Projekt izrade aplikacije

2014/15.02

Projekt

- **Projekt**
  - Projekt je vremenski određeno nastojanje da se proizvede jedinstven proizvod, usluga ili rezultat. [PMBOK – Project Management Book of Knowledge, PMI]
  - Projekt je niz jedinstvenih, složenih i povezanih aktivnosti koje imaju određeni cilj i koji se mora postići u zadanom vremenskom roku, u okviru zadanog proračuna i u skladu sa specifikacijama. [Wisocky, Beck and Crane]

- **Vremenska određenost**
  - Svaki projekt mora imati jasno određen početak i kraj. Projekti mogu biti kratki ili trajati godinama, ali će svakako završiti.
  - Projekt završava u trenutku kada postane jasno da su ciljevi projekta dostignuti ili kada se zaključi da ciljevi projekta ne mogu ili neće biti dostignuti.
- **Jedinstvenost**
  - Projekt se odnosi na rad na nečemu što prije nije postojalo i što se razlikuje od rezultata nastalih sličnim projektima.

Upravljanje projektom

- **Upravljanje, rukovođenje projektom (Project management)**
  - Upravljanje projektima je primjena znanja, vještina, alata i tehnika u projektnim aktivnostima da bi se ispunili projektni zahtjevi. [PMI]
- **Upravljanje projektom uključuje**
  - Planiranje
    - Utvrđivanje zahtjeva
    - Postavljanje jasnih i ostvarivih ciljeva
    - Uravnoteženje zahtjeva na kvalitetu, doseg, vrijeme i trošak,
    - Prilagodbu interesima i očekivanjima zainteresiranih strana – dionika (eng. Stakeholders)
  - Organiziranje
    - Formiranje projektnog tima
  - Raspoređivanje obaveza
    - Tko što i kada treba napraviti
  - Usmjeravanje
    - Nadgledanje, omogućavanje izvršenja
  - Kontroliranje
    - Provjera učinka i rezultata

Uloge na projektu

- **Korisnik, Korisnik usluga, Klijent (User, Customer, Client)**
  - osoba ili grupa, naručitelj ili krajnji korisnik
- **Sponzor projekta (project sponsor)**
  - Osoba ili grupa koja osigurava (financijske) resurse za projekt
- **Voditelj projekta (project manager)**
  - Osoba imenovana kako bi ostvarila ciljeve projekta
- **Resursi projekta**
  - Osobe, oprema, usluge, materijal, budžet ili druga sredstva.
- **Projektna ekipa**
  - Svi članovi ekipe, uključujući upravljačke, a u nekim slučajevima i sponzora
    - voditelj – upravljanje projektom
    - sistem analitičar – određivanje potreba, specifikacija zahtjeva i dizajna
    - projektant/arhitekt – uspostava osnovne arhitekture
    - razvojnik (developer, builder) – kodiranje, testiranje
    - administrator baza podataka – administriranje DBMS
    - sistem inženjer / sistem administrator – administriranje OS i mreže

Dokumentiranje projekta

- **Povelja projekta (Project Charter)**
  - Dokument kojim pokretač projekta ili sponzor odobrava projekt i ovlašćuje voditelja za primjenu organizacijskih resursa u provedbi projekta.
- **Plan projekta = Plan upravljanja softverskim projektom**
  - IEEE Standard for Software Project Management Plans 1058-1998
  - dokument koji opisuje sveukupnu organizaciju projekta
- **Primjeri 🗁Prilozi**
  - PlanProjekta.dot
  - Firma-PlanProjekta.doc
- **Plan može sadržavati raspored**
  - općenito: SoftwareDevelopment.mpp, PlanRazvoja.mpp
  - konkretno: Firma-PlanProjekta.mpp

1. Introduction
   - 1.1 Project Overview
   - 1.2 Project Deliverables
   - 1.3 Evolution of the Software Project Management Plan
   - 1.4 Reference Materials
   - 1.5 Definitions and Acronyms
2. Project Organization
   - 2.1 Process Model
   - 2.2 Organizational Structure
   - 2.3 Organizational Boundaries and Interfaces
   - 2.4 Project Responsibilities
3. Managerial Process
   - 3.1 Management Objectives and Priorities
   - 3.2 Assumptions, Dependencies, and Constraints
   - 3.3 Risk Management
   - 3.4 Monitoring and Controlling Mechanisms
   - 3.5 Staffing Plan
4. Technical Process
   - 4.1 Methods, Tools, and Techniques
   - 4.2 Software Documentation
   - 4.3 Project Support Functions
5. Work Packages, Schedule, and Budget
   - 5.1 Work Packages
   - 5.2 Dependencies
   - 5.3 Resource Requirements
   - 5.4 Budget and Resource Allocation
   - 5.5 Schedule
6. Additional Components
7. Index
8. Appendices

Izrada plana projekta

- **Koraci izrade plana projekta**
  - Izrada liste zadataka
  - Izrada hijerarhije zadataka (work breakdown structure)
  - Procjena trajanja zadataka
  - Izrada ovisnosti među zadacima
  - Dodjela resursa

	Zadatak	Trajanje (u danima)	Prethodnici	Naziv resursa
1	**Doseg**	**7**		
2	Određivanje dosega	5		Voditelj projekta
3	Određivanje sponzorstva	1,5	2	Sponzor projekta;Voditelj projekta
4	Određivanje resursa	2	2	Voditelj projekta
5	Dovršetak dosega	0	3;4	
6	**Analiza/Softverski zahtjevi**	**16**		
7	Analiza potreba	5	5	Sistem analitičar
8	Prikupljanje informacija	6	7	Sistem analitičar
9	Prijedlog izvedbe sustava	5	8	Sistem analitičar
10	Dovršetak analize	0	9	
11	**Dizajn**	**5**		
12	Razvoj funkcionalnih specifikacija	5	10	Sistem analitičar;Projektant
13	Izrada baze podataka	5	10	Administrator baze podataka
14	Dovršetak dizajna	0	13;12	
15	**Razvoj**	**35**		
16	Izrada formi korisničkog sučelja	4	12	Razvojnik
17	Izrada funkcija za pohranu podataka	12	14	Razvojnik
18	Izrada funkcija za ispis izvještaja	15	14	Razvojnik
19	Izrada izvještaja	10	18	Razvojnik
20	Izrada funkcija izračuna	12	16	Razvojnik
21	Razvojno testiranje(debugiranje)	10	16;17;18;19;20	Razvojnik
22	Dovršetak razvoja	0	21	
23	**Testiranje**	**12**		
24	Izrada testova programskig cjelina prema specifikacijama proizvoda	5	14	Tester
25	Izrada plana integracijskog testiranja prema specifikacijama proizvoda	5	14	Tester
26	Testiranje komponenti prema specifikacijama proizvoda	7	24	Tester
27	Provjera integracije modula	7	25	Tester
28	Dovršetak testiranja	0	26;27	
29	**Dokumentacija**	**5**		
30	Razvoj specifikacija i sustava pomoći	5	22	Razvojnik
31	Dovršetak dokumentacije	0	30	Razvojnik
32	**Uvođenje sustava i poduka korisnika**	**5**		
33	Ugradnja programske potpore	5	28	Razvojnik;Sistem administrator
34	Dovršetak uvođenja sustava i poduke korisnika	0	33	

Izrada liste zadataka

- **Zadaci – osnovni gradbeni elementi svakog projekta**
  - predstavljaju posao koji se mora obaviti da bi se postigao cilj projekta
  - opisuju tijek događaja, trajanja i zahtjeva za resursima na projektu
  - primitivni zadaci
    - zadaci koji se dekompozicijom ne mogu podijeliti na jednostavnije zadatke
  - skupni zadaci (summary tasks)
    - zbrajaju trajanje i troškove primitivnih zadataka
    - trajanje, datum te izračunate vrijednosti se automatski izvode iz skupa primitivnih zadataka
  - prekretnice ili miljokazi (milestones)
    - ključni događaj ili krajnji rok odnosno cilj koji treba postići
    - trajanja 0
    - služe za provjeru stupnja dovršenosti drugih zadataka
    - pomak ključnog događaja ima za posljedicu vremenski preraspored

Izrada hijerarhije zadataka

- **Faza - grupa povezanih zadataka koji se odnose na fazu projekta**
  - Zbirni zadaci se odnose na faze
- **WBS (work breakdown structure)**
  - hijerarhijska lista faza, zadataka i prekretnica
  - osnova za pregledni raspored projekta
- **Dva su pristupa razvoju zadataka i faza:**
  - Planiranje s vrha prema dolje (Top-down)
    - pristup od općeg prema specifičnom
    - identificira glavne faze i rezultate projekta prije dodavanja zadataka potrebnih za završetak tih faza
    - složeni projekti mogu imati nekoliko slojeva faza
  - Planiranje s dna prema dolje (Bottom-up)
    - pristup od specifičnog prema općem
    - identificira što više zadataka najnižeg sloja prije grupiranja u faze

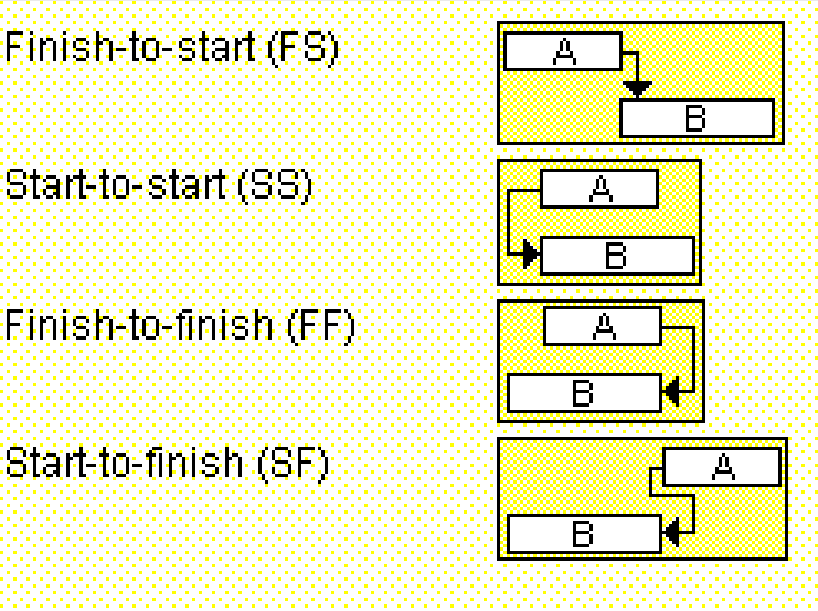
Procjena trajanja zadataka

- **Trajanje zadatka**
  - očekivana količina vremena za završetak zadataka
    - minute (m), sati (h), dani (d), tjedni (w), mjeseci (mo)

	Zadatak	Trajanje (u danima)	Prethodnici	Naziv resursa
1	**Doseg**	**7**		
2	Određivanje dosega	5		Voditelj projekta
3	Određivanje sponzorstva	1,5	2	Sponzor projekta;Voditelj projekta
4	Određivanje resursa	2	2	Voditelj projekta
5	Dovršetak dosega	0	3;4	
6	**Analiza/Softverski zahtjevi**	**16**		
7	Analiza potreba	5	5	Sistem analitičar
8	Prikupljanje informacija	6	7	Sistem analitičar
9	Prijedlog izvedbe sustava	5	8	Sistem analitičar
10	Dovršetak analize	0	9	
11	**Dizajn**	**5**		
12	Razvoj funkcionalnih specifikacija	5	10	Sistem analitičar;Projektant
13	Izrada baze podataka	5	10	Administrator baze podataka
14	Dovršetak dizajna	0	13;12	

Međuzavisnost zadataka

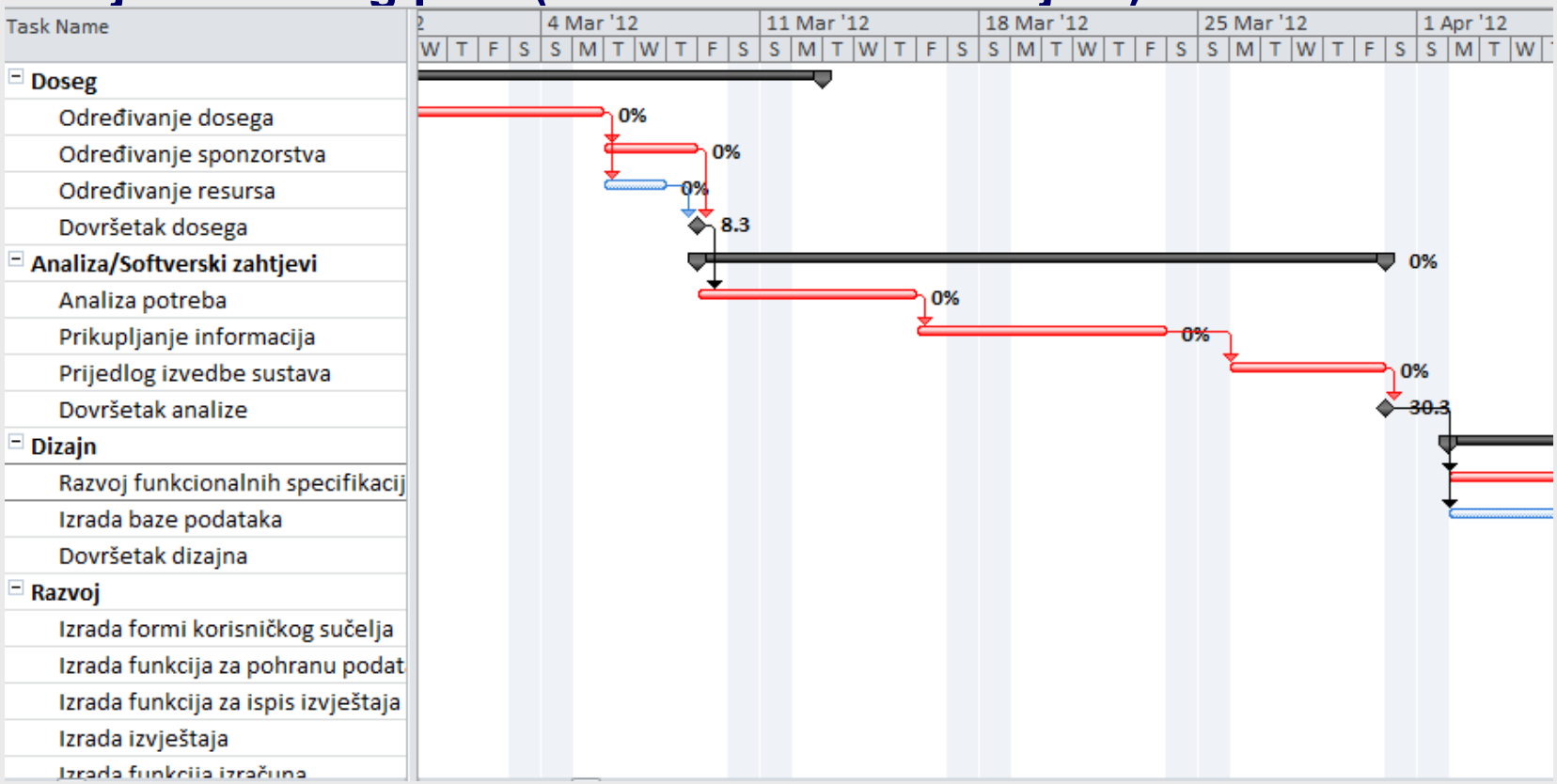
- **Projekt može zahtijevati da zadaci budu napravljeni u određenom redoslijedu**
  - Niz – iza jednog slijedi drugi zadatak
  - Zavisnost – sljedbenik (successor) može biti izvršen ako je dovršen prethodnik (predecessor)
  - Bilo koji zadatak može biti prethodnik jednom ili više sljedbenika



- Odnosi između zadataka:
  - Finish-to-start (FS) – završni datum prethodnika jest početni sljedbenika
  - Start-to-start (SS) - početni datum prethodnika utvrđuje početni sljedbenika
  - Finish-to-finish (FF) - završni datum prethodnika utvrđuje završni sljedbenika
  - Start-to-finish (SF) - početni datum prethodnika utvrđuje završni sljedbenika

Kritični put

- **Kritični put**
  - niz zadataka koji moraju završiti na vrijeme da bi projekt završio na vrijeme
  - svaki zadatak na kritičnom putu je kritični zadatak
  - kašnjenje kritičnih zadataka uzrokuje kašnjenje projekta
- **Primjer kritičnog puta (koristeći alat MS Project)**



Resursi

- **Resursi - sredstva**
  - ljudi, oprema i materijal potrebni za obavljanje zadataka
- **Vrste resursa:**
  - Resursi rada (work resources)
    - ljudi (ograničeno vrijeme rada)
    - oprema (neograničeno vrijeme rada)
  - Resursi materijala (material resources)
    - potrošni materijal koji predstavlja projektni utržak
    - daje informaciju o brzini konzumiranja resursa
- **Dva važna pogleda na resurse:**
  - Raspoloživost - u koje vrijeme određeni resurs može raditi na zadatku i koliko posla može obaviti
  - Trošak – koliko novca će biti potrošeno na resurse

Raspodjela resursa

- **Unos resursa i pratećih podataka (dostupnost, trošak)**
- **Maksimalne jedinice (max. units*)***
  - prikazuju vrijednosti raspoloživosti resursa u postocima
  - 100% predstavlja jednog čovjeka punog radnog vremena
  - 300% predstavlja tri čovjeka punog radnog vremena
- **Osnova po kojoj je resursu dodijeljeno obavljanje posla**
- **Za pojedinačnu prilagodbu uvažavaju se radni i neradni dani resursa**
  - Primjer: ako kalendar evidentira radno vrijeme samo četvrtkom i petkom 13-17 sati, 100% raspoloživosti nekog resursa ne znači 40 satno tjedno radno vrijeme, nego 8 sati rada tjedno

Dodjela resursa

- **Dodjelom resursa zadatku, dodjeljuje mu se određeni posao**

- **Razlikovati posao (work) od trajanja (duration)**
  - Posao je stvarni rad i odnosi se na
    - zadatak - stvarni rad potreban za završetak zadatka
    - dodijeljeni zadatak - stvarni rad nekog resursa na nekom zadatku
    - resurs - ukupni rad neke osobe na svim zadacima
  - Trajanje = Posao / Jedinice (Duration = Work / Units)
    - Ovisi o kalendaru rada

- **Raspoređivanje temeljem napora (effort-driven scheduling)**
  - metoda planiranja koja se koristi kod ažuriranja resursa zadatka
  - trajanje obrnuto količini resursa

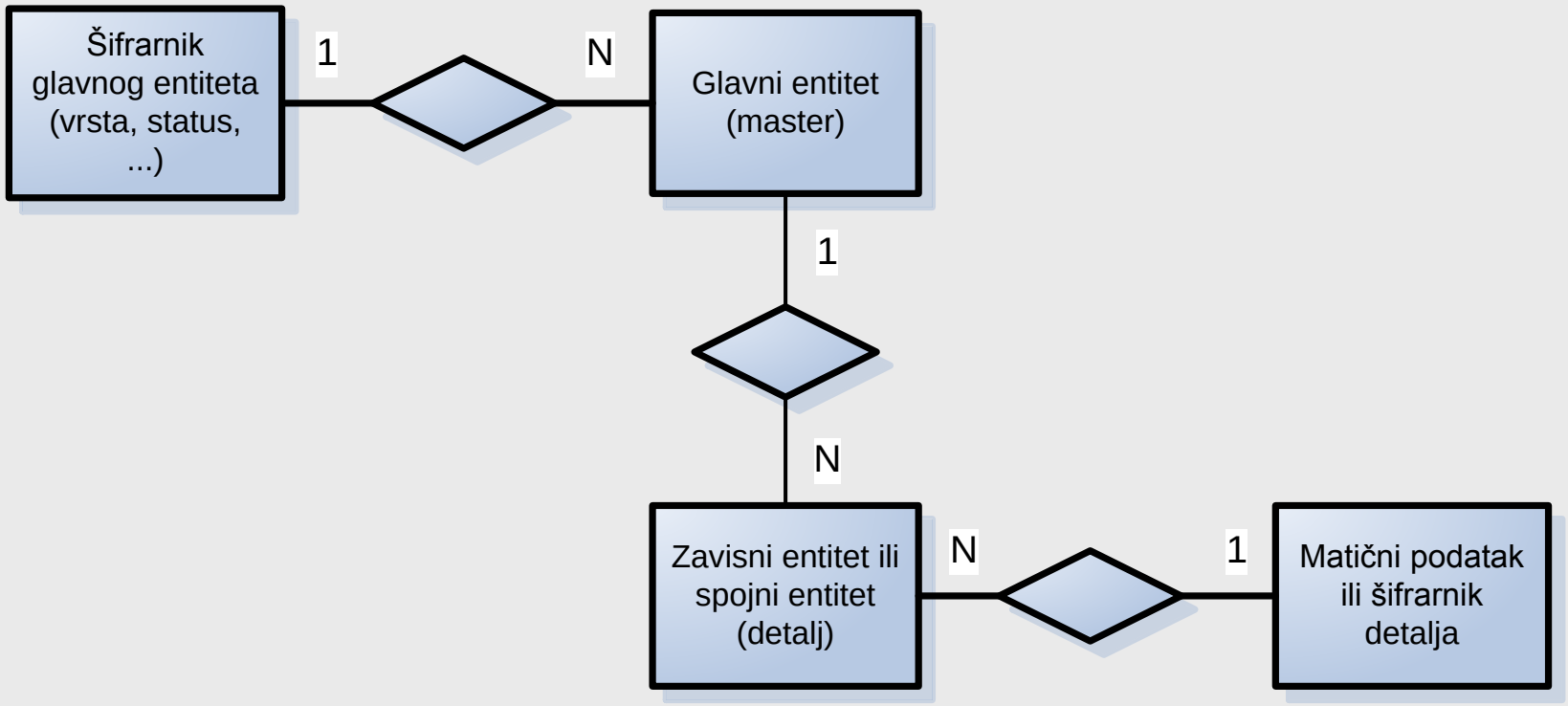
Prekapacitirani resursi

- **Rješavanje problema:**
  - Pomaknuti rokove (trajanje)
  - Dodati dodatne resurse
  - Produžiti radno vrijeme
  - Povećati jedinice posla
  - Smanjiti količinu posla
  - …

Zadaci

- **Projektni zadatak (za međuispit i završni ispit)**
  - Projektirati i izraditi sustav za podršku poslovanje ACI marina
  - Svaki član mora napraviti barem po jednu reprezentativnu zaslonsku masku za Windows i Web aplikaciju te WCF servis
  - Integraciju specifikacije posla, zajedničkih programskih knjižnica te ostale projektne dokumentacije radi ekipa zajednički.
- **1.DZ: Početni plan i raspored projekta – vidi Vrednovanje.pdf**
  - Kakav sustav ? Kako dobar sustav ? Kojim redom ? Kojim resursima ?
- **Realizacija**
  - Prikupiti i proučiti te provjeriti zahtjeve korisnika – intervjuiranjem, surfanjem, ...
  - Evidentirati problemska područja i zadatke
  - Uspostaviti životni ciklus razvoja na razini ekipe – plan faza razvoja
  - Popisati i pridijeliti nositelje zadataka (stvarne osobe, ne uloge)
  - Odrediti prekretnice s obzirom na realno stanje (cjeline posla, ciklusi, MI, ZI)
  - PlanProjekta - zajednički za ekipu
  - Evidentirati korisničke priče
  - Nacrtati konceptualni model podataka

Problemska područja



- **Matični podatak jednog problemskog područja smije biti glavni entitet u drugom problemskom području i obrnuto**

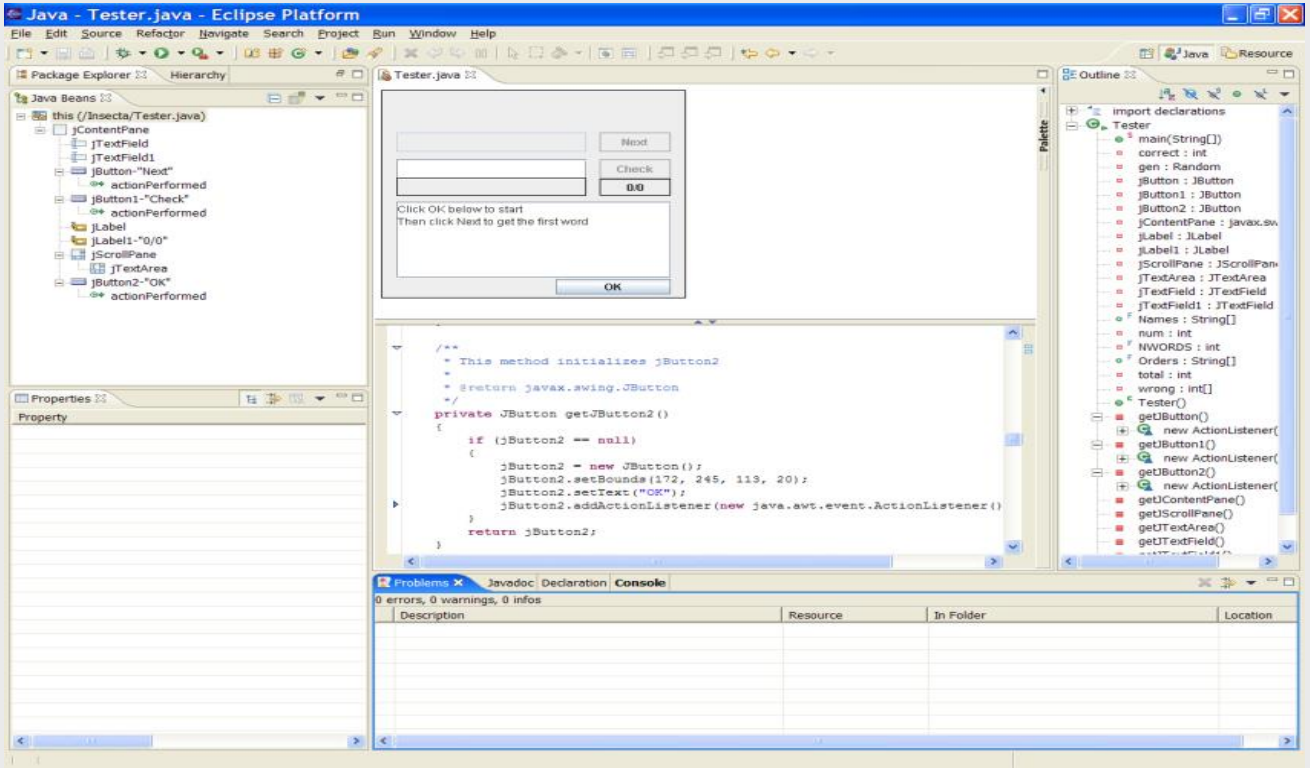
Integrirana razvojna okruženja

Integrirana razvojna okruženja

- **Integrated Development Environment (IDE) – okruženje, okolina za programiranje koje integrira**
  - uređivač izvornog koda (source code editor)
  - kompilator (compiler) i/ili interpreter (interpreter)
  - alat za izradu grafičkog korisničkog sučelja (GUI)
  - sustav za ispravljanje pogrešaka (debugger)
  - pomagalo za kontrolu verzija (version control system)
  - alate za automatsku izgradnju (build-automation tools)
  - alate za objektno orijentirano programiranje (npr. Class Browser)
  - alate za testiranje
  - …

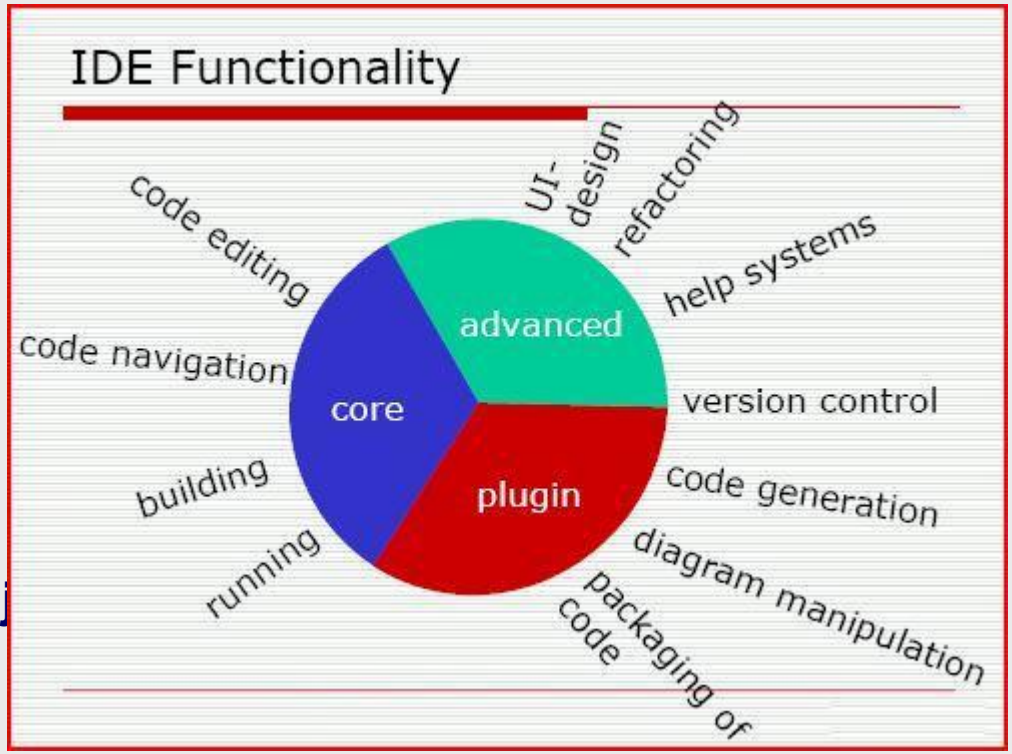
Primjeri IDE

- **Prvotno namijenjenj jednom programskom jeziku:**
  - Visual Basic, Delphi, Jbuilder....
- **Moderna okruženja podržavaju više jezika**
  - Visual Studio : C++, C#, Visual Basic.NET, ASP.NET
  - Eclipse - Eclipse primarno okruženje za Javu, ali sadrži umetke (plugins)
    - C/C++, Python, Perl, Ruby, Fortran, Cobol i dr.



Funkcionalnost i namjena IDE

- Označavanje sintakse (eng. syntax highlighting)
- Označavanje pogrešaka (eng. error highlighting)
- Pristup dokumentaciji (eng. documentation access)
- Kretanje kodom (eng. code navigation)
- Generiranje koda pomoću predložaka koda (eng. code generation through code templates)
- Podrška za preradu, refaktoriranj (eng. *refactoring)*
- Različite razine analize koda



Microsoft Visual Studio

- **Visual Studio je IDE za razvoj**
  - konzolnih aplikacija, desktop aplikacija, web aplikacija, web servisa, ...
  - na platformi .NET Framework (Windows PC, Windows Phone, ...)
- **Podržani jezici**
  - C++, C#, Visual basic, F#, …
- **Mogućnost dogradnje dodatnih razvojnih alata ...**
- **Povijest i opis izdanja**
  - http://en.wikipedia.org/wiki/Microsoft_Visual_Studio
  - aktualna: Visual Studio (2013) + .NET Framework 4.5
  - Alternative
    - Visual C# Express i Visual Web Developer Express (besplatni)
    - SharpDevelop (besplatan)

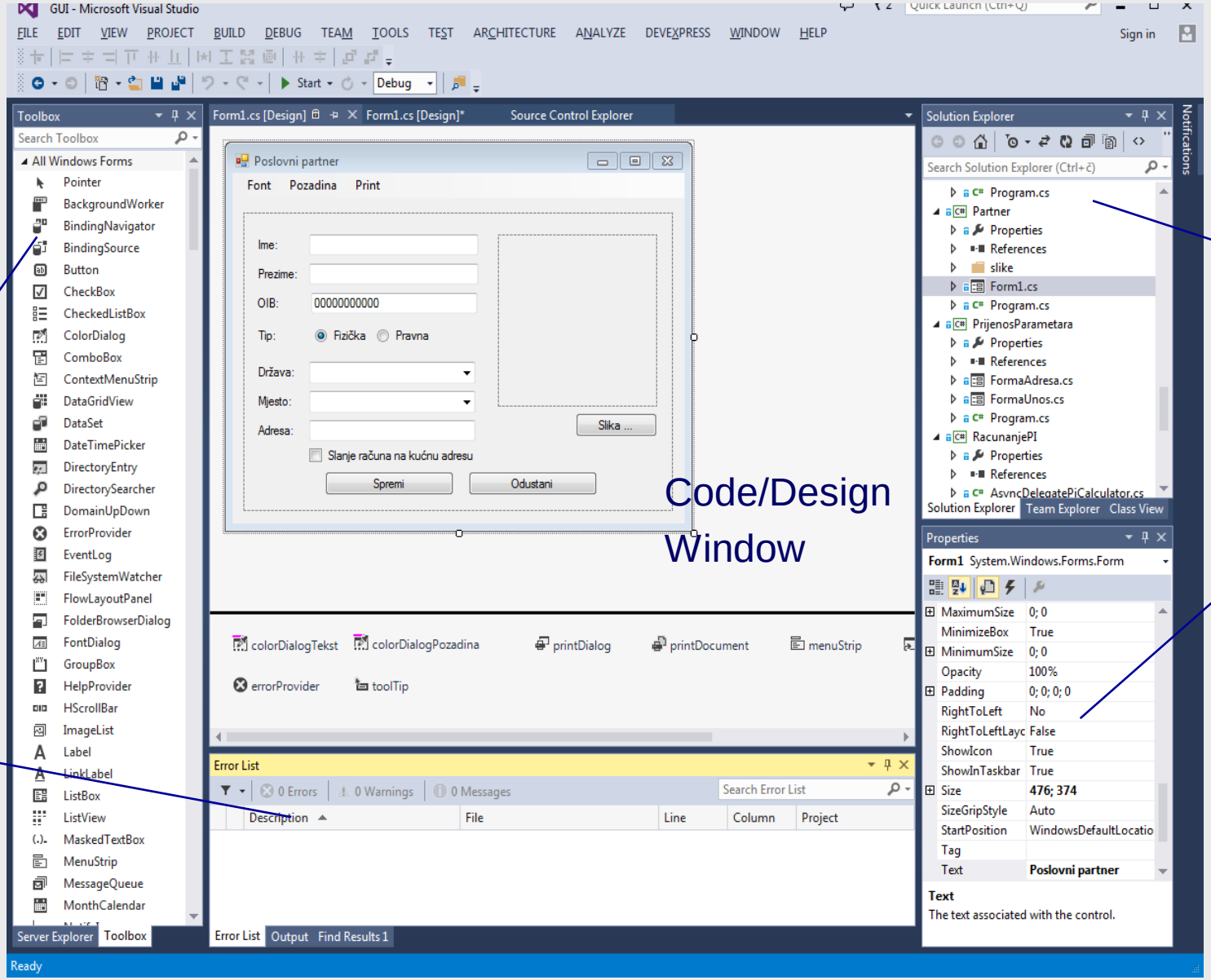
Primjena VS na izradu aplikacija

- **Konzolne aplikacije**
  - pokreću se s komandne linije i ne uključuju grafičko sučelje
- **Windows aplikacije (Windows Forms ili WPF)**
  - aplikacije s prozorima (forms)
  - Upotrebljavaju .NET Framework, koji trebaju za izvođenje.
- **Windows servisi**
  - su aplikacije koje rade u pozadini.
  - izvode programirane zadatke (scheduled tasks) ili obrađuju zahtjeve (npr. s mreže)
- **Web aplikacije (ASP.NET ili MVC)**
  - dinamičke web aplikacije, također podržane .NET Frameworkom.
- **Web servisi**
  - potprogrami na webu
- **Windows Phone aplikacije**
  - izvode se na uređajima koji imaju Compact framework (Pocket PC, Smartphone).
- **Ostale aplikacije**
  - MFC/ATL/Win32 aplikacije – u C++-u , ne trebaju .NET Framework.
  - Visual Studio dodaci (add-ins)
  - projekti za instalaciju aplikacija, rad sa bazama, kreiranje izvještaja, …
  - Korisničke biblioteke (class libraries)
  - …

Mogućnosti okoline Visual Studio (features)

- **IntelliSense**
  - pomoć uređivanju izvornog koda, predlaganje elemenata naredbi
- **Dizajnerski pogled (WSYIWYG)**
  - oblikovanje korisničkog sučelja za Windows aplikacije, ASP.NET aplikacije i Windows mobilne aplikacije
- **Otklanjanje pogrešaka (Debugging)**
  - prolaz naredbu po naredbu tijekom izvođenja
  - pregled i promjena sadržaja varijabli (auto, local) i praćenje izraza (watch)
  - moguće ulaženje u pozvane procedure
- **Interaktivna pomoć (Help)**
  - integrirana s elektroničkom knjižnicom hipertekstova MS Developer Network
  - pregled svih elemenata jezika i platforme te elementi priručnika
- **Organizacija programskog koda**
  - Project – skup datoteka koje predstavljaju istu aplikaciju ili objektnu datoteku dobivenu prevođenjem (assembly)
  - Solution – skup projekata koji čine kompletno poslovno rješenje

Sučelje razvojne okoline Visual Studio



Toolbox

Solution Explorer

Code/Design Window

Properties Window

Error Window

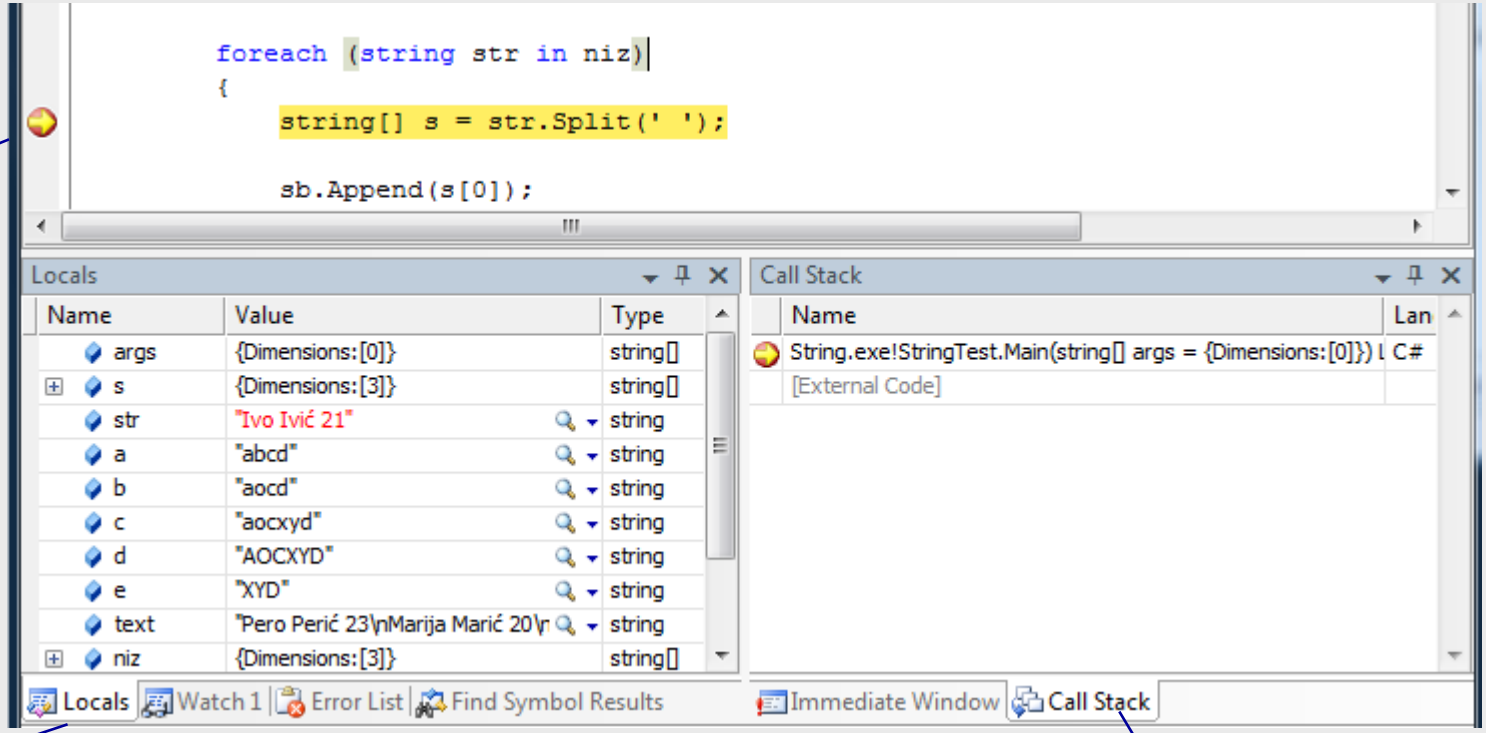
Sučelje razvojne okoline Visual Studio

- **Code/Design Window**
  - Prozor u kojem se piše kod ili dodaju kontrole u grafičkom (design) pogledu.
- **Solution Explorer**
  - Prozor s hijerarhijskom strukturom izvornih i pratećih datoteka aplikacije (projekti, datoteke, reference, ...)
- **Team Explorer**
  - Prozor za rad s rješenjima za timski rad, ukoliko je instaliran klijent za TFS.
- **Toolbox**
  - Prozor s kontrolama koje se metodom drag&drop stavljaju na grafički pogled.
- **Error Window**
  - Prozor u kojem se ispisuju poruke o pogreškama i upozorenja prilikom prevođenja programa.
- **Locals**
  - Prozor koji je vidljiv prilikom pokretanja programa. Služi za praćenje stanja varijabli tijekom izvođenja programa.
- **Properties Window**
  - Prozor za podešavanje svojstava (properties) pojedinog objekta.

- **Iz izbornika View mogu se dodati razni drugi prozori.**

Praćenje rada aplikacije (izbornik Debug)

Breakpoint – Točka prekida



Locals – praćenje vrijednosti varijabli

CallStack - Praćenje poziva postupaka

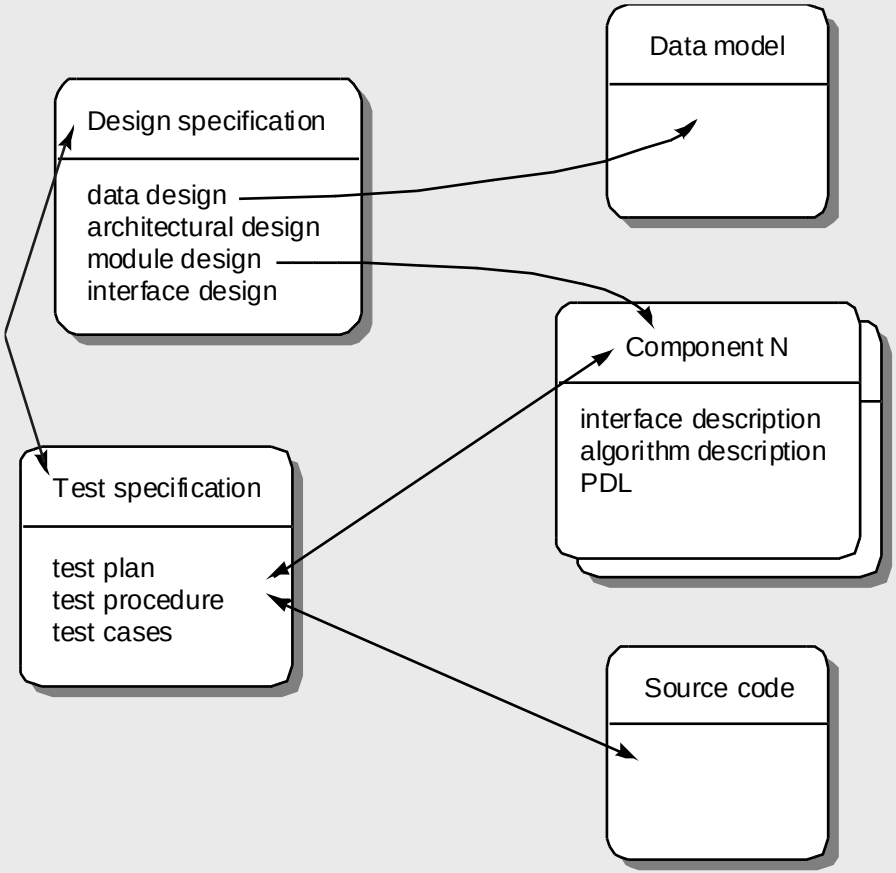
- **Watch – nadzor vrijednosti odabranih varijabli**
- **ImmediateWindow – izvršavanje naredbi u pogonu**
- **StepInto, StepOver**

Zadatak za vježbu

- **Dijelovi razvojne okoline mogu biti različito razmješteni, mogu plutati ili biti usidreni.**
  - Razmjestiti pojedine elemente
  - Pokazati/sakriti neke dijelove
  - Provjeriti dijelove izbornika View
  - Provjeriti postavke u Tools / Options
  - Resetirati razvojnu okolinu na tvorničke postavke.

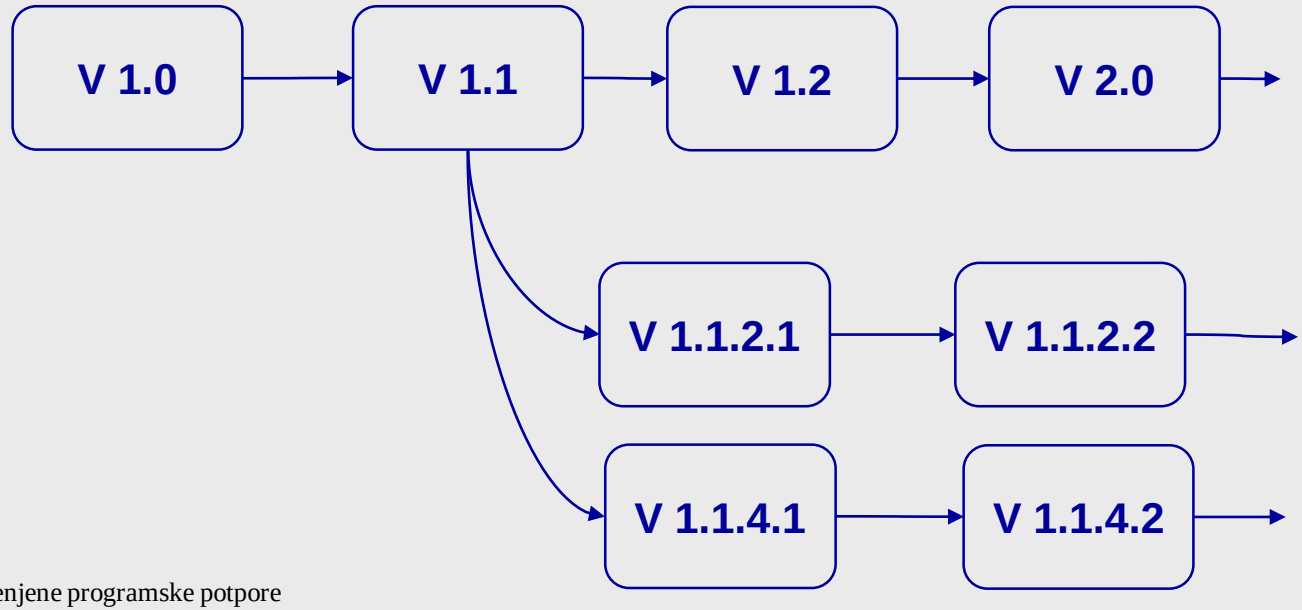
Upravljanje konfiguracijom

- **Konfiguracija**
  - imenovani skup konfiguracijskih elemenata u određenoj točki životnog ciklusa
- **Element konfiguracije (IEEE)**
  - agregacija hardvera i/ili softvera koja se tretira kao jedinka u procesu upravljanja konfiguracijom
- **Objekti konfiguracije**



Verzije konfiguracije

- verzija, inačica (version) – određeno izdanje (issue, release) proizvoda
- objava, isporuka (release) – originalna verzija u primjeni, npr. zadnja v2.0
- revizija (revision) – ona koja se koristi umjesto originalne, podrazumijeva izmjene u određenim vremenskim intervalima, npr. V1.2
- varijanta (variant) – alternativa originalu (hardverska platforma, različiti jezik), živi paralelno s njim, npr. v1.1.2.1
- osnovica (Baseline) – specifikacija proizvoda fomalno provjerena i usvojena, koja služi kao temelj razvoja i koja se mijenja samo kroz formalnu proceduru kontrole promjena, IEEE (IEEE Std. No. 610.12-1990)



Kontrola verzija

- **Kontrola verzija (Version control) = verzioniranje**
  - kombinira procedure i alate radi upravljanja različitim verzijama objekata konfiguracije, koji nastaju softverskim procesima
- **Mogućnosti sustava kontrole verzija**
  - baza projekata (project database) ili riznica (repository)
    - pohranjuje sve relevantne objekte konfiguracije
  - verzioniranje
    - razlikovanje pohranjenih inačica objekata konfiguracije
  - pomagalo za izradu (make facility)
    - prikuplja relevantne objekte i proizvodi određenu verziju softvera
  - praćenje problema (issues tracking), praćenje pogreški (bug tracking)
    - bilježenje i praćenje statusa tema koje se odnose na pojedine objekte konfiguracije

Označavanje konfiguracije

- **Verzija objektne datoteke (assembly) određena je s četiri broja:**

  `<major version>.<minor version>.<build number>.<revision>`

  - major version - mijenja se prilikom znatne promjene u (npr. kod redizajna koji prekida vertikalnu kompatibilnost sa starijim verzijama)
  - minor version - mijenja se prilikom znatne promjene, ali uz zadržavanje kompatibilnosti s prethodnim verzijama
  - build number - predstavlja ponovno prevođenje istog koda (npr. prilikom promjene platforme, procesora i slično)
  - revision - primjenjuje se npr. prilikom izdavanja sigurnosnih zakrpa i sličnih manjih promjena

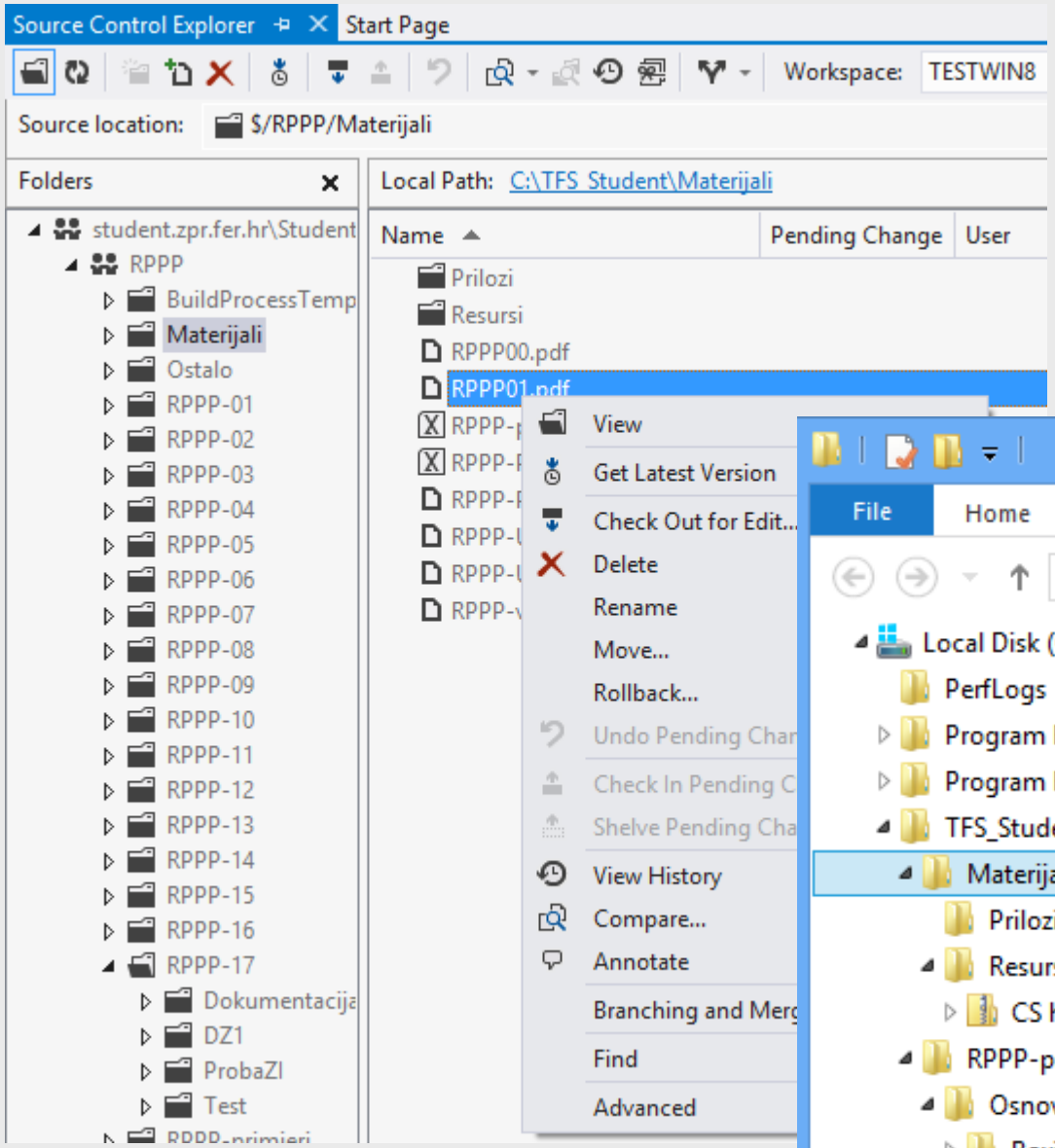
- **Primjer: projekt \ Properties \ Publish**

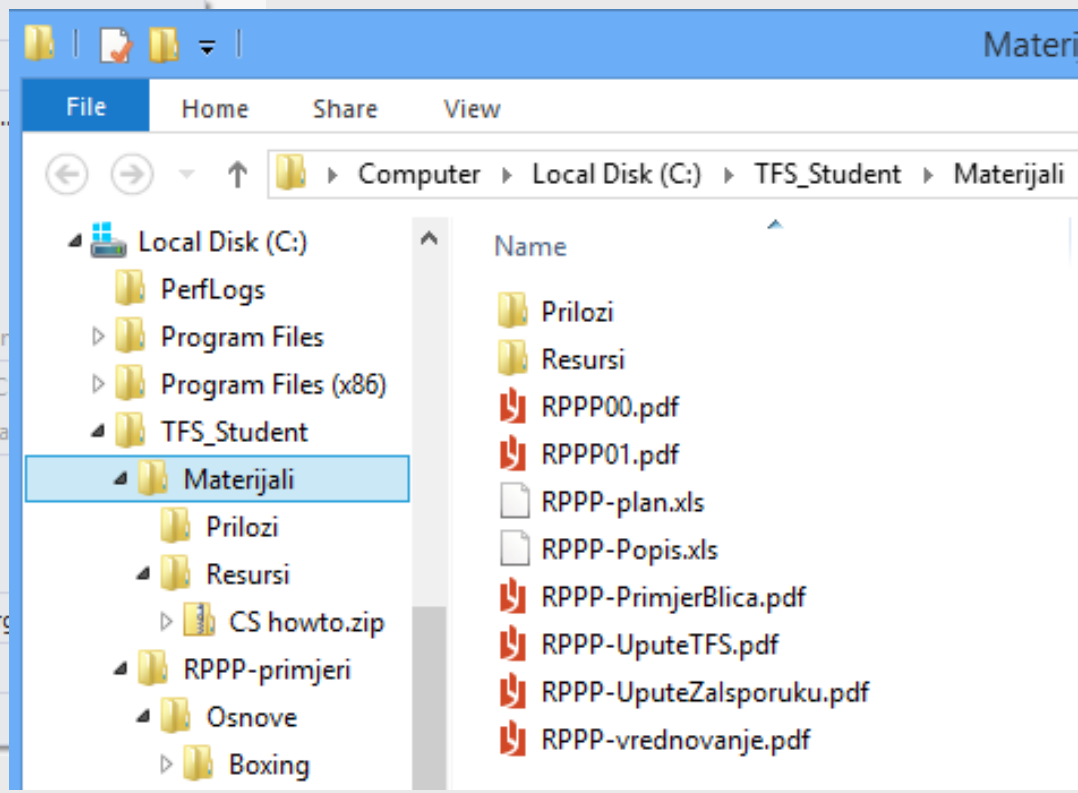
Automatsko i ručno verzioniranje

- **Automatsko označavanje**
  - prednosti:
    - eliminacija ručnog rada (npr. pisanja i izvedbe skripti)
    - ne postoje dvije inačice s istom oznakom
  - nedostaci:
    - oznaka elementa ne podudara se s oznakom cijelog sustava
    - novi brojevi ovise o danu i vremenu prevođenja
    - verzija se mijenja pri svakom prevođenju, neovisno o tome jesu li se dogodile promjene ili ne
- **Ručno verzioniranje**
  - prednosti:
    - potpuna kontrola nad brojevima verzije
    - moguća je sinkronizacija između verzije pojedinih komponenti i verzije cijelog sustava
  - nedostaci:
    - verzioniranje se mora raditi ručno
    - moguće je napraviti više različitih objektnih datoteka s istim oznakama

Kontrola programskog koda

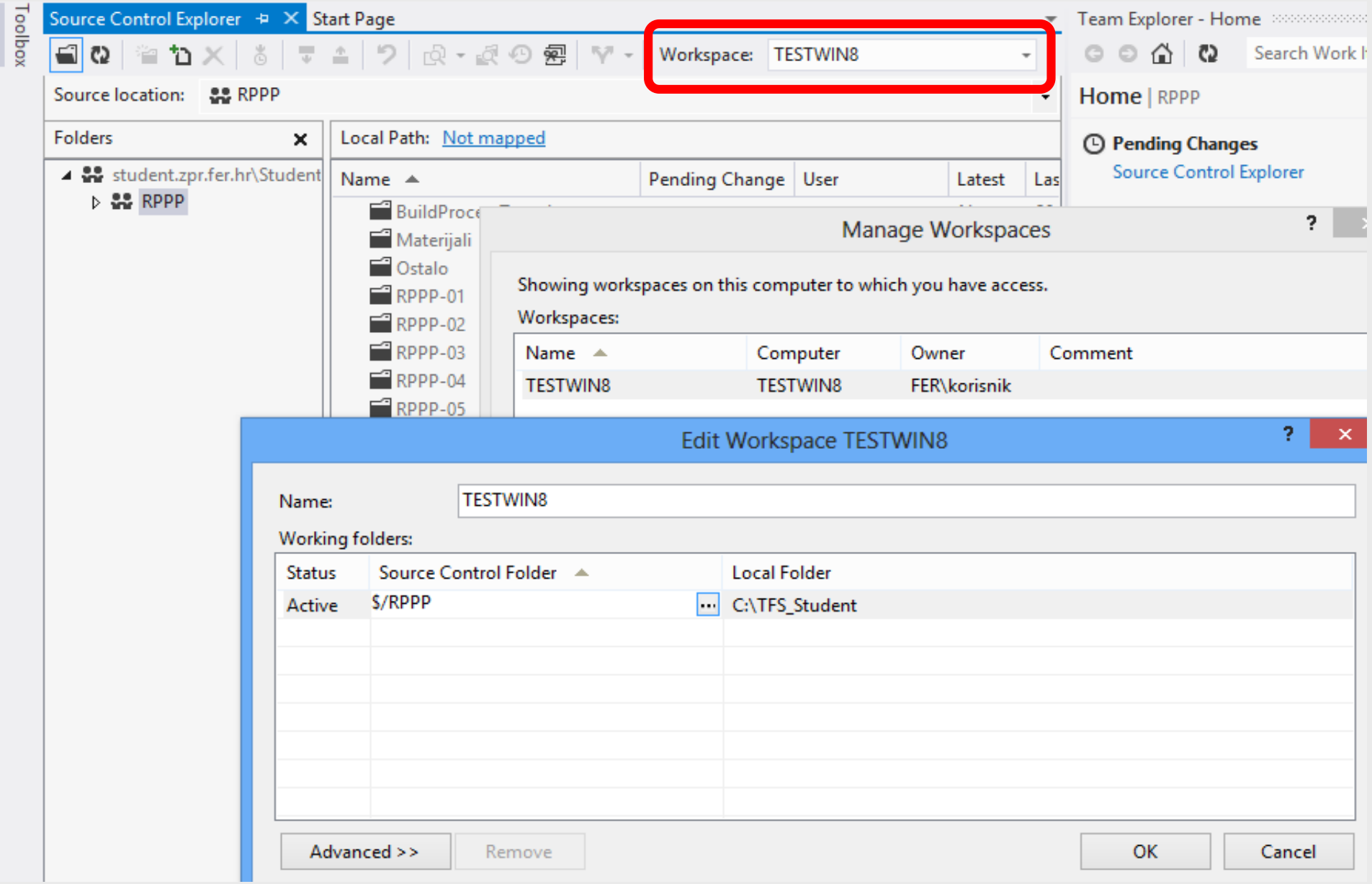
Kontrola izvornog koda





- Izvorni kod i ostale datoteke u zajedničkom repozitoriju (TFS)
- Lokalna kopija (najčešće) iste strukture

Veza između lokalne mape i mape na TFS-u



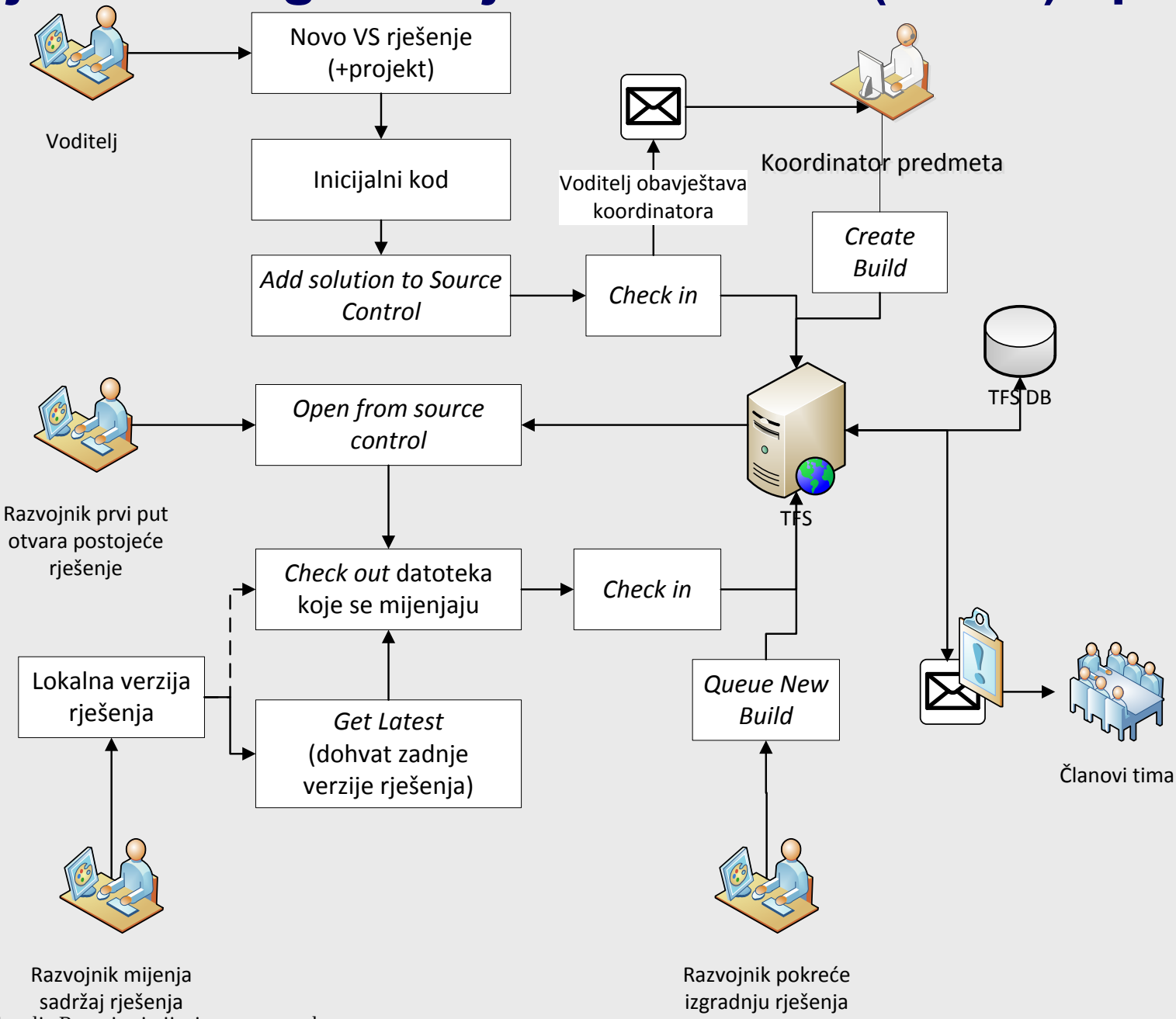
Rad s TFS-om - osnovni pojmovi (1)

- **Dodavanje rješenja na TFS**
  - *Solution Explorer* - desni klik na *solution* - Add to SourceControl (pogledati detaljnije u uputama)
- **Dodavanje pojedinačnih datoteka**
  - *Source Control Explorer* – desni klik – *Add Items To Folder*
  - Koristi se samo za datoteke koje nisu dio nekog projekta!
- **Get Latest/Specific Version**
  - dohvat zadnje/određene verzije s poslužitelja
- **Check Out for Edit**
  - najava namjere ažuriranja datoteka s TFS-a (promjena lokalne kopije)
  - opcionalno zaključavanje drugim korisnicima
- **Check In Pending Changes**
  - slanje izmijenjenih lokalnih datoteka na poslužitelj (potvrda promjena)
- **Pending Changes**
  - popis promjena koje čekaju na slanje na poslužitelj
- **Undo Pending Changes**
  - otkazivanje izmjena (uz otključavanje)
  - povratak na prethodnu verziju / verziju sa servera

Rad s TFS-om - osnovni pojmovi (2)

- **Changeset**
  - nedjeljiv skup promjena između dviju verzija u zajedničkom repozitoriju
  - 1 check-in = 1 changeset
- **View History**
  - pregled povijesti verzija
- **TFS Workspace**
  - postavke preslikavanja između mapa na TFS-u i mapa na pojedinom računalu nekog korisnika, popis još nepotvrđenih promjena i zaključavanja
- **Branch**
  - stvaranje nove kopije (paralelne grane, verzije) izvornog koda
- **Merge**
  - spajanje više grana izvornog koda ili više verzija jedne datoteke u slučaju konflikta (npr. kod istovremenog ažuriranja)
- **Shelve/Unshelve**
  - privremeno spremanje izmjena na TFS, ali bez *check-ina*. Dostupno i ostalim članovima ekipe, ali ne dovlači se s *Get Latest*
- **Cloak**
  - skrivanje/isključivanje iz dohvata s *Get Latest* jedne ili više mapa

Primjer timskog razvoja za zadaće i (među)ispite



Zadatak za vježbu

- **Proučiti upute o uspostavi i korištenju TFS (TFS-Upute)**

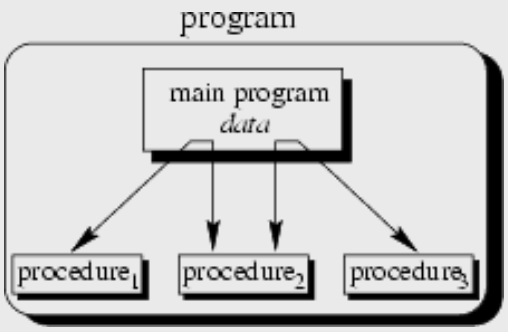
- **Kreirati zajedničko programsko rješenje ekipe (*solution*) na TFS**
  - Svaki član ekipe unutar toga kreira vlastiti projekt izvornog koda u koji dodaje barem jednu izvornu datoteku.
  - Isprobati osnovne funkcije, dodavanjem i izmjenom izvornih datoteka uz zaključavanje i otključavanje
  - U suradnji s koordinatorom predmeta isprobati automatsku izgradnju programskih rješenja
  - Isprobati rješavanje problema istovremenog ažuriranja datoteka.

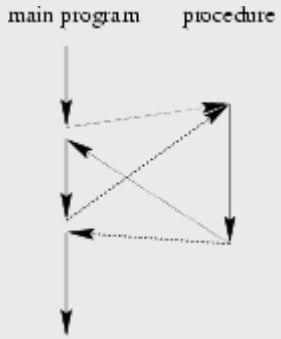
- **=> pod nadzorom izvođača na labosu u petak 20.3.2014. (16-20)**
  - Plan po grupama bit će objavljen u satnici na FER webu.

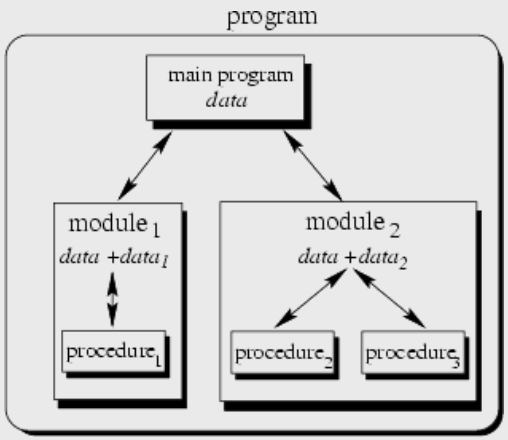
Objektno orijentirano programiranje

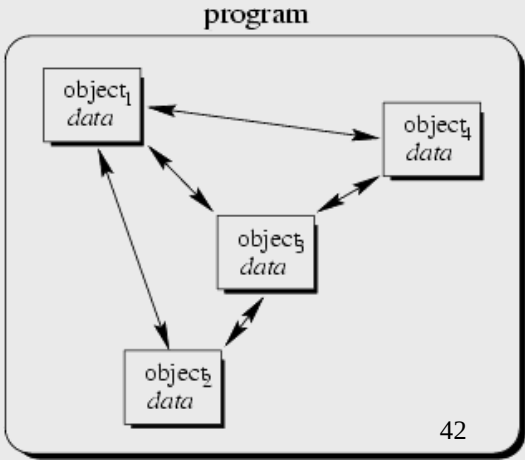
Tehnike programiranja

- **Nestrukturirano**
  - Programi koji se sastoje od slijeda naredbi i djeluju nad zajedničkim skupom podataka (globalnim varijablama)
  - Ponavljanje nekog posla znači i kopiranje naredbi.
- **Proceduralno**
  - Izdvajanjem naredbi u procedure,
  - Program postaje slijed poziva procedura.
- **Modularno**
  - Procedure zajedničke funkcionalnosti grupiraju se u module
  - Dijelovi komuniciraju pozivima, moduli mogu imati lokalne podatke
- **Objektno**
  - Objekti komuniciraju slanjem poruka









Dijelovi programa

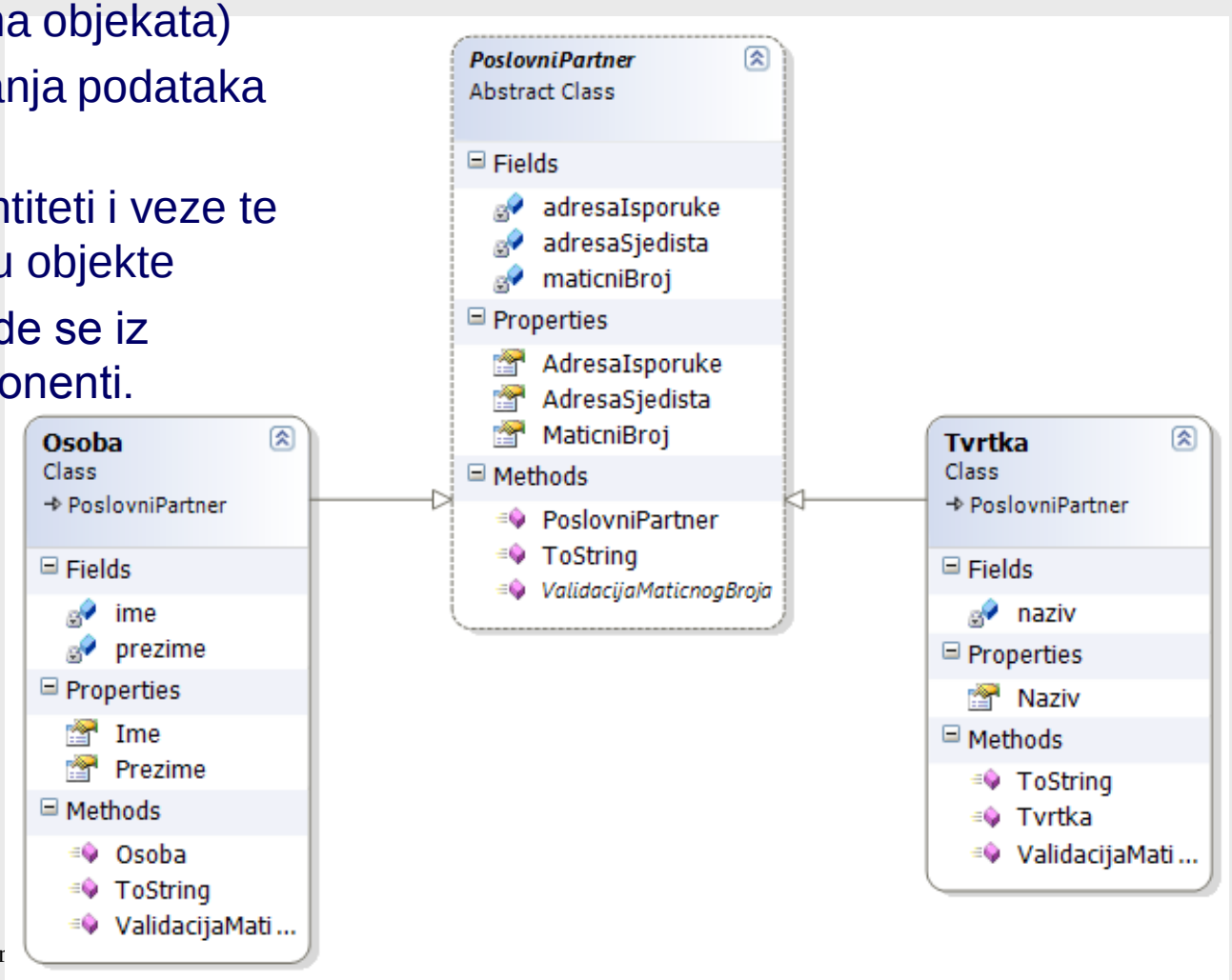
- **programska cjelina (program unit)**
  - skup programskih naredbi koje obavljaju jedan zadatak ili jedan dio zadatka, npr. glavni program, potprogrami (procedure, funkcije)

- **programski modul**
  - skup logički povezanih programskih cjelina $\rightarrow$ modularno programiranje
  - npr. C datoteka sa statičkim varijablama

- **komponenta**
  - bilo koji sastavni dio softvera, uobičajeno podrazumijeva fizičke cjeline
  - npr. assembly (skup, sklop)
    - DLL ili EXE (izvršna) datoteka uključujući pripadajuće resursne (datoteka.resx) ili sigurnosne elemente

Osnove objektno orijentiranog programiranja

- **Objektno usmjerena paradigma**
  - entiteti iz stvarnog svijeta opisuju se apstraktnim objektima (razredima, klasama objekata)
  - integracija oblikovanja podataka i procesa
  - tokovi podataka, entiteti i veze te procesi integrirani u objekte
  - složeni sustavi grade se iz pojedinačnih komponenti.



Koncepti objektnog pristupa

- **Objekt (Object)**
  - objekt - svaka pojava, pojam ili predmet (stvarni ili zamišljeni) o kojem se prate podaci i načini rukovanja podatcima (procesi)
  - pojava objekta (object instance) – pojedinačna pojava nekog tipa objekta, s vlastitim stanjem i ponašanjem
- **Apstrakcija (Abstraction)**
  - Obuhvat glavnih aspekata uz ispuštanje nevažnih detalja
- **Učahurivanje (Encapsulation)**
  - skrivanje ugradnje od korisnika objekta
  - objekti su “crne kutije” čijim se podacima rukuje putem ugrađenih metoda
- **Modularnost (Modularity)**
  - Razlaganje složenih sustava u (lakše) upravljive dijelove.
- **Hijerarhija (Hierarchy)**
  - Hijerarhijsko uređivanje apstrakcija

Koncepti objektnog pristupa (2)

- **Razred (Class)**
  - tip objekta (abstract data type) – generalizacija grupe objekata, zajednički naziv za sve objekte sa zajedničkim svojstvima (property) i načinima rukovanja (process, method), npr. Proizvod
  - opis grupe objekata koji imaju jednake atribute, operacije, veze i semantiku
  - objekt je instanca razreda, s vlastitim vrijednostima svojstava
  - razred je apstraktna definicija objekta
- **Svojstvo (Property), atribut (Attribute)**
  - definira skup vrijednosti koje instanca može poprimiti za neku karakteristiku
  - može biti nekog tipa, npr. integer, Boolean, Adresa
- **Postupak (Method), Operacija (Operation)**
  - implementacija usluge koja može biti zatražena od objekta razreda
  - ima prototip, „potpis” (naziv, parametre, povratnu vrijednost)
- **Višeobličje (Polymorphism)**
  - mogućnost "skrivanja" različitih ugradnji iza zajedničkog sučelja
  - postupci koje objekti različitih razreda tumače zavisno o specijalizaciji

Koncepti objektnog pristupa (3)

- **Nasljeđivanje (Inheritance)**
  - podrazred (subclass) nasljeđuje svojstva i metode nadrazreda (superclass), npr. Igla i Avion nasljeđuju Proizvod
- **Podsustav (Subsystem)**
  - kombinacija paketa razreda
  - realizacija jednog ili više sučelja koje definiraju ponašanje razreda
- **Komponenta (Component)**
  - nezavisan, zamjenjiv, ne-trivijalan dio sustava
  - može biti izvorni kod, pogonska inačica ili izvršni program
- **Veza, asocijacija (Association)**
  - smislena poveznica između dva ili više dijelova

Razredi

Razredi i članovi razreda

- **U jeziku C# razredi se sastoje od članova (members):**
  - Konstante (constants)
  - Atributi (fields)
  - Konstruktori (constructors)
  - Finalizatori (finalizers)
  - Postupci (methods)
  - Svojstva (properties)
  - Indekseri (indexers)
  - Operatori (operators)
  - Događaji (events)
  - Statički konstruktori (static constructors)
  - Ugniježđeni tipovi (nested types)
- **Razred je obrazac prema kojem se stvaraju instance razreda - objekti**
  - `new` – operator koji stvara novu instancu
  - `this` – referenca na trenutnu instancu razreda
  - razred s početnim postupkom (`Main`) ne mora se eksplicitno instancirati

Modifikatori tipova i članova

- **Modifikatori pristupa razredima i članovima**
  - `public` – pristup nije ograničen
  - `private` - pristup ograničen na razred u kojem je član definiran
  - `protected` – pristup ograničen na razred i naslijeđene razrede
  - `internal` – pristup ograničen na program u kojem je razred definiran
  - `protected internal` - pristup dozvoljen naslijeđenim razredima (bez obzira gdje su definirani) i svima iz programa u kojem je razred definiran
- **Neki od značajnijih modifikatora**
  - `abstract` – razred može biti samo osnovni razred koji će drugi nasljeđivati
  - `const` – atribut (polja) ili lokalna varijabla je konstanta
  - `new` – modifikator koji skriva naslijeđenog člana od člana osnovnog razreda
  - `readonly` – polje poprima vrijednost samo u deklaraciji ili pri instanciranju
  - `sealed` – razred ne može biti naslijeđen
  - `static` – jedini, zajednički član svih instanci razreda (ne kopija nastala s instancom)
  - `virtual` – postupak ili dostupni član koji može biti nadjačan u naslijeđenom razredu – (prilikom nadjačavanja dodaje se modifikator `override`)

Pretpostavljeni modifikatori

- **Ako modifikator nije naveden onda se smatra da je**
  - `internal` za razrede, sučelja, delegate i događaje
  - `private` za članske varijable, svojstva i postupke i ugniježđene razrede
  - `public` za postupke sučelja i članove enumeracija (nije ni dozvoljen drugačiji modifikator)
- **Izvedeni razred ne može imati veću dostupnost od baznog razreda**

Statički članove, konstante i atributi (fields)

- **`Static` se može primijeniti na atribute, postupke, svojstva, operatore i konstruktore**
  - Statički član pripada tipu, a ne instanci.
- **Modifikator `const`**
  - Vrijednost konstante određena je pri prevođenju i ne može se mijenjati
  - Konstante su po prirodi statičke, pa ih se ne može eksplicitno proglasiti statičkim ni pristupati im preko reference na instancu objekta
- **Atributi**
  - Varijable kojima se može promijeniti vrijednost pristupom članu
  - Nepromjenjivi atributi – modifikator `readonly`
    - ne dopuštamo promjene, a vrijednosti nisu poznate u vrijeme prevođenja
- **Primjer 🗁Razredi\Razred**

```
public int var = 0; // atribut, varijabla objekta
public const int konst = 13; // konstanta razreda
public readonly int neDiraj; // postavlja se u konstruktoru
public static int trajna = 0; // dijeljena između instanci
```

Konstruktori

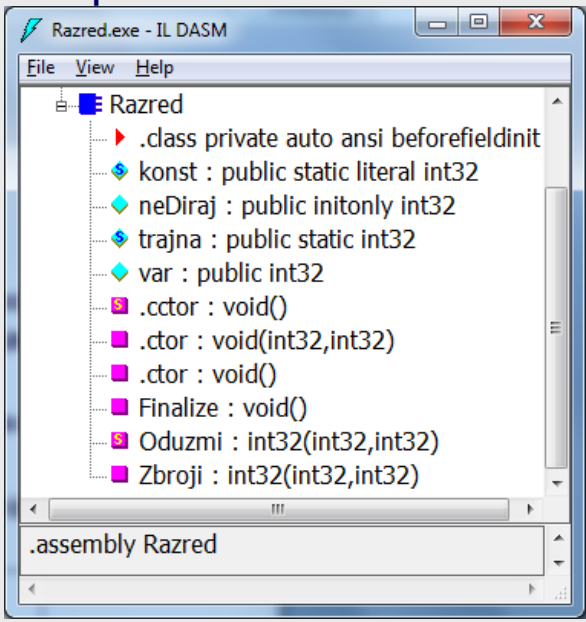
- **Konstruktor je postupak koji se obavlja pri stvaranju instance**
  - Standardni (default) konstruktor i preopterećenje konstruktora s argumentima
  - Ne vraća vrijednost, služi za inicijalizaciju instance
  - Ime mu je jednako imenu razreda
- **Primjer 🗁Razredi\Razred**

```
class Razred
{
  //atributi
  public readonly int neDiraj;
  public int var = 0;
...
  //konstruktori
  public Razred(){
    neDiraj = 0;
  }
  public Razred(int neDiraj0, int var0)
  { neDiraj = neDiraj0;
    var = var0;
  }
```

```
Razred r1 =
  new Razred();
Razred r2 =
  new Razred(13,666);
Razred r2 =
  new Razred() {var=4};
```

Finalizatori (Destruktori)

- **Finalizator je postupak (Finalize) koji se automatski poziva neposredno prije uništenja objekta od strane sakupljača smeća (Garbage Collector).**
  - Piše se u obliku destruktora - jednak je nazivu razreda s predznakom '~'
  - Ne vraća vrijednost
  - Koristi se za brisanje tzv. "unmanaged" resursa (ako su korišteni)



- **Primjer 🗁Razredi\Razred**

```
//finalizator se piše u obliku destruktora
  ~Razred(){
    Console.WriteLine("Finalizer");
  }
}
```

- **Sakupljač smeća (garbage Collector)**
  - Oslobađa memoriju koja je bila zauzeta referencama koje više ne postoje.
  - Sakupljanje smeća obavlja tijekom izvođenja programa.
  - Ne može se unaprijed točno odrediti kada će se memorija osloboditi, to jest kada će se finalizator pozvati, ali ga se može potaknuti (metodom Collect).

Postupci (metode)

- **Postupak je programska funkcija pridružena razredu.**
  - Skrivanje člana (učahurivanje - enkapsulacija)
    - Pristup skrivenom članu javnim postupcima
- **Primjer: 🗁Razredi\Postupci**

```
class Postupak{
  private int brOp = 0; // brojač pozvanih operacija
  public int Zbroji(int x, int y) {
    ++brOp; return x + y;
  }
  public int Razlika (int x, int y) { ++brOp; return x-y; }
  public int GetBrOp() { return brOp; }
...
```

```
Postupak p = new Postupak();
int zbroj = p.Zbroji(13, 15);
int razlika = p.Razlika(13, 15);
int brojOperacija =  p.GetBrOp();
```

Postupci (2)

- **Argumenti:**
  - standardno, bez modifikatora – "call by value“

```
public static void SwapByVal (int a, int b);
SwapByVal (a, b);
```

  - ref modifikator – argumenti su reference ("call by reference")
    - Prije poziva postupka argumenti MORAJU biti inicijalizirani
    - ref parametru argument mora eksplicitno biti predan navođenjem ref.

```
public static void SwapByRef (ref int a, ref int b);
SwapByRef (ref a, ref b);
```

  - out modifikator – izlazni argumenti
    - U trenutku poziva out postupka argumenti ne moraju biti inicijalizirani.
    - Pri izlasku iz postupka out argumenti MORAJU biti postavljeni.

```
public static void TestOut(out int val, int defval);
TestOut(out i, 13);
```

- **Primjer 🗁Razredi\Postupci**

Postupci (3)

- **Prijenos varijabilnog broja argumenata:**
  - Ključna riječ `params`
    - koristi se kad broj argumenata nije unaprijed poznat
    - polje argumenata može biti bilo kojeg tipa
- **Primjer: 🗁Razredi\Postupci**

```
public static void TestParams(params object[] args){
    Console.WriteLine("Params: ");
    for(int i= 0; i< args.Length; i++)
      Console.WriteLine("{0}:{1}", i, args[i])
}
```

```
TestParams("jedan", "dva", 3);
TestParams();
```

- **`Main` je postupak koji preuzima argumente pri pozivu programa**
  - `static void Main( string[] args )`
  - Primjer `TestParams(args);`

Postupci (4)

- **Opcionalni argumenti kojima se zadaje pretpostavljena vrijednost**
  - Navode se zadnji po redu
- **Primjer: 🗁Razredi\Postupci**

```
public static void TestDefault(int a, int b = 99999,
       string s = "default string") {
 Console.WriteLine("a = {0}, b = {1}, s = {2}", a, b, s);
}
```

```
TestDefault(1, 2, "RPPP");
TestDefault(2, 15);
TestDefault(3);
```

- **Imenovani argumenti:**
  - Priliko poziva navodi se naziv argumenta i vrijednost odvojeni dvotočkom
  - Imenovani argumenti se navode zadnji

```
TestDefault(2, s:"moj string");
TestDefault(s: "moj string", b:55, a:4);
```

Svojstva (properties) (1)

- **Svojstvo je postupak pristupa zaštićenim varijablama instance**
- **Primjer 🗁Razredi\Postupci\Postupak_v2.cs**
  - umjesto postupka za pristup skrivenom članu koristi se tzv. Svojstvo

```
class Postupak_v2{
  public int BrOp {
      get { //dohvat vrijednosti člana
        return brOp;
      }
      //set { //omogućava promjenu vrijednosti
      //  brOp = value;
      //}
   }
```

```
Postupak_v2 p2 = new Postupak_v2();
zbroj = p2.Zbroji(a, b);
razlika = p2.Oduzmi(a, b);
brojOperacija = p2.BrOp;
```

Svojstva (2)

- **Nije nužno da svojstvo samo vraća ili postavlja vrijednost varijabli**
  - Može sadržavati i složeniji kod
  - Primjer 🗁Razredi\SvojstvaIndekseri

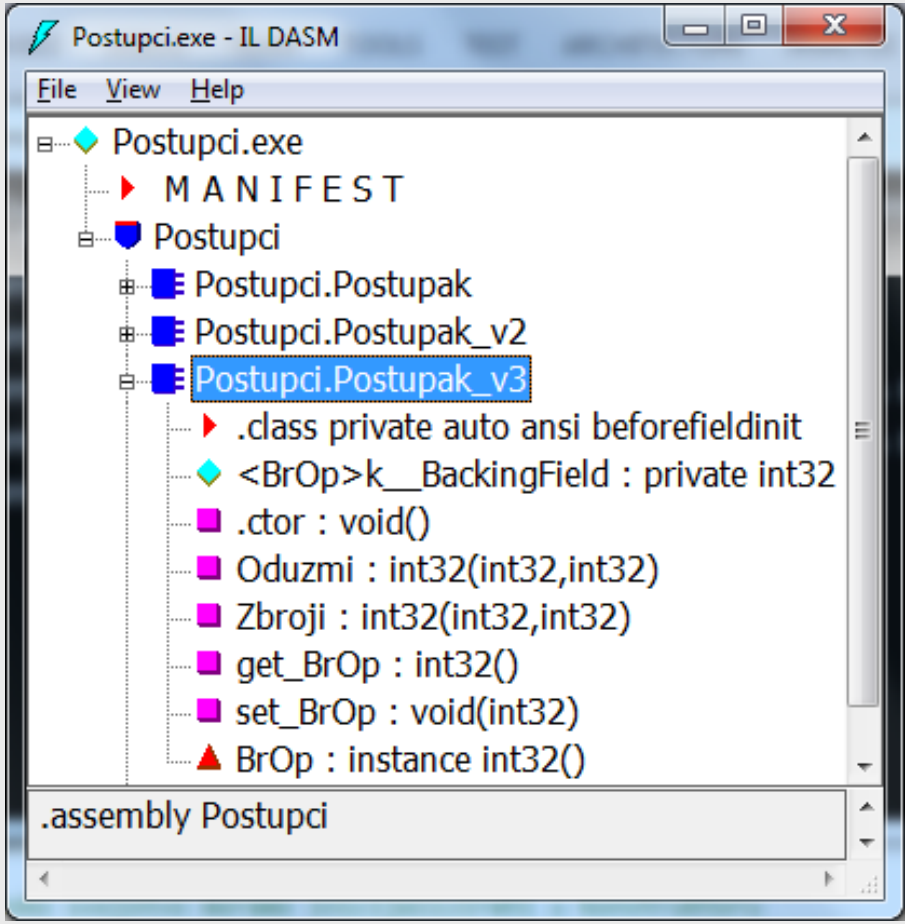
```
class Temperatura{
  private float T;
  public float Celsius{
    get { return T - 273.16f; }
    set { T = value + 273.16f; }
  }
  public float Fahrenheit{
    get { return 9f / 5 * Celsius + 32; }
    set { Celsius = (5f / 9) * (value - 32); }
  }
}
```

```
Temperatura X = new Temperatura();
X.Fahrenheit = 70;
Console.WriteLine("{0} = {1}", X.Fahrenheit, X.Celsius);
```

Automatska svojstva

- Koristi se u slučaju kad svojstvo služi samo kao omotač oko privatne varijable
- Može se postaviti modifikator pristupa (npr. private) za get i/ili set
- Interno se stvara varijabla za pohranu i kod za dohvat i pridruživanje
- Početna vrijednost se postavlja u konstruktoru
- Primjer 🗁Razredi \ Postupci \ Postupak_v3.cs

```
class Postupak_v3  {
   public int BrOp {
     get;
     private set;
   }
   public Postupak_v3() {
            BrOp = 0;
   }
   ...
```



Indekseri

- **Indekseri**
  - omogućavaju korištenje objekta kao niza (pristup elementima operatorom [ ])
  - sintaksa uobičajena za svojstva (get i set)
- **Primjer: 🗁Razredi\SvojstvaIndekseri**

```
public Temperatura this[int index]   // Indekser
{
  set{
    if (index >= 0 && index < nizTemperatura.Length)
      nizTemperatura[index] = value;
    else throw new Exception("Pogreška!");
  }
  get{
    if (index >= 0 && index < nizTemperatura.Length)
      return nizTemperatura[index];
    else throw new Exception("Pogreška!");
  }
}
```

Preopterećenje postupaka (method overloading)

- **Postupci jednakog imena definirani u istom razredu**
  - moraju imati različitu signaturu (prototip), tj. različiti tip ili redoslijed argumenata
  - uobičajeno obavljaju isti posao – nad različitim tipovima podataka
- **Primjer: 🗁Razredi\PreopterecenjePostupaka**

```
static double Maximum(double x, double y, double z){
  Console.WriteLine("double Maximum( double x, double y,
   double z )");
   return Math.Max(x, Math.Max(y, z));
}
static int Maximum(int x, int y, int z){
   Console.WriteLine("int Maximum( int x, int y, int z )");
   return Math.Max(x, Math.Max(y, z));
}
```

```
// koji Maximum se poziva (double ili int)?
Console.WriteLine("maximum1: " + Maximum(1, 3, 2));
Console.WriteLine("maximum2: " + Maximum(1.0, 3, 2));
```

Preopterećenje operatora (operator overloading)

- **Operatori koji se mogu preopteretiti**
  - unarni: + - ! ~ ++ -- true false
  - binarni: + - * / % & | ^ << >> == != > < >= <=

- **Primjer: 🗁Razredi\PreopterecenjeOperatora**

```
public static ComplexNumber operator +(ComplexNumber x,
  ComplexNumber y){
   return new ComplexNumber(x.Re + y.Re, x.Im + y.Im);
}
public static ComplexNumber operator *(ComplexNumber x,
  ComplexNumber y){
   return new ComplexNumber(x.Re * y.Re - x.Im * y.Im, x.Re *
        y.Im + y.Re * x.Im);
}
```

```
Console.WriteLine(x + " + " + y + " = " + (x + y));
Console.WriteLine(x + " - " + y + " = " + (x - y));
Console.WriteLine(x + " * " + y + " = " + (x * y));
```

Zadaci za vježbu

- **Implementirati razred `Planet`**
  - Konstruktor neka prima radijus, gravitaciju i ime planeta.
  - Razred sadrži i statičku varijablu za brojanje instanciranih planeta, svojstva `Ime, Radijus, Gravitacija` i postupak `GetCount()`.
  - Demonstrirati rad implementiranog razreda.
- **Implementirati razred `PlanetCollection`**
  - Razred predstavlja listu planeta.
  - Iskoristiti razred iz prethodnog zadatka s planetima.
  - Napisati postupak `Add` za dodavanje planeta.
  - Planetima se može pristupati preko indeksa.
- **Implementirati razred `Nebo` koji**
  - sadrži `Planet` i njegovu udaljenost od njegovog sunca.
  - Dodati postupak sortiranja planeta po udaljenosti od sunca,
  - te postupak za računanje “udaljenosti” dvaju planeta.

Dodatni materijali, reference

- **Objects, Classes, and Structs (C# Programming Guide)**
  - Objects, Classes, Structs, Inheritance, ...
  - http://msdn.microsoft.com/en-us/library/ms173109.aspx
  - http://msdn.microsoft.com/en-us/library/ms173121.aspx

- **Upravljanje projektima**
  - http://www.pmi.org
  - http://www.pmi-croatia.hr