parametara, nije dozvoljeno deklarisanje dodatnih formalnih parametara iza **params,** samo jedan **params** modifikator je dozvoljen u deklaraciji metode
- posledice izmena parametara unutar metode se poklapaju sa ponašanjem kada su ulazni parametri (prosleđivanje po vrednosti)
- stvarni parametri se mogu proslediti kao pojedinačni parametri razdvojeni zarezima ili kao niz parametara
- unutar metode su parametri dostupni kao niz

CENTAR ZA
mlade talente

Hvala na pažnji!