KOPONENTET DHE OBJEKTET

KAPITULLI 3

Prof. Ass. Dr. Isak Shabani

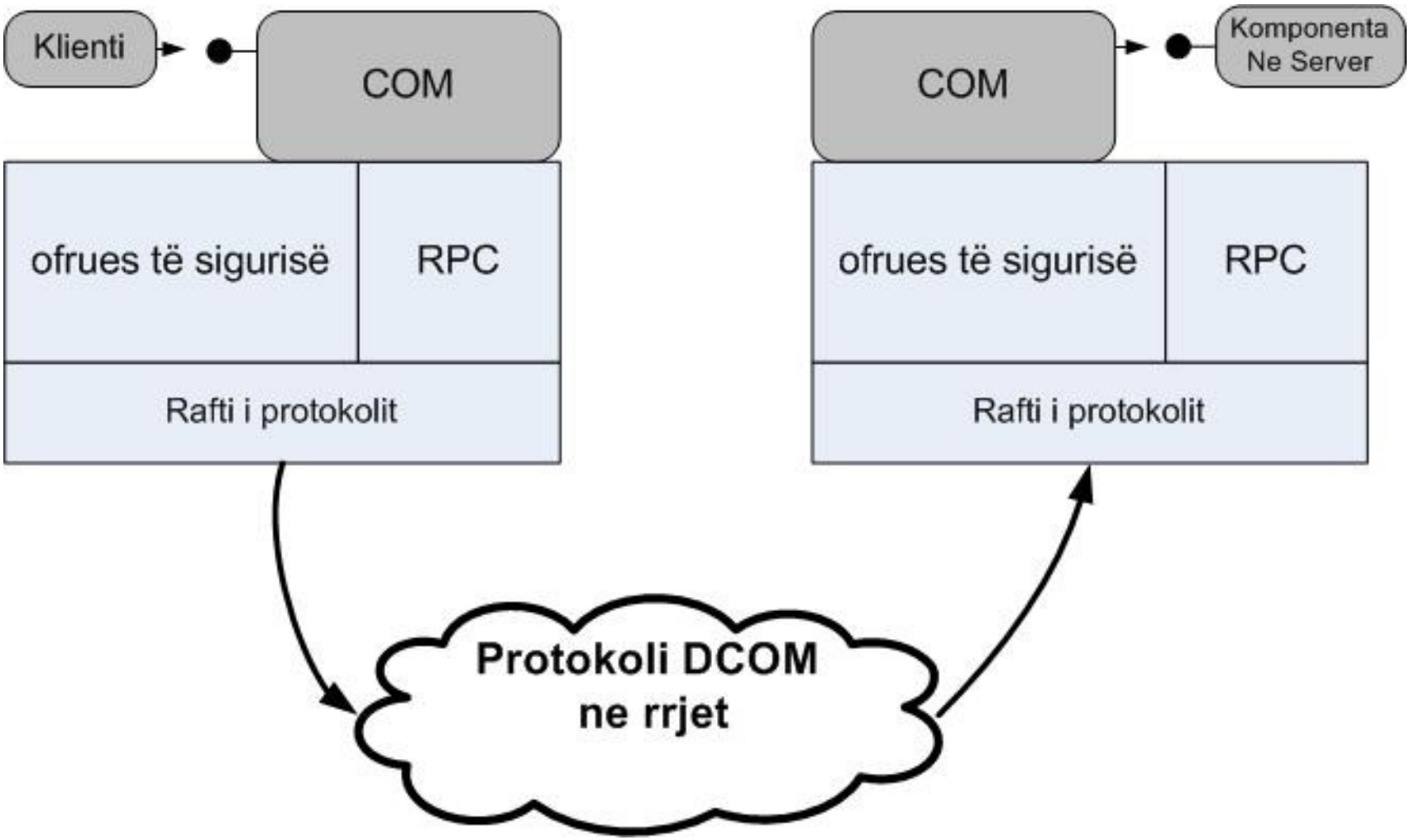
Komponentet: Component Object Model (COM dhe COM+)

- Numri i madh i gjuhëve të programimit ka të mirat e saj por problemi që paraqet për programeret është se e ndan tregun e komponentëve (programeve apo objekteve) të ripërdorueshme.
- P.sh një Klase në C++ nuk i hyn në punë për një programues në Java.
- Kodi i shkruar ne Visual Basic nuk do të ndihmonte një programues në Delphi apo COBOL.
- Probleme te tilla i bën që Component Object Model (COM) te sjellin një zgjidhje.
- Kjo zgjidhje ofron përdorimin e komponentëve (përbërësve) binare.
- COM është një specifikim dhe një bashkësi shërbimesh që bën të mundur krijimin e programeve modulare, të orientuara me objekte, lehtësisht të ndryshueshme dhe të përmirësueshme, dhe që mund të bashkëveprojnë nëpërmjet një rrjeti kompjuterësh (Network).
- Komponentët e COM-it mund të paketohen si DLL apo ne EXE (procese të ekzekutueshme) dhe COM siguron mekanizmin e komunikimit për të lejuar komponentët në module të ndryshme për të komunikuar me njeri tjetrin.
- Në COM+ të gjitha komponentët paketohen në DLL .
- COM ka një aftësi për ripërdorimin e komponentëve në skenarët lokal, por nuk është i dizajnuar që të mund të punojnë në mënyrë efikase me komponentët në distancë. Për më tepër me rritjen e kërkesave për rrjeta heterogjene dhe nevoja e aplikacione të shpërndara.
- Microsoft ka zhvilluar Modelin e Komponentëve te Objekteve të shpërndara(DCOM).

Komponenta DCOM

- DCOM është një forme e zgjeruar e COM-it.
- Qëllimi kryesor është mbështetja e ndërveprimit dhe ripërdorimi nën platformën e windowsit për komponentët e shpërndara.
- Në COM, proceset do të ekzekutohen në makinën e vetme ose të njëjtë, ndërsa
- Teknologjia DCOM është projektuar për të ekzekutuar proceset nëpër rrjetet heterogjene.
- Teknologjia DCOM është një lidhje e ngushtë me teknologjitë e tjera të Microsoft si OLE (Object Linked Embeded) dhe ActiveX.
- Janë disa arsye kryesore për përdorim të teknologjisë DCOM:
  - Së pari DCOM është një arkitekturë e cila ndërton aplikimet shpërndara.
  - Së dyti shërbimet e DCOM sigurojë një mjedis të sigurt, në dispozicion, të besueshme dhe të lartë të performancës.
  - Pra DCOM merr të gjithë përgjegjësinë e komunikimit mes makinave.
  - Arkitektura e DCOM ofron një sërë shërbimesh kryesore për të ndërtuar sisteme apo aplikacione të shpërndara.
  - Ndërfaqja e COM-it është e përdorur për të hyrë në të gjitha shërbimet e DCOM-it.

Arkitektura e teknologjise DCOM



Siguria e DCOM

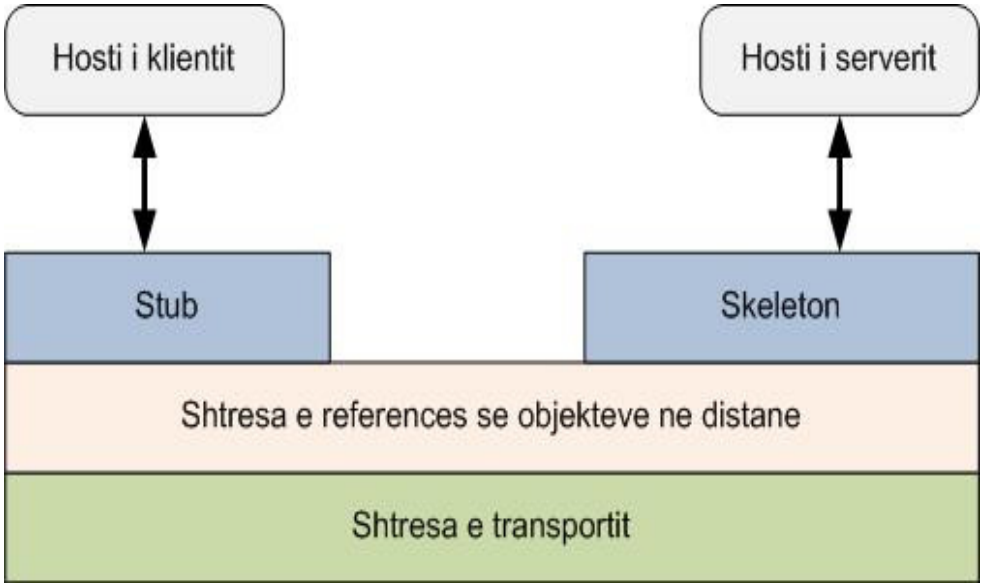
- DCOM ofron katër aspektet kryesore të sigurisë, të cilat janë:
  - *Siguria e qasjes*: Ofron mekanizimin e sigurisë në nivelin e procesit të kontrolli i qasjes, sipas sistemit operativ.
  - *Siguria fillestare*: ne kete rast DCOM kontrollon sigurinë e kredencialeve para qasjes së objektit ose një para se të thirret ndonjë metodë. Në qoftë se nuk ka kredencialet atëherë kërkesa thjesht refuzohet.
  - *Identiteti*: DCOM parashikon që thirrësi duhet të identifikojë se çfarë objekti është i lejuar të thirret kur ajo mban shenjë të sigurisë. *Rregullat e lidhjes*: Rregullat e lidhjes kanë të bëjnë me vërtetimin dhe mbrojtjen e të dhënave të transmetuara në rrjetin.

Metoda e thirrjeve ne distancë (RMI)

- Remote Method Invocation(RMI) është model i objekteve te shpërndara si COM, DCOM dhe CORBA.
- RMI jep përkrahje për zhvillimin e aplikacioneve të shpërndara në Java.
- Java virtual makinat (JVM) janë përdorur për të thirrur metodat e objektit në distancë.
- Në RMI klienti përdorë ndërfaqet në distancë për të thirrur metodat.
- RMI mbështet disa karakteristika të cilat Ueb shërbimet nuk i mbështetin të tilla si:
  - referencat e objekteve,
  - shkarkim të klasës dinamike dhe
  - mbështetje për mbledhja e mbeturinave të shpërndara.

Arkitektura e RMI

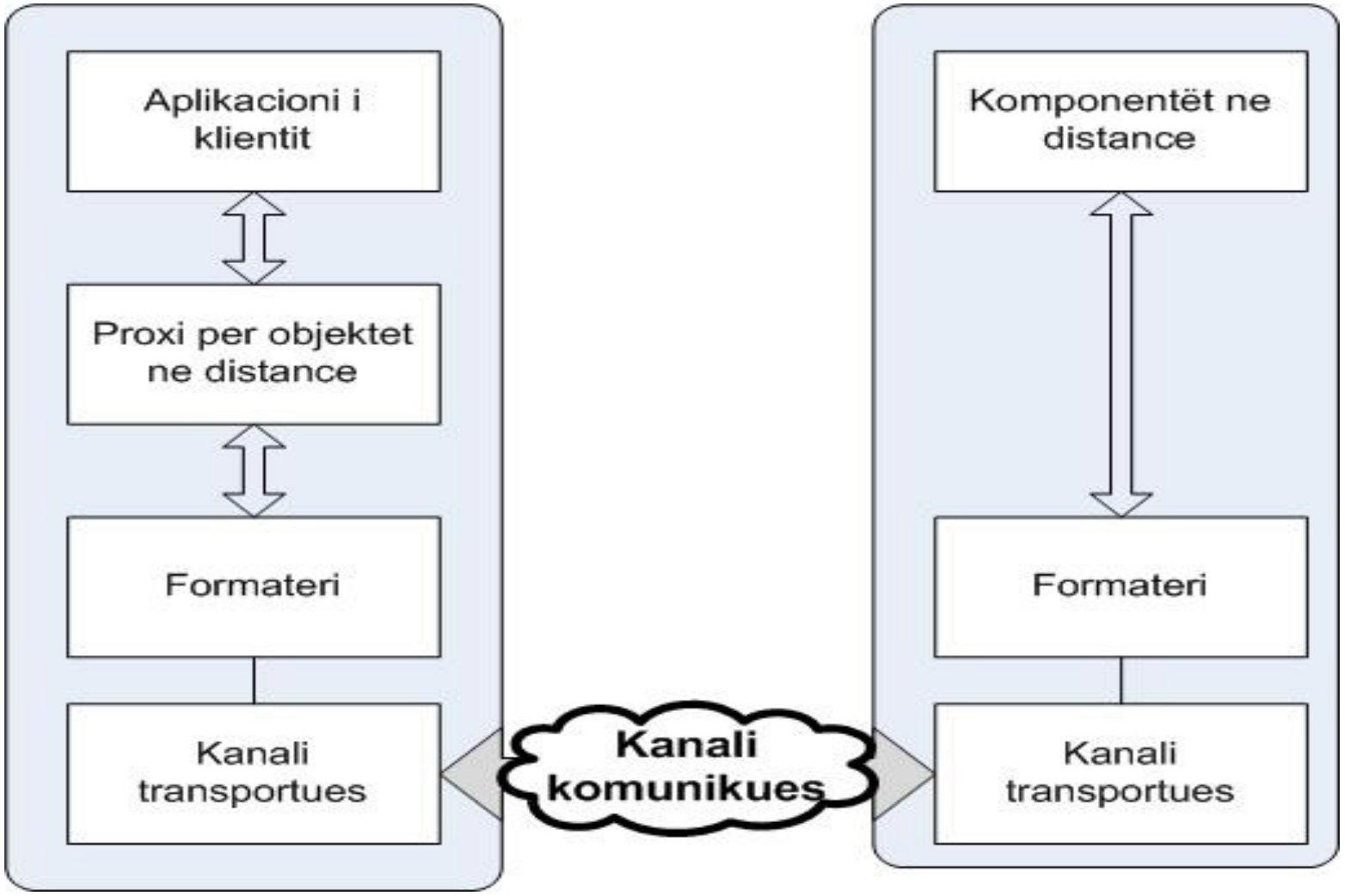
- Arkitektura RMI ofron një strukturë të plotë për objektet e orientuar në kompjuter të shpërndarë.
- Korniza e RMI midlleware-it qëndron në mes të sistemit operativ dhe atij aplikativ në çdo anë te sistemit.
- Java RMI përbëhet nga tri shtresa kryesore, të cilat janë:
  - shtresë stub / skelet,
  - shtresa e referencës dhe
  - shtresa e transportit.



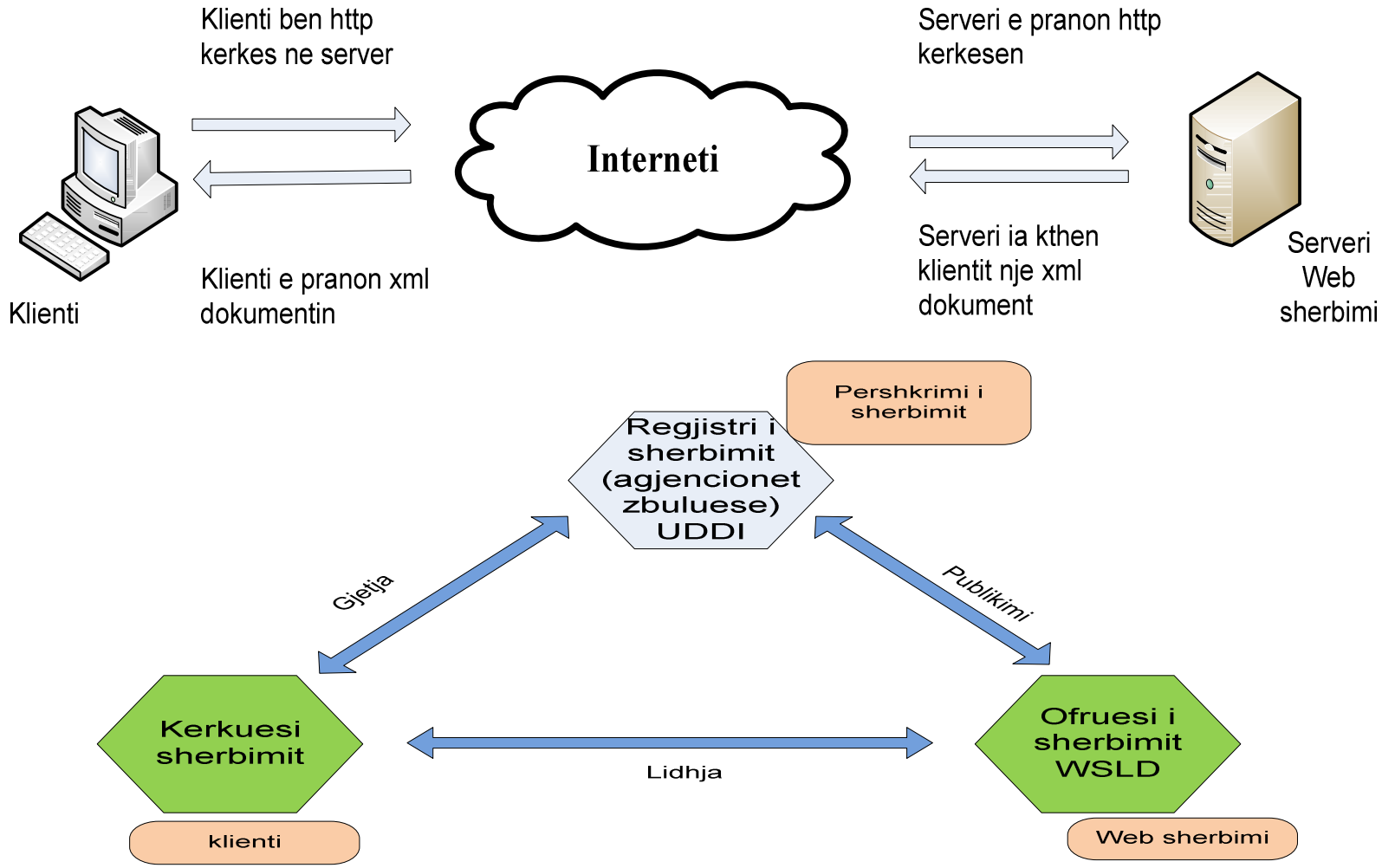
Teknologjia e Mikrosoftit .NET Remoting

- Teknologjia .Net Remoting është pjesë e Platformës së Microsoft .NET-it.
- Platforma .NET siguron një grup të madh të klasave, Strukturave, hapësira, ndërfaqeve dhe delegatëve për zhvilluesit.
- Teknologjia .NET Remoting nuk siguron vetëm ndërveprimin me Microsoft COM+ , por gjithashtu përfshin edhe teknologjinë RPC për nder-komunikimit.
- Platforma përdor gjuhën e përbashkët (CLR) si një makinë virtuale.
- Korniza e .Net Remoting ofron shërbime të ndryshme të tilla si e aktivizimit, mbështetje të jetëgjatësisë dhe kanali i komunikimit.
- Zhvilluesit e përdorin kornizën .NET Remoting për të ndërtuar aplikacionet e klientit që përdorin proceset e të njëjtit kompjuter ose ndonjë kompjuteri mbi rrjetë.
- Aplikacionet e klientit në .NET Remoting lejojë **përdorimin e komponentëve** në kompjuterët e largët dhe të trajtimin e tyre si komponent lokale.
- Nga perspektiva e komponentëve të orientuar, komponenti .NET është zëvendësimi i COM dhe DCOM komponentëve.
- Komunikimi mes **komponentëve të largëta** dhe **lokale** bëhen të mundura nga shtresa **proxy**.

Arkitektura e Microsoft .NET Remoting



Ueb Sherbimet



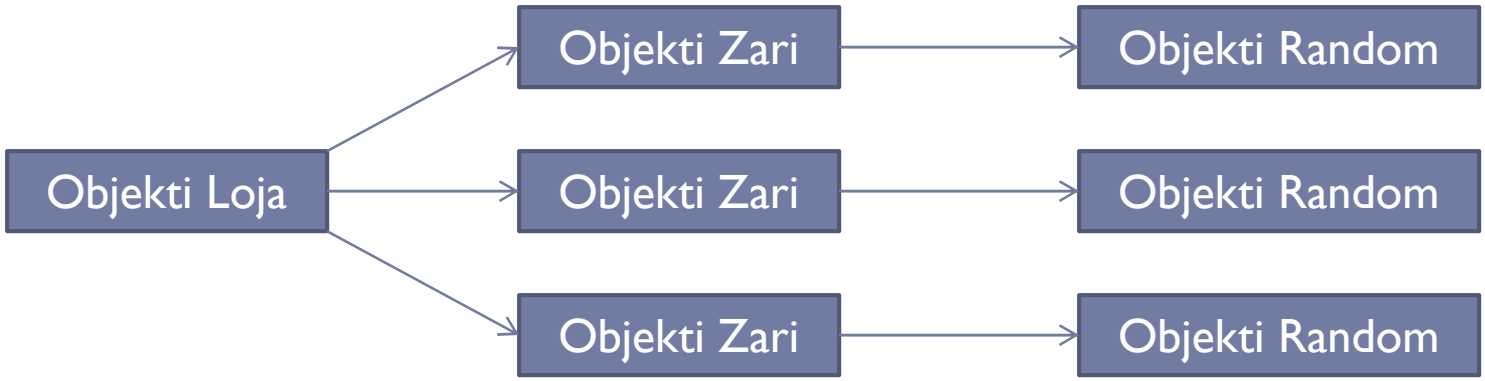
Klientët dhe Serverët

Në lojen me zari, objektet luajn rolin e *klientit* dhe të *serverit*

- Klientët dhe serverët
  - Objekti **Loja** është klient i disa objekteve **Zari**
  - Një objekt i dhënë **Zari** është klient i objektit **Random**
  - Objekti **Zari** luan rolin e një serveri tek objekti **Loja**, dhe objekti **Random** luan rolin e serverit tek objekti **Zari**

Dërgimi i Mesazheve

Në mënyrë metaforike, ne pretendojmë që objektet komunikojnë përmes *dërgimit të mesazheve*



- Dërgimi i mesazheve
  - Ne shpesh pëlqejmë të mendojmë interaksionin në mes të objekteve si shkëmbim të mesazheve
  - Marrësi i objektit e lokalizon proceduren ose funksionin i cili mund të i përgjigjet mesazhit - *shikimi i metodës*
  - Rezultati mund të kthehet prapa nga marrësi tek dërguesi i mesazhit

Objektet

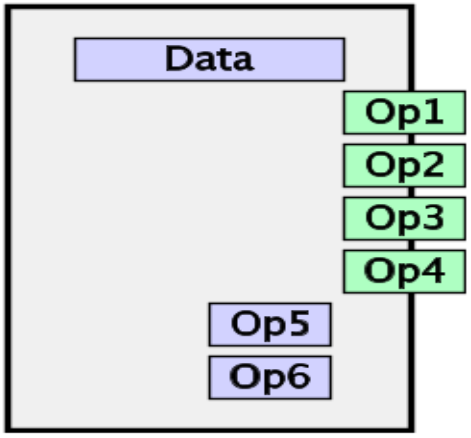
- *Nje objekt enkapsulon të dhënat dhe operacionet që manipulojnë këto të dhëna.*
  - P.sh një objekt Student, mund të konsistojë në të dhëna të tilla si: emri, mbiemri, datëlindja, adresa, tel, mosha dhe operacione që përcaktojnë dhe ndryshojnë këto të dhëna.

- *Një objekt ka:*
  - *gjendje(të dhënat e brendshme),*
  - *sjellje(metodat) dhe*
  - *identitet (ka një adresë unike në memorje).*

Klasat

- Një klasë duhet të përcaktohet përpara se të mund të krijoni një instancë (objekt) të kësaj klase.
- Një klasë është një lloj modeli që tregon cfarë lloj objektesh mund apo nuk mund të krijohen.
- Shohim klasën si sinonim i "tipit".

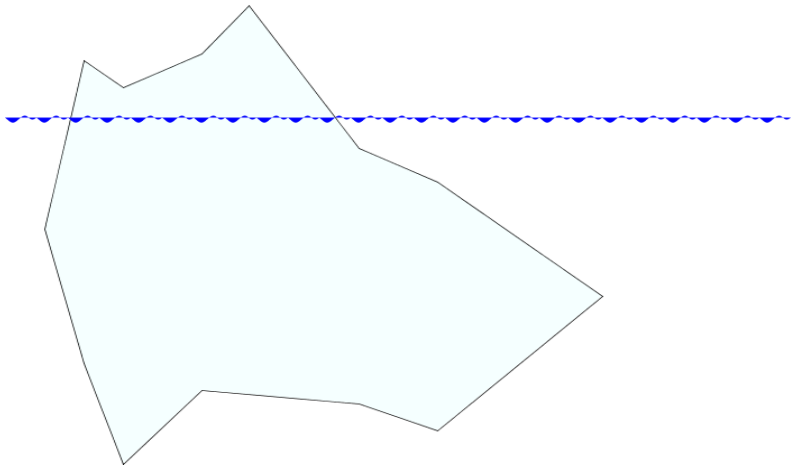
Klasa enkapsulon të dhënat dhe operacionet që përkasin me njëra tjetrën, dhe kontrollon dukshmërinë e të dyjave të dhënave dhe operacioneve.
Klasa mund të përdoret si tip në gjuhën programuëse.



Pjesa e dukshme e klasës e përbën *ndërfaqen* e klasës, ashtu si shihet nga klasat klientë

Dukshmëria – Analogjia me akullin

Klasa mund të shihet si një akull: Vetëm një pjesë e vogël e saj duhet të shihet nga jashta. Shumica e detaleve të klasës duhet të fshihen.



Klientët e klasës C nuk mund të varen direkt nga pjesët e fshehura të klasës C. Kështu, pjesët e fshehta në C mund të ndryshohen më lehtë sesa pjesët të cilat e përbëjnë ndërfaqen e klasës.

Aspektet e Dukshmërisë dhe Fshehjes

Cilat aspekte të klasës janë të fshehura, dhe cilat aspekte janë të dukshme nga jashta klasës?

- Aspektet e dukshme
  - Emri i klasës
  - Nënshkrimi i operacioneve të zgjedhura: ndërfaqia e klasës
- Aspektet e fshehura
  - Përfaqësimi i të dhënave
  - Trupat e operacioneve
  - Operacionet që shërbejnë vetëm si ndihmës të operacioneve të tjera

Modifikimi i programit – analogjia me zjarrin

Një modifikim i vogël në program mund të shpërndahet në tërë programin sikurse flaka në tërë shtëpinë.
Zjarri i tillë mund të shumicën e programit në sens të asaj se shumica e pjesëve të programit do të duhej të riprogramoheshin.

Përdorimi i mureve të zjarrta e pengon përhajen e flakës.
Ngjashëm, enkapsulimi dhe kontrolli i dukshmërisë është preventivë e modifikimeve në program që të mos kenë pasoja globale

Klasat në C#

Në C#
Modifikatorët e klasës **class** emri-klasës {
deklarimi-variablave, definimi-konstruktoreve definimi-metodave }

- **Anëtarët:** përbërësit e klasës quhen anëtarë
  - Variablat, konstruktorët, metodat
  - Anëtarët e tjerë të klasave do të diskutohen gjatë ligjëratave të ardhshme
- **Dukshmëria**: Anëtarët mund të jenë privat ose publik
  - Dukshmëria me paracaktim është private
  - *Mos u mbështet në dukshmërinë e paracaktuar!*
- **Rënditja:** nuk ka ndonjë përshkrim të rënditjes në mes anëtarëve
  - Një stil i dhënë i kodimit rekomandon rënditje të caktuar të tyre.

Vështrimi i anëtarëve të klasës

• Anëtarët më të rëndësishëm të klasës janë variablat dhe metodat.

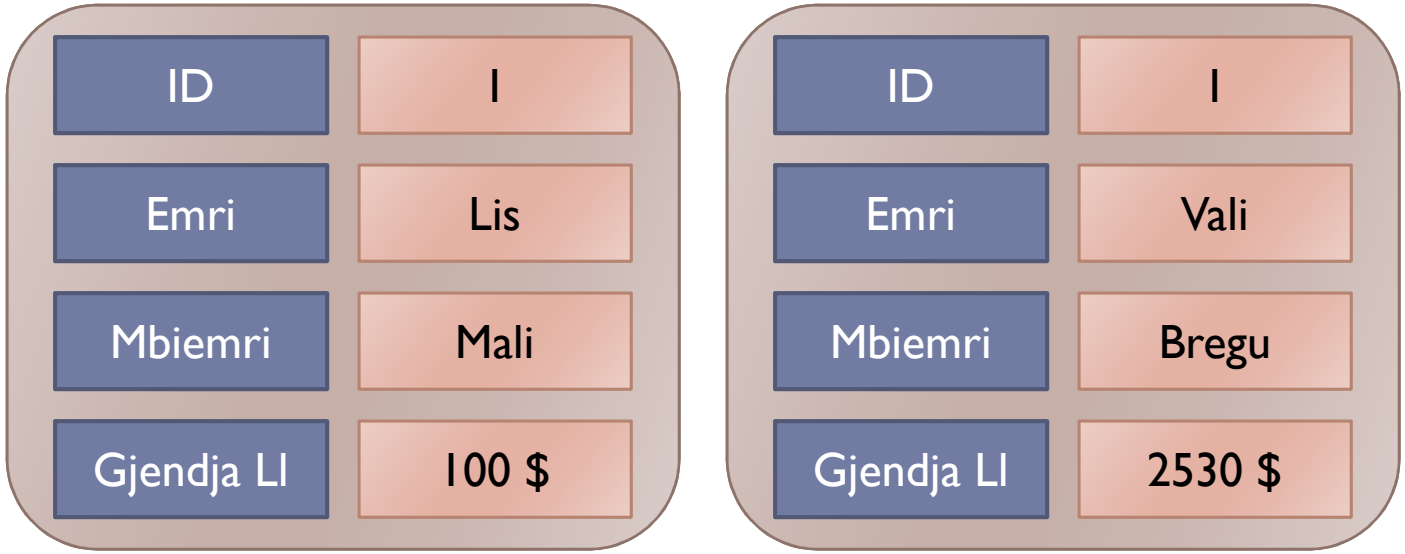
- **Variablat e instances**
  - Përcaktojnë gjendjen e cila varet nga objekte individuale
- **Variablat e klasës**
  - Përcaktojnë gjendjen e cila është e ndarë për të gjitha klasat
- **Metodat e instancës**
  - Aktivizojnë objektin. Mund t'u qasen variablave të klasës dhe instancës
- **Metodat e klasës**
  - Qasja përmes klasës. Mund t'u qasemi vetëm variablave të klasës.

• Variablat e klasës dhe metodat e klasës shënohen me modifikuesin **static**.

• Nuk përdoret modifikatorë **static** për variablat e instancës dhe metodat e instancës.

Variablat e instancës

- Variabla e instancës përcakton një pjesë të të dhënave në klasë.
- Secili objekt, krijohet si instancë e klasës, mban kopje të ndarë të variablave të instancës.



- -> Llogaria bankare

Klasa Banka.cs

```
public class Banka
   {
      protected int intLlogariaID = 0;
      protected string strEmriPronarit = "";
      protected string strMbiemriPronarit = "";
      protected double gjendjaLlogarise = 0.0;

      // konstruktori me parametrat LlogariaID , emriPronarit dhe MbiemriPronarit
      public Banka(int accountID, string EmriPronarit, string MbiemriPronarit)
      {
          this.intLlogariaID = LlogariaID;
          this.strEmriPronarit = EmriPronarit;
          this.strMbiemriPronarit = MbiemriPronarit;
      }



      // kontrollimi i gjendjes se llogarise
      public void ShiqoGjendjenELlogarise()
      {
          Console.WriteLine("Gjendja e llogarise tuaj eshte: {0} euro.", gjendjaLlogarise.ToString());
      }
```

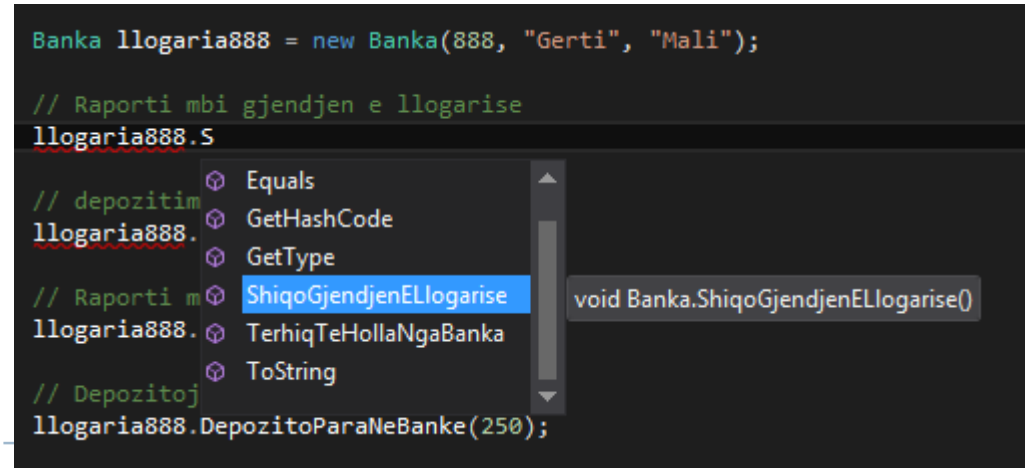
Klasa Banka.cs (vazh.)

```
// depozitimi i parave ne llogarine bankare
        public void DepozitoParaNeBanke(double shumaParave)
        {
                Console.WriteLine("Raport: Ju keni depozituar {0} euro ne llogarinë tuaj. \nGjendja e tanishme e
                llogarise tuaj është {1} euro"
            , shumaParave.ToString(), gjendjaLlogarise.ToString());
        }
public bool TerhiqTeHollaNgaBanka(int ShumaPerTerheqje)
        {
                if (ShumaPerTerheqje <= gjendjaLlogarise)
                   {
            this.gjendjaLlogarise -= ShumaPerTerheqje;
            Console.WriteLine("Raport: Ju keni terhequr {0} euro nga llogaria juaj. \nGjendja e tanishme e llogarise
                tuaj është {1} euro"
           , ShumaPerTerheqje.ToString(), gjendjaLlogarise.ToString());

            return true; // transaksioni u realizua me sukses!
          }
          else
          {
            // nuk ka fonde te mjaftueshme
            Console.WriteLine("Nuk keni te holla te mjaftueshme per ta terhequr shumen e kerkuar.");
            return false; // transaksioni nuk u krye me sukses!
          }

        }
    }
}
```

Metodat e instancës

• Metoda e instancës është një operacion në klasë e cila mund të lexoj ose modifikoj një ose më shumë variabla të instancës

- Një metodë e instancës M në klasën C
  - Duhet të aktivizohet në objektin i cili është instancë e C
  - Aktiviozehet përmes **objekti.M**(...) nga jashta C
  - Aktiviohet përmes **this.M(...)** ose vetëm **M(...)** brenda C
  - Mund të u qaset të gjithë anëtarëve të C



Variablat e klasës

Variabla e *klasës* i përket klasës, dhe ndahet në mes të gjitha instancave të klasës

- Variablat e klasës
  - Deklarohen duke shfrytëzuar modifikatorin **static** në C#
  - Mund të shfrytëzohen si variabla globale – të shoqëruara me klasën e dhënë
  - Përmbajnë të dhëna *meta* për klasën, si numrin e instancave

Metodat e klasës

Metoda e klasës shoqërohet me vet klasën, krahas asaj të objektit të klasës

- Metoda e klasës **M** në klasën **C**
  - Deklarohet duke shfrytëzuar modifikatorin **static** në C#
  - Mund të u qaset vetëm anëtarëve static të klasës
  - Duhet të thirret/aktivizohet në klasë
  - Aktiviohet si **C.M(...)** nga jashta **C**
  - Mund të aktivizohet edhe si **M(...)** nga brenda C

Klasat Statike dhe Parciale në C#

- Klasa statike C mund të ketë vetëm anëtarë statik
- Klasa parciale definohet në dy ose më shumë fajlla burimor

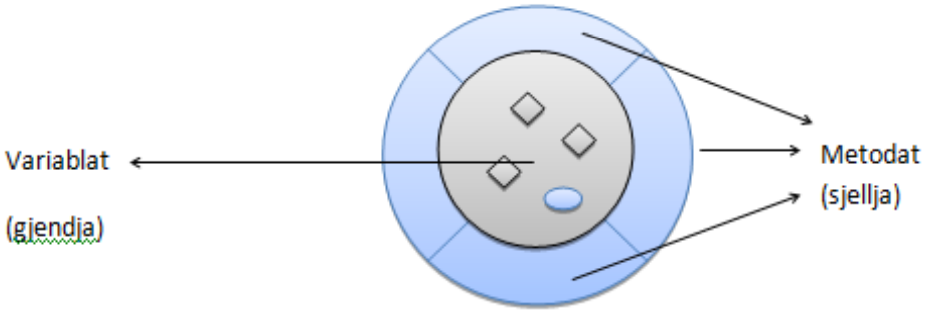
- **Klasa Statike**
  - Shërben më shumë si modul sesa *klasë*
  - Pengon instancimin, nënklasë, anëtarët e instancës, dhe shfryëtzimin si tip
  - Shembujt: System.Math, System.IO.File, dhe System.IO.Directory
- **Klasa Parciale**
  - Përdorimi: Të kombinoj pjesët e krijuara manualisht ato pjesë të gjeneruara automatikisht

Me shume per Klasat dhe Objektet

•Klasat implementohen dhe përshkruhen në fjallin burimor të programit

•Objektet krijohen dhe ekzistojnë gjatë ekzekutimit të programit

- Objektet karakterizohen me
  - Identitet
  - Gjendje
  - Sjellje



Objekti (gjendja dhe sjellja)

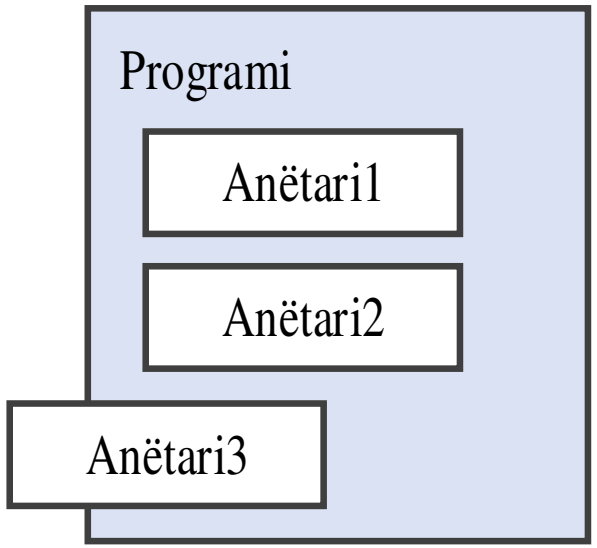
•Të gjitha objekteve u mbaron ekzistenca kur mbaron ekzekutimi i programit.

•Kjo është në konflikt me fenomenin e botës reale, dhe shkakton probleme dhe sfida në shumë programe.

Dukshmëria publike dhe private e klasës

- Figura në formë të drejtkëndëshit përfaqëson klasat, siç tregohet në Figurën.
  - Anëtarët e klasës paraqiten si drejtkëndësha më të vegjël brenda drejtkëndëshit të klasës.
  - Anëtarët privatë paraqiten tërësisht brenda drejtkëndëshit të tyre të klasës
  - Anëtarët publikë paraqiten pjesërisht jashtë drejtkëndëshit të tyre të klasës.

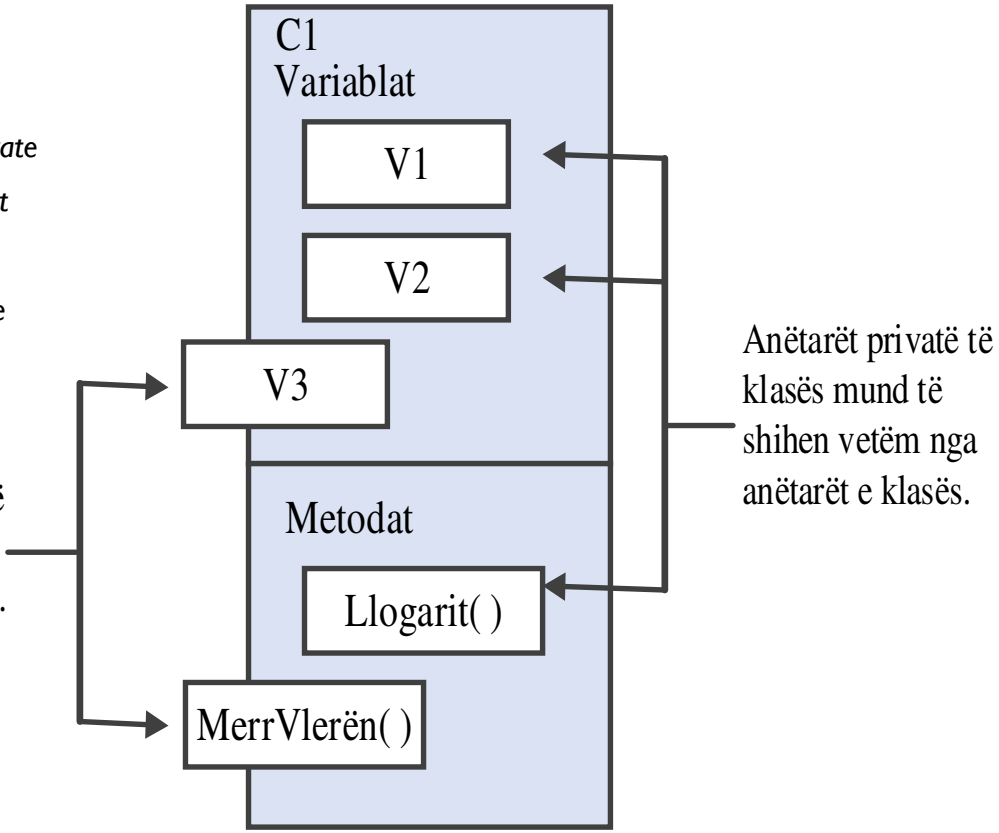
```
class Programi
{
        int Anëtari1;
    private int Anëtari2;
     public int Anëtari3;
}
```



Dukshmëria publike dhe private e klasës (Vazhdim)

- ***Shembull i dukshmërisë së anëtarëve të klasës:*** Në klasën C1 në kodin në vijim janë deklaruar variablat e instancës dhe metodat, si publike dhe private. Figura ilustron dukshmërinë e anëtarëve të klasës C1.

```
class C1 {
int V1;                    // Variabël implicite private
private int V2;            // Variabël eksplicite privat
public int V3;             // Variabël publike
void Llogarit()            // Metodë private implicite
{ ... }
public int MerrVlerën()    // Metodë publike
{ ...}}
```



Modifikatorët e qasjes

Modifikatori i qasjes	Kufizimet
Publik	Anëtarët *publik* të klasës C janë të gatshëm te çdo metodë e çdo klase. Asnjë kufizim.
Private	Anëtarët *privat* të klasës C janë të mundshëm vetëm te metodat e klasës C.
Protected	Anëtarët *protected* të klasës C janë të gatshëm te metodat e klasës C dhe gjithashtu edhe te metodat e klasave që derivojnë nga klasa C.
Internal	Anëtarët *internal* të klasës C janë të gatshëm te metodat e secilës klasë në asamblenë e klasës C.
Protected internal	Anëtarët *protected internal* të klasës C janë të gatshëm te metodat e klasës C, te metodat e klasave të derivuara nga C dhe gjithashtu te çfarëdo klase në asamblenë e klasës C.

Krijimi dhe Fshirja e Objekteve

Mundësitë e krijimit dhe fshirjes së objekteve

- Krijimi i objekteve
  - Duke instancuar klasat
    - Implicite: përmes deklarimit të variablave
    - Eksplicite: me kërkesë, ose me komandë
  - Duke kopjuar objektet ekzistuëse – *klonimi*
- Fshirja e objekteve
  - Ekplicite: me kërkesë , komandë
  - Implicite: fshi kur nuk ka përdorim të mëtejmë – përmes shfrytëzimit të *garbage collection (GC)*

Gjuhët moderne të orientuara në objekte përkrahin krijimin eksplicit të objekteve dhe fshirjen implicite (duke shfrytëzuar *garbage collection*)

Instancimi i Klasave

Instancimi është proces i alokimit të memories te një objekt i ri i ndonjë klase

- **Instancimi statik**
  - Objekti krijohet automatikisht (dhe shkatërrohet) kur objekti rrethuës ose kur krijohet blloku
- **Instancimi dinamik**
  - Objekti krijohet në kërkesë, duke thirrur operatorin përkatës (**new**).

Instancimi i Klasave në C#

Klasat dinamikisht duhet të instancohet duke shfrytëzuar operatorin **new**
Operatori **new** kthen një referencë tek objekti i ri

```
Banka llogariaX =

  new Banka(111, "Gerti", "Mali");
```

Inicializimi i objekteve

Inicializimi është proces i vendosjes së vlerave inicuëse te variablat e instancave të objektit

- Inicializimi i objekteve të tipit T
  - Përmes shfrytëzimit *vlerave të paracaktuar (defaults)* të T
    - Zero për tipe të numrave, *false* për boolean, '\x0000' për *char*, dhe *null* për tipet referenciale
  - Përmes shfrytëzimit të *initializer*
  - Përmes metodave speciale të quajtura *konstruktorë*

Është shumë e rëndësishme që një objekt i posalindur të inicohet si objekt i shëndoshë në popullaten e objekteve
Inicimi eksplicit preferohet gjithmonë krahas atij implicit
Gjithmonë inicializoni variablat në konstruktorë.

Konstruktorët dhe dekonstruktorët

- Konstruktoët dhe dekonstruktorët janë funksione speciale të klasës.
- Zakonisht këto "Funksione" i tregojnë komajlerit si te krijoje tipin e ri, sasin e memories se nevojitur për të dhënat anëtare te këtij tipi dhe si te liroje memorien e rezervuar.
- Kur shfrytëzuesi e deklaron një variabël te tipit te ri, atëherë komapjleri automatikisht thirr konstruktorin baze.
- Konstruktori ka emrin e njëjte sikurse klasa e deklaruar dhe nuk kthen asnjë tip vlere (as void) si dhe mund te ketë çfardo numri te parametrave te cfardo tipi.
- Konstruktori baze është konstruktori pa asnjë parametër.
- Nëse gjate definimit te klasës se re nuk deklarojmë asnjë konstruktorë, atëherë kompajleri krijon automatikisht një konstruktorë bazë (pa parametra).
- Dekonstruktori është e kundërta e konstruktorit, po ashtu funksion special, i cili paraprihet me shenjen ~ dhe me emrin e klasës.
- Kompajleri thërret dekonstruktorin e çdo tipi kur variabla e atij tipi del nga hapësira e veprimtarisë.

Konstruktorët dhe dekonstruktorët (Vazhd.)

Konstruktori është metodë speciale e cila thirret automatikisht në mënyrë që të inicializoj instancën e re të klasës

- Konstruktorët në C#
  - E kanë emrin e ngjashëm me atë të klasës
  - Nuk përcaktojnë ndonjë vlerë kthyëse
  - Shpesh janë të mbingarkuar – disa konstruktorë mund të shfaqen në një klasë
  - Mund – në mënyrë të veçantë – të delegoj punën tek konstruktori tjetër
  - Në rast se asnjë konstruktor nuk është definuar, është konstruktori pa parametra (*default constructor)*
    - Si aksion të vetëm, e thërret konstruktorin pa parametra të superklasës
  - Në rast të definimit të ndonjë konstruktori me parametra, nuk do të ketë konstruktor pa parametra.