2.Uvod u C#

**1)Tipovi podataka**

* Vrednosni tipovi
  + instance se skladište u stek memoriji(enumi,struktura(kao klasa samo je vrednosni tip ))
* Adresni tipovi(klase,interfejsi(Interfejs se ne nasledjujeee vec imnplementira ),nizovi,delegate)
  + adrese instanci se skladište u stek memoriji
  + instance se skladište u heap memoriji
* Garbage Collection
  + oslobađanje memorije koju zauzimaju nekorišćeni objekti
  + mehanizam je pod kontrolom CLR i automatski se aktivira

Automatsko određivanje tipa

* Kompajler može da automatski odredi tip promenljive na osnovu vrednosti koja se dodeljuje

**var x = 5;**

* + Ako se koristi **var?**, mora se odmah izvršiti dodela vrednosti

3. Objektno programiranje u C# jeziku

**2)Sta je objektno programiranje ?**

Identifikuje entitete (objekte) sa kojima program operiše.

* Za entitete se identifikuju
  + podaci koji se evidentiraju
  + operacije koje se mogu izvršavati nad entitetom

Apstrakcija (Za neki opis objekta)

* + Zanemaruju se osobine entiteta koje nisu važne (u datom kontekstu)za konkretan problem

Klasifikacija

* + Entiteti se grupišu po tipu( npr Doktor)
  + Time su identifikovane klase entiteta
  + Pojedinačni entiteti su primerci svoje klase

**3)Objekat vs Klasa**

* Objekat je konkretan entitet(Razlika izmedju objekta I klase je u nivou apstrakcije klasa je visi od objekta)
  + Ima podatke koji opisuju njegove osobine
* Klasa je šablon koji opisuje sve entitete određenog tipa i definiše

Ima atribute i metode

Preklapanje metoda (overloading)

* Može da postoji više metoda istog imena, ali sa različitim argumentima

**4)Enkapsulacija** get/set metode

* + program koji koristi klasu ne treba da poznaje detalje implementacije te klase
  + atributi ne trebaju direktno da budu dostupni spolja

**5) Properties**

* + obezbeđuju efikasniji mehanizam enkapsulacije
  + get/set metode koje se koriste kao atributi

**private double x;**

**public double X**

**{**

**get { return x; }**

**set { x = value; }**

**}**

**6)Prostori imena (namespaces)**

* Tipovi su organizovani u prostore imena zbog
  + jedinstvene identifikacije tipova
  + organizacije tipova po srodnosti

**7) .NET**

Slično kao paketi u Javi, ali .NET prostori imena predstavljaju logičku strukturu koja ne mora biti u skladu sa strukturom fajlova i foldera

**8)Nasledjivanje**

Različite klase mogu da dele deo osobina i operacija

Sve klase u C# implicitno nasleđuju klasu **System.Object**

sa metodama :

1ToString()-konv sadrz obj u tekst

2Equals(Object)- poredi 2 objekta

3GetHashCode()-vraca hash code(sadržaj objekta mapiran na numeričku vrednost)

4GetType()-Vraća **Type** objekat

**9) Polimorfizam**

Objekat se može posmatrati kao da je višestrukog tipa (polimorfno)

* Implicitna konverzija - iz nasleđenog tipa u bazni tip
* Eksplicitna konverzija – iz baznog tipa u nasleđeni tip

**10) Dynamic (Late)binding**

* Omogućuje nezavisno i naknadno proširivanje funkcionalnosti bez izmene postojećeg koda
* Ako postoji jedna osnovna klasa koju nasleđuje više klasa naslednica

**11) Virtuelne metode i redefinisanje metoda (overriding)**

* Metoda pretka čije ponašanje može biti u klasi naslednici izmenjeno (redefinisano) metodom koja ima istu deklaraciju
* To je preduslov da bi *dynamic binding* funkcionisao

**12) Apstraktne klase i metode**

SLUZI SAMO DA BUDE NASLEDJENA I NJENE METODE TAKODJe

Može da sadrži apstraktne metode

**13) Interfejsi**

INTERFEJS JE SKUP METODA KOJE KLASA MORA DA IMA , AKO KLASA IMPLEMENTIRA TAJ INTERFEJS

KOD INTERFEJSA MORA SVE DA SE IMPLEMENTIRA ZA RAZLIKU OD APSTRAKTNE KLASE

**14) Statičke klase**

Staticka klasa je zatvorena klasa sadrzi staticke atribute i metode i staticka klasa se nece nasledjivati

**15) Modifikatori pristupa**

* Za klasu
  + internal
  + public

Za atribut

Private - atribut dostupan samo unutar klase

Protected - atribut dostupan unutar klase i iz klasa naslednica

Internal-Atribut dostupan unutar klasa istog sklopa

Public - javno

Protected internal – mesavina ove dve

**16) Zapečaćene (sealed) klase i metode**

* Zapečaćena klasa je klasa koju nije moguće naslediti
  + Zapečaćena metoda je redefinisana metoda koju nije dalje moguće redefinisati

**17)Ugradjene klase**

* Klasa definisana unutar neke klase

18)**Anonimne klase**

* Moguće je definisati novu klasu u trenutku kreiranja objekta te klase
* Ova klasa nema ime

**19) Extension Methods**

* Moguće je naknadno proširiti klasu metodama bez izmene izvornog koda klase
* Koristi se za klase preuzete iz postojećih biblioteka

4)Windows Presentation Foundation - WPF

**20) WPF**

* WPF je tehnologija za razvoj korisničkog interfejsa u .NET aplikacijama

**21)Komponente interfejsa**

* \*Prikaz teksta (Labele)
* \*Dugmad (..)
* \*Polja za unos (Input polja ,combo boxovi)
* Grupisanje elemenata (group boxovi , tagovi)
* \*Prikaz podataka(tabele , liste (data grid))
* \*Meniji (meniji za rad)

**21) Grid**

* + Panel za raspoređivanje elemenata(redova,kolona ,...)

pogledati pred usmeni wpf posle grida clanove neke …. Do 47 slajda

5. WPF – Povezivanje kontrola sa podacima (data binding )

**22) Veza kontrole sa podacima**

* Iz objektnog modela potrebno je prekopirati podatke u kontrole u kojima se podaci prikazuju
* Nakon izmene u kontrolama, potrebno je osvežiti objektni model

**23) Data Binding**

- WPF mehanizam koji automatski vrši ove operacije ^^^^^^

-Generalno, Data Binding obezbeđuje automatsko sinhronizovanje vrednosti određenog objekta sa njegovim izvorom podataka

**24) Smer povezivanja**

* Jednosmerno (OneWay) – izmena izvornog podatka automatski menja ciljni podatak, ali obrnuto ne važi
* Dvosmerno (TwoWay) – izmene nad izvornim podatkom menjaju ciljni podatak i obrnuto
* Jednosmerno ka izvornom podatku (OneWayToSource) – izmena ciljnog podatka automatski menja izvorni podatak, ali ne i obrnuto
* Samo jedanput (OneTime) – pri inicijalizaciji, ciljni podatak dobija vrednost izvornog podatka, ali se kasnije promene izvornog podatka ne reflektuju na vrednost ciljnog podatka

**25) Trenutak povezivanja**

* Svojstvo definiše u kom trenutku se podaci iz jednog objekta kopiraju u drugi

**26)DataBinding – Implementacija**

* Prikaz imena studenta u TextBox komponenti
* TextBox koji prikazuje ime studenta – **Path** definiše izvorno svojstvo koje se kopira u ciljni objekat

6 -pred pogledati Icollection view i Collection view source

**27) Filtriranje**

* Korišćenjem **ICollectionView**
* Korišćenjem **CollectionViewSource**

**6.1 C# Događaji**

**28)Delegati**

* **U toku izvršavanja određuje** se koja metoda će obaviti ispisivanje
* Delegat je kao referenca na metodu (Pokazivacccc)

**29)Dogadjaj**

* Mehanizam kojim objekat obaveštava korisnika objekta da je izvršena određena operacija nad objektom
* Implementiraju se korišćenjem delegata
* U trenutku kada se događaj desi, klasa obezbeđuje poziv metode, a korisnik klase određuje koja se konkretno metoda poziva i obrađuje događaj

6.2 C# Životni ciklus objekta

**30) Kreiranje objekta**

**Racunar r = new Racunar();**

* 1. Inicijalizuje se objekat
     + Pod kontrolom programera kroz kod u konstruktoru

**31) Uništavanje objekta**

* Garbage Collector
  + deo CLR koji se povremeno aktivira i uništava objekte koji se ne koriste
* Uništavanje se sastoji od
  + Izvršavanja destruktora (ciscenje na kraju neko )
    - Nekad je potrebno izvršiti programski kod nakon uništavanja objekta da bi sistem nastavio ispravan rad
    - Programer piše destruktor
  + Dealociranja memorije
    - CLR oslobađa memoriju koja je bila zauzeta od strane objekta

**32) Garbage Collector – aktiviranje**

* GC se aktivira samo kada ima potrebe
* (Zasto se ne preferira destructor ? ) 1:00 8 pred otp
* Niti znamo da li ce se pozvati nit ga mozemo eksplicitno pozvati jedini eksplicitni nacin se ne preporucuje sto? Zato sto je citav concept urajdjen po tome da ne razmisljamo kada ce se izvrsavati
* Preporucuje se reader.Close(); ili Using(){}….
* GC uništava samo objekte na koje ne postoji nijedna aktivna referenca

7 ADO .Net

**33)Ado .net**

* ADO.NET predstavlja skup klasa koje obezbeđuju servise za pristup podacima
* Dve grupe servisa
  + Preuzimanje podataka iz izvora podataka
  + Pristup podacima

**34) Preuzimanje podataka**

* zaduzen Data Provider
* Data Provider - driveri sa kojima mozemo da se povezemo sa razlicitim izvorima podataka

**35) Uspostavljanje veze – klasa Connection** pr6slajd

* Connection
  + Uspostavljanje i upravljanje vezom sa izvorom podataka
* Različiti tipovi konekcija za različite izvore podataka
  + SqlConnection – za Microsoft SQL Server
  + OleDbConnection – za pristup izvoru podataka preko Microsoft OLE DB API
  + OdbcConnection – za pristup izvoru podataka preko ODBC API
  + OracleConnection – za pristup Oracle DBMS

**36) Uspostavljanje veze – klasa Connection String**

* Connection String – tekst koji specificira izvor podataka i parametre povezivanja
* **conn.ConnectionString = @"**
* **Data Source = .\SQLExpress;**
* **Integrated Security = true;**
* **Initial Catalog = ComputerShop";**

**37) Uspostavljanje veze – Connection pooling**

* *Connection pooling* – mehanizam za optimizaciju korišćenja resursa pri povezivanju sa jednim izvorom podataka
* Automatski se inicijalizuje određen broj veza ka jednom izvoru podataka

**conn.Open(); // kreira se pool**

**38) Command**

* + Izvršavanje upita nad izvorom podataka
  + Moguće je poslati i preuzeti parametre
  + DbCommand – klasa predak za sve tipove komandi. Postoje različiti tipovi komandi za različite izvora podataka

**39) Kreiranje komande**

* Imamo praznu komandu moramo sad neki upit da joj def
* Prva varijanta
  + Instanciranje komande i postavljanje konekcije
  + **SqlCommand comm = new SqlCommand();**
  + **comm.Connection = conn;**
* Druga varijanta
  + Odgovarajući tip komande se dobija iz konekcije
  + **SqlCommand comm = conn.CreateCommand();**

**40) Izvrsavanje komande**

* Pozivom različitih metoda komande, zavisno od tipa podatka koji je rezultat izvršavanja:
  + ExecuteReader - Vraća **DataReader** objekat; Koristi se kada je rezultat upita kolekcija podataka

**(SELECT \* FROM STUDENT)**

* + ExecuteScalar - Vraća jedan podatak

**(SELECT IME FROM STUDENT WHERE ID=1)**

* + ExecuteNonQuery – Za upite koji ne vraćaju podatke (fejs slike primer)

**(UPDATE STUDENT SET IME=‘PETAR’ WHERE ID=1)**

* + ExecuteXMLReader – Vraća **XMLReader** objekat; Podržano samo za SqlCommand klasu

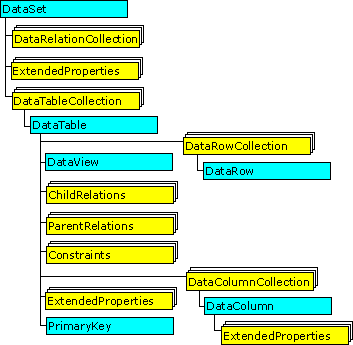
**41) Parametrizovane komande** [1]

* Komanda može da sadrži parametre upita
* (Kada imamo insert update delete upite znaci kada treba nalepiti na nas upit nesto sto treba da korisnik unosi koristi se parametrizovani upit ) def
* Tipovi parametara
* Input (ulazni - default), Output (izlazni), InputOutput (ponaša se i kao ulazni i kao izlazni), ReturnValue (sadrži povratnu vrednost komande – npr. kod poziva uskladištene procedure)

**42) DataReader**

* Objektna struktura koja sadrži preuzete redove iz izvora podataka

**43) Pristup podacima**

* DataSet - memorijska reprezentacija podataka preuzetih iz izvora podataka
* Preko njih cemo najlakse raditi presipanje iz baze u memoriju
* 

**44) DataSet**

* Struktura DataSet objekta odgovara relacionoj bazi podataka
* DataSet sadrži tabele
* Tabela sadrži redove
  + Red sadrzi podatke
* Tabela sadrži kolone
* Između tabela mogu se definisati relacije
  + DataRelation klasa predstavlja relaciju između dve tabele

**45) DataAdapter**

* Kopiranje podataka iz izvora podataka(Baze ) u DataSet i obrnuto
* Iz memorije u bazu i obrnuto
* Zaduzen za azuriranje podataka
* Popunjavanje DataSet objekta
  + Metoda **Fill** izvršava komandu predstavljenu svojstvom **SelectCommand**

8.1 WPF Resursi

**46) Wpf Resursi**

* Deljeni objekti koji se mogu koristiti u različitim delovima aplikacije
* Primer
  + Konverter objekat koji se može koristiti bilo gde unutar **Window** elementa

**<Window.Resources>**

**<local:ValidatorConverter x:Key="converter" />**

**</Window.Resources>**

**47) Staticki resursi**

* Vrednost statičkih resursa je poznata u početku izvršavanja aplikacije
* Ne menjaju se pri izvršavanju aplikacije
  + Prvi način

**<object property="{StaticResource key}" .../>**

**48) Dinamički resursi**

* Vrednost dinamičkih resursa se postavlja tek u toku izvršavanja aplikacije
* Primer – u toku izvršavanja programa postavlja se boja dugmeta

8.2 WPF **Validacija**

**49) Validacija**

* Provera ispravnosti podataka pri unosu
* Osnova iz 2 stvbari : utvrdjivanje da nesto nije ispravno I da kazemo to korisniku
* Ukoliko unesena vrednost nije u skladu sa određenim kriterijumom, korisnik u grafičkom interfejsu dobija obaveštenje da je vrednost neispravna

**50) Vrste (tehnike ) validacije**

* + Validacija u slučaju izuzetka
  + Validacija putem validacionih pravila
  + Validacija korišćenjem **IDataErrorInfo** interfejsa

Validacija u slučaju izuzetka:

* Najjednostavniji oblik validacije
* Ako se pri prebacivanju podatka iz izvornog objekta u ciljni pojavi izuzetak, korisnik dobija obaveštenje o neispravnoj vrednosti

Validacija putem validacionih pravila:

* Koristi se u slučaju kada neispravan podatak ne izaziva izuzetak, već je potrebno u skladu sa određenim kriterijumom utvrditi da li je podatak ispravan
* Kreira se klasa naslednica klase **ValidationRule**
* Primer:
* Unos E-mail adrese studenta
* E-mail je u klasi **Student** tipa **string**, pa bilo koja unesena vrednost ne izaziva izuzetak
* Vrednost je validna ukoliko je u skladu sa formatom e-mail adrese

Validacija putem IDataErrorInfo interfejsa :

* Koristi se kada validnost podatka zavisi od vrednosti drugih podataka
* Primer
* ako je za studenta unesen podatak o datumu odbrane diplomskog rada, tada je obavezno uneti i naslov teme rada

Svojstvo **Error** definiše grešku na nivou celog objekta

**Indekser** za validaciju konkretnog atributa klase(konkretno kaze sta je problem )

Primer Error:

**public string Error**

**{**

**get**

**{**

**return "Neispravni podaci o studentu";**

**}**

**}**

Primer indekser :

**public string this[string propertyName]**

**{**

**get**

**{**

**switch (propertyName)**

**{**

**case “Email”:**

**bool valid = EMailValidacija(); if (!valid)**

**return “Neispravan e-mail”;**

**51) Prikaz greške pri validaciji**

* 1. Default – crveni okvir oko kontrole
* 2. Prilagođeni prikaz
  + U posebnoj labeli poruka o grešci – direktan prikaz poruke o grešci
  + U posebnoj labeli poruka o grešci – konverzija greške preko konvertera
  + Definisanjem šablona za prikaz greške pri validaciji

**8.3 WPF Stilovi**

**52) WPF Stilovi**

* WPF Ima podrsku za izgled kroz stilove .
* Logika izmedju tih WPF je da radvojimo organizaciju aplikacije od njenog vizuelnog izgleda
* Tehnologija za dizajniranje izgleda komponenti grafičkog interfejsa u WPF aplikacijama

**53) Primena stila na komponentu**

* Tag **Style uvodjenjee krozz**
* **TargetType** – tip komponente na koju se stil primenjuje
* **Setter** – postavljanje vrednosti za određeno svojstvo komponente
* **Key**
  + naziv stila
  + **ako nije definisan Key, stil se primenjuje na sve komponente**

**54) Proširenje stila**

**-Mozemo naslediti stil uzmemo postojeci pa dodamo jos neke stvari**

**-Based on - naziv stila na kojem je novi baziran**

* – naziv stila na kojem je novi stil baziran //\*\*\*\*

**<Style TargetType="TextBlock" BasedOn="{StaticResource {x:Type TextBlock}}" x:Key="Title">**

**<Setter Property="FontSize" Value="18"/>**

**<Setter Property="FontWeight" Value="Bold"/>**

**</Style>**

**<TextBlock Name="lbTitle" Text="Podaci o studentu" Style="{StaticResource Title}" />**

?) Sablon podataka , Sablon kontrole

DataTemplate – sablon koji definise kako se prikazuju podaci

ContolTemplate – definise strukturu prikaza grafickih elemenata u kontroli

**9. Language-Integrated Query LINQ**

**55) LINQ**

* LINQ je jezik za upite nad podacima (SLICAN PRINCIPU SQL)
* Integrisan je u .NET kao deo sintakse jezika

**56) LINQ upit**

* LINQ upit
  1. Definisanje izvora podataka
  2. Kreiranje upita
  3. Izvršavanje upita

Izvor podataka –BILO STA NAD CIME MOZEMO UPIT IZVRSITI

* Upit se izvršava odloženo
  + pri korišćenju promenljive koja predstavlja rezultat upita
* Povratna vrednost je (NESTO STO JE ENUMERABILNO)**IEnumerable<T>** ili **IQueryable<T>**
* **Struktura linq upita je : from , where ,select , orderby,group,join**

**57) Razlika izmedju LINQ I ADO .NET**

* Kao alternativa za LINQ to SQL razvijen je ADO .NET Entity Framework koji pruža širi skup funkcionalnosti i mogućnost rada sa različitim SUBP