Objektno orijentisano programiranje 2

Platforma .NET

Literatura

- Watkins, D., Hammond, M., Abrams, B., *Programming in the .NET Environment*, Addison Wesley, 2002.
- Jones, A., Freeman, A., *C# for Java Developers*, Microsoft Press, 2002.
- *Microsoft C# Language Specification*, Microsoft Press, 2015, https://www.microsoft.com/en-us/download/details.aspx?id=7029 .
- Kraus, L., *Programski jezik C# sa rešenim zadacima*, Akademska misao, 2016.
- Kraus, L., *Rešeni zadaci iz programskog jezika C#*, Akademska misao, 2017.
- Stutz, D., Neward, T., Shilling, G., *Shared Source CLI Essentials*, O'Reilly, 2003.

Uvod

- Jezik C# je definisan u Microsoftu, ali je standardizovan (ECMA-334)
- Internacionalni standard: ISO/IEC 23270:2023 (novembar)
  - ISO/IEC standard iz 2023. godine odgovara verziji C# 7.0 iz 2017. godine
- Multijezička .NET platforma
  - osnovica za razvoj i izvršenje programa na C#
  - C# nije jedini jezik za razvoj aplikacija za .NET platformu
- U osnovi – slična ideja kao kod Jave:
  - C# se prevodi u međukod
  - međukod se ne interpretira,
    već se JIT prevodiocem prevodi u toku izvršenja

Motivacija

- Od pojave Jave (1996), do pojave .NET platforme i jezika C# (2002.) značajne promene na polju mogućnosti i korišćenja ICT:
  - eksplozija Interneta
    - raspoloživost visoko-propusnih veza za firme i individualne korisnike
  - velike promene u projektovanju softverskih sistema preduzeća
    - tanki klijenti, višeslojni serveri
  - razvoj i usvajanje otvorenih standarda za podršku integracije sistema
    - XML – široko prihvaćeni standard za reprezentaciju i razmenu podataka
  - ekspanzija inteligentnih uređaja
    - mobilni telefoni, PDA, konzole za igre
  - porast raspoloživih besplatnih softverskih alternativa
    - *open source* inicijativa
  - orijentacija kompanija na nove ICT u interakciji B2C i B2B
    - eksplozija interesovanja za veb servise u komunikaciji između preduzeća i potrošača

Istorijat

- Anders Hejlsberg, glavni arhitekta za C#
  - 1996. prešao iz Borlanda u Microsoft
  - tvorac Turbo Pascal-a, glavni arhitekta za Borland Delphi
- Januar 1999. – započet razvoj, pod imenom jezika "*Cool*"
- 11.07.2000. – Microsoft najavljuje .NET radni okvir
  - za objedinjenje jezika za Web namene
- Sredinom 2000. – Microsoft podnosi predlog standarda C#
  - organizacija ECMA (*European Computer Manufacturers Association*)
- 13.12.2001. – ECMA standardizuje C# (334) i CLI (335)
- Trenutno aktuelne ECMA specifikacije:
  - 7. izdanje C# (2023., C# v7.0) i 6. izdanje CLI (2012.)
- Standardi međunarodne organizacije za standarde:
  - ISO/IEC 23270:2023 (C# v7.0) i 23271:2012 (CLI)
- Aktuelne verzije Microsoft specifikacija:
  - C# v12.0 (2023.) i .NET (CLR): v8.0.4 (2024.)

Ciljevi i ciljne karakteristike

- Microsoftova namera da izađe na tržište:
  - softverskih sistema za preduzeća široke skale (*large-scale enterprise systems*)
  - Internet i distribuiranih sistema
  - mobilnih uređaja
- Potreba za integrisanom platformom za
  - razvoj
  - distribuciju i
  - izvršenje aplikacija
- Ciljne karakteristike radnog okvira (platforme) .NET
  - platformska nezavisnost (načelno)
  - jezička neutralnost
  - racionalizacija i konsolidacija tehnologija
  - integracija sistema korišćenjem XML-a
  - ekvivalent Java platformi koji se sastoji od
    - skupa klasnih biblioteka
    - izvršnog okruženja CLR (odgovara JVM)

Platformska nezavisnost

- Java slogan – "piši jednom, izvršavaj svuda“
  - JVM, JDK za većinu platformi
- Nedostatak
  - slabije mogućnosti korišćenja specijalizovanog hardvera i API-ja OS
- .NET – drugačiji pristup:
  - izgrađen na Windows platformi i podržava osobine Win32 API
  - definisan PAL (*Platform Adaptation Layer*)
    - definiše korišćene Win32 API pozive
  - CLI (*Common Language Infrastructure*)
    - zajednička infrastruktura za (razne) jezike
- ECMA standardizuje CLI (osnova PAL, obuhvata CTS, CLS,VES,CIL, bibl.)
  - implementacija PAL je moguća za bilo koju platformu, preslikavanjem Win32 API poziva
- JVM – široko raspoloživa, *de facto* standard
- CLI – standard (ECMA i ISO/IEC)

Rotor i Mono

- Rotor
  - realizacija ECMA CLI u vidu deljenog izvornog koda (*shared source*) - SSCLI
  - nije zasnovan na izvornom kodu komercijalnog .NET okruženja
  - sponzor projekta Microsoft
  - sadrži podskup mogućnosti radnog okruženja .NET, isključeni važni elementi:
    - pristup bazama podataka (ADO.NET)
    - alati za razvoj grafičkih korisničkih interfejsa (*Windows Forms*)
    - mehanizmi za pisanje serverskih aplikacija (ASP.NET)
  - isključivo je namenjen nekomercijalnim aplikacijama
    - ne može da se koristi za razvoj proizvoda koji su konkurencija
  - raspoloživ za Windows i FreeBSD
- Mono
  - sponzor: Novell (inicijalno), Xamarin, Microsoft
  - otvoreni kod, nema ograničenja za primenu (MIT licenca)
  - raspoloživ za Linux, Windows, Mac OS, Solaris, iOS, Android i druge OS
- Postoje i drugi: .NET Core, Portable .NET (deo dotGNU projekta)

Jezička neutralnost

- Java je u početku bio jedini jezik Java platforme (JVM)
  - kasnije i drugi jezici koji se prevode u bytecode, npr. Jython (Python), Scala,…
- .NET platforma je osmišljena kao jezički neutralna
  - aplikacije i komponente se pišu na raznim jezicima
- Opšti sistem tipova – CTS (*Common Type System*)
- Opšta specifikacija jezika – CLS *(Common Language Specification)*
- Programi pisani na svakom jeziku koji poštuje CLS,
  a stvara i koristi CTS tipove, mogu da se izvršavaju na .NET platformi
  - C#, VB, C++, F#, J# (Java), A# (Ada), L# (Lisp), P# (Prolog), ...
- Postojeći jezici koji se prilagođavaju .NET platformi malo menjaju sintaksu
  - Visual Basic i Visual Basic.NET nisu identični jezici
  - C++ za .NET (upravljani C++) razlikuje se od std. C++ (*Managed Extensions*)
- .NET je prilagođen OO jezicima, proceduralni jezici se teško adaptiraju
  - ipak, postoji čak i COBOL.NET, kao i funkcionalni jezici kakav je SML

MSIL i JIT prevođenje

- Izvorni program na nekom višem jeziku se prevodi u MSIL
- MSIL (*Microsoft Intermediate Language*)
  - međukod (odgovara bajtkodu)
- MSIL instrukcije se prevode u toku izvršenja programa
  - u instrukcije mašinskog jezika domaćina u toku izvršenja
  - JIT prevođenje (*Just-In-Time Compilation*)
- Nije moguća interpretacija MSIL
  - JIT prevođenje ne može da se isključi
  - razlika u odnosu na Java platformu,
    gde je prvobitno zamišljeno da se bajtkod interpretira
- Standard CLI definiše CIL (*Common Intermediate Language*)

Izvršno okruženje (CLR)

- CLR (*Common Language Runtime*)
- Pandan JVM
- Izvršno okruženje CLR je odgovorno za:
  - upravljanje izvršenjem koda
  - proveru valjanosti tipova
  - JIT prevođenje
  - automatsko upravljanje memorijom
  - izvršavanje više konkurentnih niti
  - bezbednost i sigurnost
  - integraciju sa operativnim sistemom računara-domaćina
- CLR se startuje automatski kada se neka .NET aplikacija pokrene
- Standard CLI definiše VES (*Virtual Execution System*)

CTS i CLS

- Zajednički sistem tipova
  - CTS (*Common Type System*)
  - definiše kako se tipovi opisuju, koriste i upravljaju u vreme izvršenja
  - osnova za pisanje tipova u jednom jeziku koji će se koristiti u drugom
- Zajednička specifikacija jezika
  - CLS (*Common Language Specification*)
  - objekti stvarani u različitim jezicima moraju da ispolje prema klijentima samo one karakteristike koje su zajedničke za sve jezike
  - Primeri:
    - globalni statički podaci i funkcije nisu prilagođeni specifikaciji CLS
    - pre pristupa nasleđenim podacima, konstruktor mora da pozove konstruktor osnovne klase
  - komponentama koje poštuju CLS se garantuje da će da budu upotrebljive od drugih komponenata

Biblioteka klasa

- Tipovi u biblioteci podržavaju CTS
  i mogu da se koriste iz jezika čiji prevodilac poštuje CLS
- Raspon .NET biblioteke klasa je sličan rasponu biblioteke klasa za Javu
- Oblasti koje pokrivaju tipovi biblioteke klasa:
  - osnovni tipovi podataka i izuzetaka
  - ulaz/izlaz
  - rad sa nitima i konkurentno programiranje
  - rad preko mreže i distribuirano programiranje
  - refleksija (introspekcija) tipova
  - sigurnost platforme i aplikacije
  - integracija sa operativnim sistemom domaćina
  - grafički korisnički interfejs
  - pristup bazama podataka
  - XML Web servisi
  - ...

Alati za razvoj i jezici

- Alati za rad iz komandne linije:
  - .NET SDK (slično JDK)
  - besplatni, mogu da se koriste i za razvoj komercijalnih aplikacija
- Alat sa integrisanim okruženjem (IDE – *Integrated Development Environment*):
  - Visual Studio.NET (Microsoft)
  - komercijalno raspoloživ – izdanja: Professional, Enterprise
  - besplatna *Community* varijanta
  - izuzetno doprinosi produktivnosti u odnosu na alate iz komandne linije
  - integrisanost sa .NET platformom i .NET serverima (MS SQL Server, MS IIS)
- Višejezična podrška – fundamentalni cilj CLR
- Jezici koje podržava Visual Studio.NET:
  - C#.NET, Visual Basic.NET, Visual C++.NET, F#.NET, JScript.NET
- Drugi jezici za .NET:
  - dugačka lista (http://en.wikipedia.org/wiki/List_of_CLI_languages)

Integracija platformi

- Preduzeća se često opredeljuju između dve platforme: Java i .NET
- Ponekad obe platforme figurišu u istom preduzeću
- Delovi sistema rade na različitim platformama
- Potrebna je komunikacija između sistema koji rade na različitim platformama
  - neophodna je integracija platformi

- Rešenje za integraciju platformi – otvoreni standardi:
  - XML za reprezentaciju podataka
  - XML Web servisi za integraciju sistema
    - omogućavaju komunikaciju između slabo spregnutih delova sistema
    - delovi sistema mogu da se izvršavaju na različitim platformama (Java, .NET, ...)

Migracija među platformama

- Predviđena dva alata za migraciju softvera sa Java na .NET platformu
- Visual J# .NET
  - sintaksa odgovara sintaksi jezika Java,
  - kod se prevodi na MSIL, ne na bajtkod
  - J# ne koristi Java biblioteku klasa, već .NET bibiloteku klasa
  - J# nije kompatibilan sa J2SE, nedostaju neke funkcionalnosti iz J2SE
  - omogućava relativno laku migraciju koda pisanog na J++ u J# za .NET
- Java Language Conversion Assistant (JLCA)
  - konvertuje J++ u C# aplikacije
  - uspešno se konvertuju sve jezičke konstrukcije
  - samo deo biblioteke klasa se konvertuje
- Oba alata Microsoft je kasnije povukao iz Visual Studio paketa

Izvršni sklopovi

- Izvršni sklopovi (*Assemblies*) su jedinice za:
  - isporuku,
  - reupotrebu,
  - verzionisanje i
  - sigurnost
- Sklopovi su više logičke nego fizičke strukture (mogu da budu u više fajlova)
- Sklopovi sadrže:
  - manifest sklopa
  - metapodatke koji opisuju sadržane tipove
  - međukod (MSIL) sadržanih tipova
  - resurse (kao što su, na primer, ikone, slike, podaci)
- Manifest sadrži podatke o sklopu koje koristi CLR
- Na osnovu manifesta CLR:
  - puni sklop i potrebne biblioteke
  - obavlja proveru valjanosti tipova
  - podržava verzionisanje
  - vrši bezbednosne provere

Primer "Zdravo"

- Trivijalna klasa sa glavnim programom koji ispisuje “Zdravo!”
- Klasa je napisana u fajlu `MojaAplikacija.cs`:

```
class Pozdrav {
  static void Main(string[] args){
    System.Console.WriteLine("Zdravo!");
  }
}
```

- Koristi se ASCII ili UTF-8 editor
- Prevođenje: `csc MojaAplikacija.cs`
- Stvara se sklop `MojaAplikacija.exe` (a ne `Pozdrav.exe`)
- Opšta sintaksa: `csc [<opcije>] <lista fajlova>`
- Prilikom izvršenja aplikacije ne poziva se eksplicitno CLR
  - CLR se poziva implicitno, CLR poziva `Main` metod

Vrste sklopova

- Mogu da se naprave sledeće vrste sklopova:
  - konzolni izvršni (opcija: `/target:exe`, datoteka: `*.exe`)
    - može da se izvršava kao samostalna konzolna aplikacija
    - sklop mora da ima jednu ulaznu tačku definisanu kao `Main` metod
  - grafički izvršni (opcija: `/target:winexe`, datoteka: `*.exe`)
    - može da se izvršava kao samostalna *Windows* aplikacija
    - sklop mora da ima jednu ulaznu tačku definisanu kao `Main` metod
  - modul (opcija: `/target:module`, datoteka: `*.netmodule`)
    - kolekcija prevedenog koda koja se koristi samo u drugim sklopovima
    - ne može da se izvršava samostalno
  - biblioteka (opcija: `/target:library`, datoteka: `*.dll`)
    - sklop je biblioteka tipova koju dinamički koriste drugi sklopovi

Manifest

- Manifest sadrži metapodatke koji opisuju sklop i tipove koje sklop sadrži
- Sklop identifikuju sledeći podaci:
  - ime (*name*) – ime sklopa
  - verzija (*version*) – četiri brojača: glavni, sporedni, revizija i gradnja (*build*)
  - kultura (*culture*) – opcione informacije o kulturi i jeziku koji podržava sklop
    - zahteva se za sklopove koji sadrže lokalizovane resurse
  - jedinstveno ime (*strong name*) – jedinstveno ime sklopa
    - obuhvata ime, verziju, kulturu i javni ključ koji odgovara privatnom kojim je potpisan sklop
    - digitalni potpis onemogućava promenu nakon stvaranja
    - koristi se za deljene sklopove između aplikacija
- Ostali podaci manifesta:
  - sadržaj (*assembly contents*) – lista fajlova koji čine sklop
  - tipovi (*type information*) – detalji o tipovima koje izvozi sklop
  - reference (*references*) – detalji o drugim sklopovima koji se statički koriste iz tipova u sklopu
  - opšte informacije (*general information*) – opšti detalji (proizvođač, opis, pravo kopiranja)
- Većina podataka manifesta se automatski generiše
  - opšte informacije navodi programer

Moduli

- Moduli su predviđeni kao podrška za razvojni proces
  - moduli mogu da se razvijaju nezavisno
  - moduli mogu da se pišu na različitim jezicima koji poštuju CLS
  - takvi moduli se integrišu u izvršnom sklopu
- Karakteristike:
  - moduli sadrže jedan ili više tipova opisanih na MSIL jeziku
  - koncept modula je nezavisan od koncepta prostora imena
    - moduli mogu da sadrže tipove iz više prostora imena
    - jedan prostor imena može da se proteže na više modula
  - moduli se prevode iz jednog ili više fajlova sa izvornim kodom
  - moduli ne mogu da se nezavisno (direktno) koriste od strane CLR
  - moduli se kombinuju u sklopovima

Prevođenje i sadržaj modula

- Modul se prevodi na sledeći način:
  `csc /target:module /out:MojModul Fajl1.cs Fajl2.cs`
- Podrazumevana ekstenzija imena modula je `.netmodule`
- Rezultat prevođenja je sklop-modul: `MojModul.netmodule`
- Moduli sadrže sledeće informacije:
  - metapodatke koji opisuju tipove u modulu i njihove zavisnosti od tipova u drugim modulima
  - tipove prevedene iz izvornih fajlova, grupisane prostorima imena (tipovi su prevedeni na MSIL)
- Modul koji zavisi od tipova u drugom modulu (`MojModul.netmodule`) se prevodi:
  `csc /addmodule:MojModul.netmodule /target:module Fajl.cs`

Sklopovi od jednog ili više fajlova

- Sklop može da bude u jednom fajlu ili u više fajlova
- Sklop od jednog fajla
  - kombinuje jedan modul sa metapodacima u istom fajlu
- C# prevodilac podrazumevano pravi sklopove u jednom fajlu
- Sklop od više fajlova
  - sastoji se od jednog ili više modula i posebnog fajla sa manifestom
- Sklopovi od više fajlova se koriste u sledećim situacijama:
  - za kombinovanje modula pisanih u različitim jezicima
  - da se retko korišćeni tipovi smeste u poseban modul, jer se modul puni samo po potrebi
  - za podršku nezavisnom razvoju komponenata sistema

Pravljenje sklopa od jednog fajla

- Za razliku od Jave, izvršni .NET sklop se ne pokreće navođenjem klase
- Ako više od jedne klase definiše `Main` metod
  - potrebna je `csc` opcija `/main:<klasa>.Main`
  - u sklopu se čuva identitet ulazne tačke aplikacije
- Metapodaci sklopa mogu da budu definisani u posebnom C# izvornom (`.cs`) fajlu
- MS VS .NET automatski stvara `AssemblyInfo.cs`
  - u ovom fajlu se definišu metapodaci aplikacije
  - fajl se koristi za formiranje manifesta sklopa
- Primer `AssemblyInfo.cs`:

```
using System.Reflection;
using System.Runtime.CompilerServices;
[assembly: AssemblyTitle("MojSklop")]
[assembly: AssemblyDescription("Primer sklopa u jednom fajlu")]
```

- Stvaranje sklopa izvršne aplikacije:

```
csc /target:exe /out:Aplikacija.exe
MojaAplikacija.cs AssemblyInfo.cs
```

Pravljenje sklopa od više fajlova (1)

- Sklopovi sastavljeni od više fajlova se stvaraju pomoću linkera `al.exe`
- Linker generiše fajl koji sadrži samo podatke manifesta sklopa
  - u donjem primeru će da se formira fajl: `aplikacija.exe`
- Primer (prva klasa u fajlu `StampacStringova.cs`, a druga u fajlu `Pozdrav.cs`):

```
public class StampacStringova{
  public void StampajString(string stringPoruke){
    System.Console.WriteLine("Poruka: " + stringPoruke);
  }
}
class Pozdrav{
  public static void Main(string[] argumenti){
    StampacStringova mojStampac=new StampacStringova();
    mojStampac.StampajString("Zdravo!");
  }
}
```

Pravljenje sklopa od više fajlova (2)

```
csc /target:module StampacStringova.cs
csc /addmodule:StampacStringova.netmodule /target:module Pozdrav.cs
al /out:aplikacija.exe /target:exe /main:Pozdrav.Main
                Pozdrav.netmodule StampacStringova.netmodule
```

- Ako neki od fajlova sklopa nedostaje u vreme izvršenja:
  - CLR prijavljuje grešku
- Metapodaci sklopa ne mogu da se zadaju u fazi prevođenja, već tek u fazi povezivanja
- Primer:

```
al    /out:aplikacija.exe
      /target:exe
      /main:Pozdrav.Main
      /title:MojSklop
      /description:"Primer sklopa u više fajlova"
      Pozdrav.netmodule StampacStringova.netmodule
```