NAPREDNO I OBJEKTNO ORIJENTIRANO PROGRAMIRANJE

Objektno orijentirani jezici

Kod proceduralnih jezika podaci su pasivan element.

OO jezici imaju i proceduralne i neproceduralne elemente, procedure i podaci povezani su u smislenu cjelinu - OBJEKT.

Svaki objekt ima svoja unutrašnja stanja i operacije, analogno objektima u realnom svijetu.

Dobro oblikovani objekti vjerna su preslika stvarnih, logički su zatvorene i međusobno nezavisne programske jedinice koje se lako mogu uklopiti u funkcionalnu cjelinu – aplikaciju.

Aplikaciju (program) čini skup objekata koji između sebe i s vanjskim svijetom komuniciraju putem „poruka” (upotrebom svojstava, metoda i događaja).

Usporedba

Proceduralno	Objektno-orijentirano
Definiraj varijablu ime Definiraj varijablu prezime Učitaj ime Učitaj prezime Pozdravi(ime, prezime) ...	Kreiraj objekt Učenik Učenik.Učitaj(ime) Učenik.Učitaj(prezime) Učenik.Pozdravi() ...

Osnovna načela OOP-a

Programer apstrakcijom radi MODEL objekta iz stvarnog svijeta, koristi samo bitne detalje.

Objekti iste vrste posjeduju iste osobine ili svojstva (statičke) i iste postupke, operacije ili metode (dinamičke). Podaci (statičke osobine) i operacije (dinamičke osobine) sastavni su dio objekta.

Apstrakcija je proces u kojem na temelju promatranja pojedinačnih predstavnika neke vrste iz stvarnog svijeta gradimo programski model koji ih opisuje – KLASU. Ona nam služi za stvaranje (instanciranje) objekata.

Instanciranom objektu klasa određuje SUČELJE (interface) koje uključuje njegova svojstva i metode, te događaje.

Osnovni OOP koncepti su kontrola pristupa, učahurivanje, nasljeđivanje i polimorfizam.

Kontrola pristupa članovima sučelja objekta

Za svaki član klase moramo definirati razinu pristupa (dostupnosti).

Osnovne razine pristupa su private i public.

Privatni članovi klase dostupni su samo metodama te klase i to je najviša razina zaštite.

Javnim članovima klase može pristupiti bilo koji vanjski objekt.

Detalji implementacije članova sučelja objekta (metoda, svojstava, događaja) skriveni su unutar klase, pa korisnik instanciranog objekta preko sučelja vidi samo članove koji su u klasi definirani kao javno dostupni, dok mu privatni članovi ostaju skriveni (data hiding).

Tako se klasa brine o očuvanju integriteta i funkcionalnosti svakog objekta kao zatvorene programske cjeline.

Učahurivanje (enkapsulacija)

Objekt uključuje unutrašnja stanja i metode i omogućuje pristup samo svojim javno dostupnim članovima (svojstvima, metodama i događajima).

Nasljeđivanje

Posljedica generalizacije (suprotno od specijalizacije).

Npr. Klasa Učenik, Klasa Nastavnik, Klasa Osoba.

Klase nastale specijalizacijom nasljeđuju sva svojstva i metode polazne ili BAZNE klase.

Polimorfizam

Ista metoda implementirana je i djeluje na različite načine, ovisno o objektu na koji se primjenjuje. Mijenja se npr. implementacija metode naslijeđene iz bazne klase.

Upotreba OO pristupa omogućuje modeliranje stvarnog sustava na visokoj razini apstrakcije.

Razvojna platforma .NET

.NET Framework je softverski okvir.

Uključuje veliku biblioteku klasa (.NET Class Library) i podržava nekoliko programskih jezika koji se mogu koristiti istodobno.

Programi se izvršavaju u višejezičnom softverskom okruženju CLR (Common Language Runtime).

Ta višejezična platforma je oblik aplikacijske virtualne mašine koja .NET programima pruža bitne servise kao što su sigurnost, upravljanje memorijom i obrada grešaka.

.NET biblioteka klasa, .NET programski jezici, programski jezik C#

Objektni model klasa, .NET Class Library (npr. klasa Form, klasa Button, klase za rad s datotekama i datotečnim sustavom).

Klase .NET biblioteke neovisne su funkcionalne cjeline koje se mogu upotrebljavati iz bilo kojeg od ponuđenih programskih jezika (VB, VC++, VC#...) koji omogućuju i stvaranje novih tipova – klasa (uključujući metode, svojstva i događaje).

Instaliranje, razvojna okolina Visual Studio, pisanje programa

Projektno rješenje koje se sastoji od jednog ili više projekata. Kod kreiranja koristi se predložak.

Console Application.

AssemblyInfo.cs sasrži općenite informacije (naziv, opis, tvrtka, autor, verzija..) i reference do standardnih biblioteka. U program.cs je programski kod. Reference su poveznice prema već prevedenim dijelovima.

Metoda Main

Polazna točka izvršavanja programa.

```
Console.WriteLine(„---- „);
Console.ReadKey();
```

Pomoćni alati: IntelliSense, prikaz detalja o elementu, pomoć pri određivanju parametara

Intellisense: konteksni izbornik za automatsko upisivanje teksta (imenski prostor, klasa, svojstvo, metoda ili događaj... označava se odgovarajućom ikonom)

Prikaz detalja: kratki opis u pomoćnom okviru kad se približimo miš.

Pomoć pri određivanju parametara u pozivu metode: pomoćni okvir kod otvaranje zagrade, pomaže kod preopterećenih metoda.

Error list prikazuje poruke o greškama.

.NET objektni model

Biblioteke klasa organizirane su u više datoteka (.NET sklopovi ili assemblies). Svaki assembly ima definirani logički smisleni imenski prostori (namespaces) koji sadrže podprostore sa klasama.

Referenciramo samo biblioteke čijim se klasama namjeravamo koristiti. Osnovnu biblioteku mscorlib.dll nije potrebno eksplicitno referencirati. Uključuje imenske prostore i klase za čitanje i pisanje datoteka, grafičko iscrtavanje, interakciju s bazom podataka i manipulaciju XML dokumentima, sadrži imenski prostor System i klasu Console.

Klase, imenski prostori i naredba using

Kad u projektu imamo referencu do nekog sklopa (assembly) moramo znati i imenski prostor u koji je klasa svrstana.

Za kraće pisanje koristimo using koji uključuje imenski prostor.

Operator . omogućuje kretanje kroz hijerarhiju objektnog modela: imenski prostor.klasa.metoda.

Osnovna pravila pisanja koda

Razlikuju se velika i mala slova.

Potrebno je pisati stroge deklaracije i inicijalizacije varijabli.

Treba poštivati pravila imenovanja, preporučuje se upotreba engleske abecede.

Program treba pisati prema pravilima hijerarhije .NET objektnog modela: imenski prostor je najviša razina organiziranja klasa na logički smislen način. Imenski prostor definira se pomoću ključne riječi namespace i imena.

Klase se definiraju unutar imenskog prostora.

Unutar klase definiraju se svojstva, metode i događaji. Ključna riječ static određuje da se radi o metodi koja se poziva preko imena klase. Postoje i metode instance koje se pozivaju preko instanciranog objekta. U zagradama se navode parametri metode.

Naredbe uglavnom završavaju sa ;. Komentari se pišu kao u C-u.

Osnove programiranja – tipovi podataka

Tip podataka određuje način spremanja podataka u memoriji, skup mogućih vrijednosti i skup mogućih akcija.

Dijele se na ugrađene (intrinsic, dio jezika) i korisnički definirane.

Prema načinu spremanja vrijednosti dijele se na **vrijednosne (vrijednost čuvaju na stogu, jednostavni tipovi, enumeracije i strukture)** i **referentne (vrijednost čuvaju na hrpi, objekti)**.

Stog (stack) je područje radne memorije unaprijed rezervirano kod pokretanja programa.

Hrpa (heap) je područje momorije odvojeno od bloka u kojem se izvršava program. Prostor na hrpi ne dodjeljuje se unaprijed već dinamički u trenutku instanciranja objekta.

Ugrađeni tipovi

Type	Range	Size
sbyte	-128 to 127	Signed 8-bit integer
byte	0 to 255	Unsigned 8-bit integer
char	U+0000 to U+ffff	Unicode 16-bit character
short	-32,768 to 32,767	Signed 16-bit integer
ushort	0 to 65,535	Unsigned 16-bit integer
int	-2,147,483,648 to 2,147,483,647	Signed 32-bit integer
uint	0 to 4,294,967,295	Unsigned 32-bit integer
long	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Signed 64-bit integer
ulong	0 to 18,446,744,073,709,551,615	Unsigned 64-bit integer

Type	Approximate range	Precision
float	±1.5e−45 to ±3.4e38	7 digits
double	±5.0e−324 to ±1.7e308	15-16 digits

Type	Approximate Range	Precision	.NET Framework type
decimal	$(-7.9 \times 10^{28}$ to $7.9 \times 10^{28}) / (10^{0}$ to $28)$	28-29 significant digits	System.Decimal

bool	**System.Boolean**
byte	**System.Byte**
sbyte	**System.SByte**
char	**System.Char**
decimal	**System.Decimal**
double	**System.Double**
float	**System.Single**
int	**System.Int32**
uint	**System.UInt32**
long	**System.Int64**
ulong	**System.UInt64**
object	**System.Object**
short	**System.Int16**
ushort	**System.UInt16**
string	**System.String**

bool	Boolean	
byte	Unsigned, numeric, integral	
char	Unsigned, numeric, integral	
decimal	Numeric, decimal	M or m
double	Numeric, floating-point	D or d
enum	Enumeration	
float	Numeric, floating-point	F or f
int	Signed, numeric, integral	
long	Signed, numeric, integral	L or l
sbyte	Signed, numeric, integral	
short	Signed, numeric, integral	
struct	User-defined structure	
uint	Unsigned, numeric, integral	U or u
ulong	Unsigned, numeric, integral	UL or ul
ushort	Unsigned, numeric, integral	

https://msdn.microsoft.com/en-us/library/exx3b86w%28v=vs.110%2

Tipovi int i double (predodređeni brojčani), char i string

char: jednostavni (‘A’), unicode (‘\u0041’) i kontrolni (‘\t’).

String (referentni, memorija se dodjeljuje dinamički) je niz pojedinačnih znakova tipa char („A”).

Kontrolni znakovi:
\' - single quote, needed for character literals
\" - double quote, needed for string literals
\\ - backslash
\0 - Unicode character 0
\a - Alert (character 7)
\b - Backspace (character 8)
\f - Form feed (character 12)
\n - New line (character 10)
\r - Carriage return (character 13)
\t - Horizontal tab (character 9)
\v - Vertical quote (character 11)

Varijable, deklaracija i inicijalizacija

Varijabla je naziv (identifikator) memorijske lokacije u kojoj program čuva vrijednost.

Sve varijable prije upotrebe moraju biti strogo (eksplicitno) deklarirane navođenjem **tipa** i inicijalizirane na početne vrijednosti, a **nazivi** moraju zadovoljavati pravila imenovanja (alfanumerički znakovi i _).

Vrijednost varijabli dodjeljujemo pomoću operatora dodjele =. Početno dodjeljivanje vrijednosti varijabli naziva se inicijalizacija.

Varijable vrijednosnog tipa i varijable referentnog tipa

Sve varijable jednostavnih ugrađenih tipova su vrijednosnog tipa, odnosno vrijednost im je sadržana u varijabli koja se sprema na stog u istom memorijskom bloku s programom.

Objekti su referentni tipovi koji se spreamju na hrpi u odvojenom memorijskom bloku koji se dodjeljuje objektu u trenutku njegovog instanciranja. Varijabla pomoću koje dohvaćamo objekt ne sadrži zato izravno njegovu vrijednost već samo referencu (pokazivač) do memorijskog bloka u kojem je objekt spremljen. Zato se varijable referentnog tipa definiraju najčešće upotrebom operatora new: `ImeKlase naziv=new ImeKlase();`

`ImeKlase` je ime referentnog tipa nakon kojeg slijedi naziv varijable – deklaracija. Inicijalizacija je provedena pozivom operatora `new` nakon kojeg slijedi `ImeKlase` pozvano kao metoda. Na taj se način instancira novi objekt tipa `ImeKlase`. Varijabla `naziv` sadrži tako referencu do bloka na hrpi gdje je objekt spremljen.

Konstante

Točno određena nepromjenjiva vrijednost u programu: literali (doslovno navedene vrijednosti), simboličke konstante (sa imenima: `const tip naziv=vrijednost;`) i enumeracije.

Enumeracija je vrijednosni tip sastavljen od popisa imenovanih brojčanih konstanti. Definira se:

enum PSP

{AK=16, BK=18, VS=12, BPO=21}

Dohvaćamo preko PSP.AK.

Enumeraciju ne možemo definirati unutar metode već kao člana klase ili člana imenskog prostora.

Pretvaranje tipova

Podatak jednog tipa može se **eksplicitno** ili **implicitno** pretvoriti u podatak nekog drugog tipa.

Implicitne pretvorbe izvršavaju se automatski i u njima se informacije ne mogu izgubiti (npr. tip `int` u `double`).

Eksplicitna pretvorba izvodi se preko operatora pretvorbe (**cast operator**),
npr. `int x=(int)y;`

Izrazi i operatori

Niz varijabli i konstanti međusobno povezanih operatorima (i odvojenih zagradama) koji kao rezultat daje novu vrijednost nazivamo **izraz**.

Prema broju operanada ili izraza koje povezuju operatori mogu biti unarni, binarni ili ternarni.

Prema vrsti operatori mogu biti aritmetički (+, -, *, /, %, ++, --), relacijski (usporedba dviju vrijednosti, vraćaju `true` ili `false`, nižeg prioriteta od aritmetičkih, ==, !=, <=, >=, <, >) i logički operatori (&&, ||, !), operator spajanja teksta (konkatenacija, +, ako je bar jedan operand tipa string rezultat je tipa string) te operatori dodjele (=,za kraće pisanje, +=, -=, *=, /=) i operator pretvorbe ().

Operatori imaju prioritete.

U slučaju nepodudaranja tipova, ako je moguća, provodi se implicitna konverzija.

Naredbe za kontrolu izvođenja programa u programskom jeziku C#

Osnovna pravila:

;

Blok naredbi - skup naredbi smješten unutar vitičastih zagrada

Svako dodjeljivanje je izraz

Category	C# keywords
Selection statements	if, else, switch, case
Iteration statements	do, for, foreach, in, while
Jump statements	break, continue, default, goto, return, yield
Exception handling statements	throw, try-catch, try-finally, try-catch-finally

Bezuvjetno grananje, metode

Bezuvjetno grananje može se pozvati pozivanjem metode ili pomoću ključne riječi (goto, break, continue, return ili throw)

```
static void Main(...)
{...
NovaMetoda();
...}
static void NovaMetoda()
{
...
}
```

Naredba uvjetnog grananja: if-else

Primjer:

Napraviti program koji ispisuje kakav je unijeti broj (manji, veći ili jednak nuli).

Za pretvorbu koristiti int a=int.Parse(Console.ReadLine());

https://msdn.microsoft.com/en-us/library/xt4z8b0f.aspx

Upravljanje greškama: try..catch

```
...
try
    {
       int a=int.Parse(Console.ReadLine());
       Console.WriteLine(„Unijeli ste broj: {0}”,a);
    }
catch (Exception ex)
    {
       Console.WriteLine(„Greska: {0} ", ex.Message);
    }
...
```

Često se koristi za osiguravanje od grešaka kod parsiranja korisničkog unosa.

Klasa System.Exception učahuruje svojstva i metode pomoću kojih .NET razvojna okolina opisuje sve vrst grešaka koje se mogu dogoditi izvršavanjem programa. Osim na ekran, detalje o grešci obično se ispisuje u datoteku ili pošalje administratoru e-mailom, te se zatraži i ponovni unos.

Iteracijske naredbe (petlje): while, for

Niz (polje, array), deklaracija, inicijalizacija, pristupanje

Za pohranjivanje više podataka (varijabli) istog tipa međusobno logički povezanih (pristup pojedinom podatku je preko cjelobrojnog indeksa).

int[] brojevi = new int[3] {1, 2, 3}; //istovremeno se izvodi i inicijalizacija

Pridruživanje:

Brojevi[1] = 2; //početni indeks je nula

Za veličinu niza koristi se svojstvo brojevi.Length

(Primjer. Učitavanje i ispis.)

String varijabla (složeni znakovni tip string) implementiran je kao niz tipa char.

(Primjer. Ispis pojedinih slova niza.)

Otkrivanje i ispravljanje grešaka

Za greške logičkog tipa (daju neispravne rezultate).

Korisni alati i tehnike kod pokretanja programa u debug načinu upotrebom točki prekida (breakpoints) izvršavanja i kontekstnog preglednika stanja varijabli te analizom tijekom izvršavanja programa liniju po liniju.

Točka prekida postavlja se tako da na odgovarajućoj liniji (npr. pozivu metode) iz izbornika Debug odabereno opciju Toggle Breakpoint (F9), zatim pokrenemo sa Start Debugging (F5). Izvršavanje programa privremeno se zaustavlja.

Kontekstni preglednik stanja prikazuje nam stanja (vrijednosti) pojedinih varijabli dok je program privremeno zaustavljen u debug načinu (zadržimo se pokazivačem miša na imenu neke varijable).

Ako zadržimo pokazivač iznad neke klase, moguće je pregledati čitavu hijerarhiju njenih svojstava.

Program privremeno zaustavljen u nekoj točki prekida možemo nastaviti izvršavati liniju po liniju kontrolirajući pritom stanje pojedinih varijabli (Step Into, Step Over, Step Out).

Vrijednosti varijabli možemo pratiti i pomoću kontekstnog izbornika Locals, prikazuju se sve lokalne varijable iz bloka kojem pripada linija na kojoj je točka prekida, varijablu kroz više blokova pratimo sa Add watch.

Za prikaz stanja stoga služi okvir Call Stack.

Osnove naprednog programiranja