CENTAR ZA
mlade talente

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

Osnovni koncepti u C#

Osnovni koncepti objektno orijentisanog programiranja

1) Enkapsulacija
   - mogućnost programskog jezika koja omogućava grupisanje i zaštitu srodnih podataka (klase)
2) Nasleđivanje
   - Mogućnost da se na osnovu postojećih klasa izvedu nove klase koje treba da prošire, iskoriste ili ismene ponašanje definisano u postojećim klasama
3) Polimorfizam
   - Zasniva se na ideji da metoda, deklarisana u osnovnoj klasi, može biti implementirana na više različitih načina u različitim izvedenim klasama. Polimorfizam se realizuje preko virtuelnih metoda

Nasleđivanje

Omogućava kreiranje nove (nasleđene/izvedene) klase koja koristi, proširuje i modifikuje ponašanje definisano u nekoj drugoj (osnovnoj) klasi

```
public class NasleđenaKlasa : OsnovnaKlasa
{
    . . .
}
```

Višestruko nasleđivanje **nije podržano!!!**

Pristupanje članovima osnovne klase

Ključna reč **base** se koristi za pristup članovima osnovne klase iz nasleđene klase

Može se koristiti za pozivanje određenog konstruktora osnovne klase u konstruktoru nasleđene klase

```
public NasleđenaKlasa() : base()
{
    . . .
}
```

Može se takođe koristiti za pozivanje metode osnovne klase koja je reimplementirana u nasleđenoj klasi

```
public override void MetodaKlase()
{
    base.MetodaKlase();
    . . .
}
```

Static modifikator

- **static** modifikator se može primeniti na klase/strukture i na njihove članove.
- Klasa/struktura deklarisana **static** modifikatorom sadrži samo statičke članove.
- Nije moguće instancirati objekat statičke klase/strukture.
- Ako se **static** modifikator primeni na člana, to znači da taj član pripada samoj klasi/strukturi, a ne instanci klase/strukture.
- Statičkim članovima se pristupa preko klase/strukture, a ne preko instance.
- **static** modifikator se može primeniti na polja, metode, svojstva, operatore, događaje i na konstruktore, ali ne i na destruktore.

Polimorfizam

- **virtual** modifikator se primenjuje na članove (metode, svojstva, događaje) u osnovnoj klasi i omogućava redefinisanje (reimplementaciju) tih članova u nasleđenoj klasi;
- **override** modifikator se primenjuje na članove izvedene klase koji su nasleđeni iz osnovne klase, i omogućava njihovu redefiniciju i modifikaciju;
- **sealed** modifikator se može primeniti na klase da bi se onemogućilo njihovo nasleđivanje, ili na redefinisane članove nasleđenih klasa da bi se zabranilo (zaustavilo) njihovo dalje redefinisanje;
- **new** modifikator, koji se primenjuje na članove nasleđenih klasa, eksplicitno sakriva odgovarajuće članove (sa istim imenima i potpisima) koji su nasleđeni iz osnovne klase, ako se izostavi, kompajler generiše upozorenje.

Apstraktne klase

abstract modifikator se može primeniti na klase i na njihove članove da se naznači da su te klase apstraktne

Ako se primeni na klasu, bar jedan od njenih članova je apstraktan

- Ne mogu se kreirati instance apstraktne klase
- Namenjena je da bude osnovna klasa
- Čista apstraktna klasa je klasa koja sadrži isključivo apstraktne članove

```
public abstract class ImeKlase { ... }
```

Može da se primeni na metode, svojstva i događaje da bi se naznačilo da su ti članovi apstraktni i da nemaju implementaciju

- Moraju se implementirati u nasleđenim klasama

```
public abstract void MetodaApstraktneKlase();
```

Object klase

Osnovna klasa za sve klase i tipove podataka, uključujući i ’value’ tipove

Definiše *public* i *protected* metode koje su implicitno nasleđene od strane svih tipova podataka

Većina tih metoda se mogu redefinisati

Neke od metoda definisane u klasi su:

- **ToString** – vraća string reprezentaciju trenutne vrednosti objekta
- **Equals** – proverava da li su dva objekta jednaka

Interfejsi

Čisto apstraktni referentni tipovi koji definišu šta klase/strukture treba da rade, a ne kako to da implementiraju, njihova imena imaju prefiks **I** (po konvenciji)

Sadrže samo apstraktne članove (bez implementacije), tj. mogu da sadrže potpise metoda, svojstava, delegata i događaja

Ne mogu da sadrže polja, konstante, operatore, konstruktore ili destruktore

S obzirom da C# ne podržava višestruko nasleđivanje, interfejsi služe kao alternativa za takvu funkcionalnost

```
interface IImeInterfejsa
{
    . . .
}
```

Razlike između apstraktnih klasa i interfejsa

- Klase i strukture implementiraju interfejse, dok samo klase nasleđuju apstraktne klase
- Klasa može da implementira proizvoljan broj interfejsa, ali može da nasledi samo jednu apstraktnu klasu
- Struktura može da implementira proizvoljan broj interfejsa, ali ne može da nasledi nijednu apstraktnu klasu
- Interfejs ne sadrži ni jednu implementaciju nego samo deklaracije, dok apstraktna klasa može, ali i ne mora, da sadrži implementacije
- Članovi interfejsa su uvek javni dok se bilo koji modifikator pristupa može primeniti na članove apstraktne klase

Primeri ugrađenih interfejsa

IComparable – tip koji implementira ovaj interfejs implementira metodu **CompareTo** koja vrši upoređivanje

IEquatable – tip koji implementira ovaj interfejs implementira metodu **Equals** koja proverava jednakosti

IEnumerable – tip koji implementira ovaj interfejs implementira metodu **GetEnumerator**, koja vraća **IEnumerator** interfejs, i podržava iteraciju kroz kolekcije

IDisposable – tip koji implementira ovaj interfejs implementira metodu **Dispose**, koja služi da oslobodi zauzete resurse

CENTAR ZA
mlade talente

Hvala na pažnji!