KLASAT APSTRAKTE DHE TRAJTIMI I GABIMEVE

KAPITULLI 8

Prof. Ass. Dr. Isak Shabani

Klasat apstrakte

• *Klasë abstrakte: klasë së cilës i mungojnë disa nga* metodat e veta; metodat që mungojnë vëhen në pah me ballinë e cila përmban fjalën kyçe `abstract`.

• *Metodë abstrakte: metodë pa trup e cila ndodhet në një* klasë abstrakte. Trupi i metodës shkruhet në një nënklasë të klasës abstrakte.

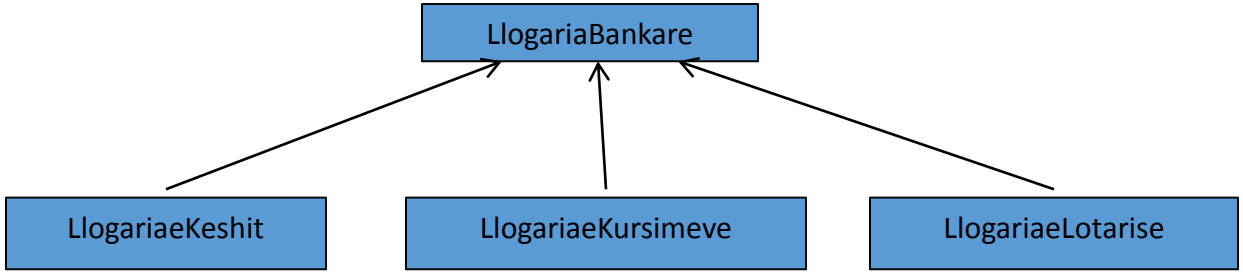
• *Klasë konkrete: klasë „normale“, të gjitha metodat e së* cilës kanë trupa.

Klasat apstrakte (2)

• Kur ne programojmë në modelin e orientuar në objekte (object-oriented) është e rëndësishme të punojmë jashtë koncepteve të përgjithshme nëse është e mundur.

• Në gjuhët e programimit të orientuar në objekte (object-oriented programming languages) ne organizojmë klasat në hierarki. Klasat më afër rrënjës janë klasat më të përgjithshme.

• Marrim, një shembull, hierarkinë e klasës për llogarinë bankare (`LlogariaBankare`) të paraqitur si në figurën e më poshtme, ku klasa `LlogariaBankare` është më e përgjithëshme se `LlogariaeKeshit`, `LlogariaeKursimeve`, etj.



Një hierarki e specializuar e Llogarive bankare

Nga figura shifet se kemi dhënë tri klasat që specializojnë klasën `LlogariaBankare`

Klasat apstrakte (2)

• Le të paraqesim më shumë detaje për klasat dhe operatorët abstrakt :

- Klasat abstrakte
  - mund të rrjedhë nga një klasë jo-abstrakte
  - nuk ka nevojë që të ketë anëtarë abstrakt
  - mund të ketë konstruktorë
- Metodat abstrakte
  - janë absolutisht virtuale

Fakti që një klasë abstrakte nuk mund të instancohet është i qartë, si pasojë operacionale e punës me klasat abstrakte

Në situatat e rralla një klasë abstrakte mund të trashëgohet prej një klase jo-abstrakte

Siç është diskutuar më lartë qëllimi i shenjëzimit është të parandalojmë instancimin e klasës.

TRAJTIMI I GABIMEVE

- **Pyetjet themelore për Trajtimin e Përjashtimit**

- **Çfarë është një gabim?**

- Gabimet në programin burimor
  - Përdorimi ilegal i gjuhës së programimit

- Gabimet në ekzekutimin e programit-gabimet gjatë ekzekutimit
  - Përjashtimet – të shoqëruara nga trajtimi potencial i përjashtimeve
  - Viruset – të shoqëruara nga pastrimi potencial i viruseve për të korigjuar programin burimor

- Gabimet në procesin ndërtimit/zbatimit
  - Për shkak të një vendimi të gabuar në një pikë më të hershme në kohë - një defekt mendor
  - Zbatimi i gabuar i një algoritmi

TRAJTIMI I GABIMEVE

- **Çfarë është normale?**
- **Çfarë është e jashtëzakonshme?**

- Aspektet normale të programit
  - Situatat të parashikuara dhe të trajtuara me rrjedhjen konvencionale të programit
  - Të programuar me përdorimin e strukturave përzgjedhëse dhe përsëritëse të kontrollit

- Aspektet e jashtëzakonshme të programit
  - Situatat e parashikuara, por që nuk trajtohen “në mënyra normale” nga programuesi

Kur zbulohen gabimet ?

- Gjatë ndërtimit dhe programimit – Kërkoje.
- Gjatë testimit - Shumë punë. Por e nevojshme.
- Gjatë ekzekutimit dhe përdorimit final të programit.
- Gjatë përpilimit – gabimet e sintaksës ose gabimet e llojit - Tërheqëse.
  - Gabime të trajtuara - Në rregull. Por vështirë.
  - Gabime të pa trajtuara - Shumë frustrime.

Si trajtohen gabimet *?*

- Injoro
  - Alarmi i rrejshëm – Naiv
- Raporto
- Përfundo
  - Ndal ekzekutimin e programit në një mënyrë të kontrolluar dhe të mirë - Ruaj të dhënat, lidhje të ngushta
- Riparo
  - Rindreq nga gabimi gjatë ekzekutimit - Vazhdo ekzekutimin normal të programit kur problemi zgjidhet
  - Shkruaj një mesazh në ekran ose në një regjistër - E dobishme për korrigjimin vijues të programit burimor

- **Përjashtimi në një program C#**

```
using System;
class DemostrimPerjashtimi{

    public static void Main(){
        int[] tabela = new int[6] { 10, 11, 12, 13, 14, 15 };
        int idx = 6;

        M(table, idx);
    }

    public static void M(int[] tabela, int idx){
            Console.WriteLine("Qasja ne elemente {0}: {1}",
                                idx, tabela[idx]);
    }
}
```

***Programi** 6.1 Një program C# me një përjashtim.*

```
Perjashtimi Patrajtuar: System.PerjashtimiIndeksitJashtNgaVargu:
 Indeksi ishte jasht kufijve te vargut.
   at DemostrimPerjashtimi.M(Int32[] tabela, Int32 idx)
   at DemostrimPerjashtimi.Main()
```

***Lista** 6.1 Daljet nga programi C# me një përjashtim.*

Formulimi provo-kap C# (The try-catch statement C#)

- Trajtimi i një përjashtimi përfshihet në disa situata që ne përpiqemi për t'i përmisuar nga gabimi i cili është përfaqësuar nga përjashtimi, në atë mënyrë që mund të vazhdoj ekzekutimi i programit.

```
try
  try-body
catch (exception-type-1 name)
  catch-body-1
catch (exception-type-2 name)
  catch-body-2
...
```

```
using System;
class DemostrimPerjashtimi{

   public static void Main(){
     int[] tabela = new int[6]{10,11,12,13,14,15};
     int idx = 6;

     M(tabela, idx);
  }

  public static void M(int[] tabela, int idx){
     try{
      Console.WriteLine("Qasja ne elemente {0}: {1}",
                        idx, tabela[idx]);
      Console.WriteLine("Qasja ne elemente {0}: {1}",
                        idx - 1, tabela[idx - 1]);
    }
    catch (PerjashtimiIndeksitJashtNgavargu e){
      int newIdx = RregulloIndeksin(idx,0,5);
      Console.WriteLine("Ne morem elementin numer {0}: {1}",
                        newIdx, tabela[newIdx]);
   }
```

Hierarkia e Përjashtimit në C#

- Lloji i përjashtimeve mund të jet
  - Përjashtim Aplikacioni (ApplicationException)
  - Përjashtim Sistemi (SystemException)
  - Përjashtim Argumenti (ArgumentException)
  - Përjashtim i pavlefshëm Argumenti (ArgumentNullException)
  - Përjashtim Argumenti Jashtë Nga Vargu(ArgumentoutofRangeException)
  -

Përjashtimet dhe Trajtimi i Përjashtimit në C#

- PërjashtimiPjesëtuarMeZero (*DivideByZeroException*)
- PërjashtimiIndeksitJashtëNgaVargu*(IndexOutOfRangeException*)
- PërjashtimiPavlefshëmIReferencës (*NullReferenceException)*
- PërjashtimiIRradhës(*RankException*)
- PërjashtimIStrukturësSëTëDhënaveTëMbushuraTepër (*StackOverflowException)*
- PërjashtimIO (*IOException*)
- PërjashtimIFunditTëRrjedhjes (*EndOfStreamException*)
- PërjashtimFajlliNukËshtëGjetur (*FileNotFoundException*)
- PërjashtimFajlliËshtëNgarkuar (*FileLoadException)*

Përjashtimet dhe Trajtimi i Përjashtimit në C#

- **Provo-kap (try-catch)**
- **me një klauzolë**
- **më në fundë**
- **(finally clause)**
- **në C#**

```
try
  try-body
catch (exception-type name)
  catch-body
...
finally
  finally-body
```

Përjashtimet dhe Trajtimi i Përjashtimit në C#

Shembulli:

```
using System;

class DemostrimMeNeFund{

 internal enum Control { Kthimi, Kercimi, Vazhdimi, Nderprerja,
                                                        Hedhja }

   public static void M(Control reason){
     for (int i = 1; i <= 1; i++) // Nje perseritje e vetme
         try{
           Console.WriteLine("\nEnter try: {0}", reason);
           if (reason == Control.Kthimi) return;
           else if (reason == Control.Kercimi) goto finish;
           else if (reason == Control.Vazhdimi) continue;
           else if (reason == Control.Nderprerja) break;
           else if (reason == Control.Hedhja) throw newException();
                  Console.WriteLine("Inside try");
          }
          catch (Exception){
                  Console.WriteLine("Inside catch");
          }
          finally{
                  Console.WriteLine("Inside Finally");
          }
    finish: return;
    }

    public static void Main(){
         for (int i = 0; i <= 5; i++)
              M((Control)i);
    }
}
```

***Programi** 6.10 Ilustrimi i provo-kap-më në fund (try-catch-finally).*

Të shkruarit e një Manovruesi të Gabimit

➢**Provo(Try).....Kap(Catch)....Më në Fund(Finally)**

➢Janë tri forma të formilimeve provo (try):

- •Një bllok provo(try) i pasuar nga një ose më shume blloqe kap (catch blocks).
- •Një bllok provo(try) i pasuar nga një bllok më në fund (finally block).
- •Një bllok provo(try) i pasuar nga një ose më shume blloqe kap (catch blocks) dhe i pasuar nga një bllok më në fund (finally block).

Konkludime për trajtimin e gabimeve

- Zhvillimi i teknologjisë informative, bashkë me të natyrisht edhe i softuerëve është një ndër kërkesat e pakontestueshme të kohës.
- Zgjidhja e veglës së duhur për secilin problem është çelës për me qenë programer efektiv. Teknikat e programimit mbrojtës i bëjnë gabimet të gjenden më lehtë, të rregullohen më lehtë dhe më pak të dëmshme për kodin në produksion.
- Vendimi se si të trajtohen hyrjet e këqija(bad inputs) është vendim kryesor i trajtimit të gabimeve që operon në dimensione të ndryshme nga rrjedhja normale e kodit.
- Trajtimi i gabimeve është një ndër aspektet më të rëndësishme për zhvillimin e e suksesshëm dhe të fuqishëm të softuerit.
- Aspekt i rëndësishëm i trajtimit të gabimeve, është dallimi në mes të gabimeve në procesin e zhvillimit, gabimet në programin burimor dhe gabimet në programin ekzekutues.
- Pra, programimi mund të jetë argëtues, por zhvillimi i softuerët cilësor natyrisht është i vështirë. Në mes ideve të mira, kërkesave apo vizionit, dhe një produkti softuerik ka më shumë se programim.

• Analiza dhe dizajni në atë mënyrë që të jenë të lehta për të komuniku, rishiku, implementu dhe evolu- është ajo që qëndron në thelb të programimit.