Избацивање изузетака

У програмима који враћају вредности у зависности од унете вредности од стране корисника, најсигурнији начин реакције програма ако је унета непостојећа вредност (нпр назив месеца у календару на основу неодговарајућег редног броја месеца) је да се креира метода која ће за такаву улазну вредност избацити изузетак.

Библиотека класа .NET обухватају мноштво класа изузетака посебно предвиђених за овакве околности.

У овом случају се може користити класа ArgumentOutOfRangeException.

Изузетак се избацује коришћењем исказа throw.

Пример:

```
public static string NazivMeseca(int mesec)
{
     switch (mesec)
     {
          case 1:
              return "januar";
          case 2:
              return "februar";
          ...
          case 12:
              return "decembar";
          default:
              throw new ArgumentOutOfRangeException("Pogresan mesec");
     }
}
```

Исказу throw је неопходан објекат изузетка који треба да се избаци.

Тај објекат садржи детаље о изузетку укључујући могуће поруке о грешци.

У примеру се користи израз који прави нов објекат `ArgumentOutOfRangeException` .

Објекат се иницијализује стрингом који попуњава његово својство Message коришћењем методе конструктора.

Коришћење изузетка throw

Изузетак throw је семантички сличан са исказом throw, осим по томе што се може користити било где где с екоридти израз.

Нпр, треба стринг ime поставити на вредност унету у програм, али само је ако корисник заиста и унео неку вреност, иначе треба избацити изузетак “Nedostaje unos“.

```
string ime, unetoIme;
if (unetoIme != "")
{
     ime = unetoIme;
}
else
{
     throw new Exception("Nedostaje unos"); //izraz throw
```

```
}
```
Ако промењива unetoIme није празно, вредност се чува у промењивој ime.
Иначе, процењује се израз throw, који са своје стране избацује изузетак.

Коришћење блока finaly

Када се избаци изузетак, он мења ток извршавања кода кроз програм; због могућег избацивања изузетка, неки делови кода се неће извршавати.

```
TextCitac citac = ...;
...
string linija = citac.ReadLine();
while (linija != null)
{
     ...
     linija = citac.ReadLine();
}
citac.Dispose();
```

Према делу кода се види да ће се линија `citac.Dispose();` десити увек када се заврши петља.
Овде би био проблем када се та линија не би извршила пошто је она неопходна да би се ослободили меморијски ресурси које претходни део кода користи.
Да би били сигурни да ће нека линија кода да се обавезно извршава, без обзира на избацивање изузетка, јесте да се исказ напише унутар блока finaly.
Блок finaly се налази непосредно после блока try или непосредно после последње процедуре catch после блока try.
Увек када програм уђе у блок try којем је придружен блок finaly, блок finaly ће се извршити, чак и ако се деси изузетак.
Ако се неки изузетак избаци и ухвати локално, прво се извршава процедура за обраду изузетка а затим блок finaly.
Решење за претходни проблем:

```
TextCitac citac = ...;
...
try
{
     string linija = citac.ReadLine();
     while (linija != null)
     {
          ...
          linija = citac.ReadLine();
     }
}
finally
{
     if(citac != null)
     {
          citac.Dispose();
```

```
    }
}
```

Аритметички проверени и непроверени бројеви

Сви типови података имају своја ограничења, максималне и минималне вредности, унутар којих граница вредности се могу кретати.
Покушај прекорачења тих граница се такође сматра изузетком и може изазвати прекид програма.
Провера аритметичког прекорачења се врши помоћу кључних речи checked и unchecked (укључивање и искључивање провере).

Исказ који се проверава јесте блок коме претходи реч checked.
Све аритметичке операције са целим бројевима у провераваном исказу увек избацују изузетак OverflowException ако прорачун у том блоку премаши границе.
Пример:

```
int broj = int.MaxValue;
checked
{
    int izbacuje = broj++;
    Console.WriteLine("ovo se nece ispisati");
}
```

Исказ који се не проверава јесте блок коме претходи реч unchecked.
Све аритметичке операције са целим бројевима у непровераваном исказу никад не избацују изузетак OverflowException.
Пример:

```
int broj = int.MaxValue;
unchecked
{
    int neizbacuje = broj++;
    Console.WriteLine("ovo ce se ispisati");
}
```

Могуће је ове кључне речи користити за проверу прекорачења на целобројним изразима:

```
int neizbacuje = unchecked(int.MaxValue + 1);
int izbacuje = checked(int.MaxValue + 1);
```

Задаци за самосталан рад

1. Написати програм за добијање квадрата унетог целог броја. Код треба да спречи прекид програма ако корисник унесе неодговарајућу вредност. Користити try-catch-finally блокове и исписивање несистемске поруке.

2. Написати програм за добијање квадрата унетог целог броја. Код треба да спречи прекид програма ако корисник унесе неодговарајућу вредност. Користити try-catch-finally блокове и класу Exception која даје одговарајућу системску поруку кориснику.

3. Коришћењем try-catch-finally блока спречити прекид програма ако је садржај који се чита из текстуалног фајла једнак null.