УПРАВЉАЊЕ ГРЕШКАМА И ИЗУЗЕЦИМА

Постоје три основна типа грешака. **Синтаксне грешке** су грешке при писању кода, јављају се при компајлирању и најлакше их је отклонити. **Логичке грешке** је најтеже открити јер често не постоји никакав наговештај да грешка постоји док се не уочи да су очекивани резултати погрешни. **Грешке у време извршавања** се јављају после компајлирања програма када нпр. желимо да спроведено недозвољену радњу дељењем нулом.

Изузеци се користе како би контролисали нежељене ситуације (нпр. дељење са нулом, приступ елементу низа са непостојећим индексом …)

За рад са изузецима користимо кључне речи: **try, catch** и **finally.**

try	блок у којем одређене наредбе могу изазвати грешку.
catch	блок за обраду грешака, тј. овај блок се извршава у случају настанка грешке.
finally	блок који се извршава иако се деси и ако се не деси грешка.

Структура за управљање изузецима тј. блок try – catch – finally за руковање изузецима дата је на слици

```
try
{
        kod koji se izvršava do pojave greške

}
catch (exception)
{
        kod koji se izvršava ukoliko dođe do greške

}
finally
{
       kod koji se izvršava bez obzira da li je bilo greške ili nije

}
```

Правила:

- Блок **try** мора имати бар један **catch** блок.
- **Блок try** може имати више **catch** блокова, али ако наведемо више **catch** блокова битан је њихов редослед. Тачније, редослед **catch** блокова мора бити од специфичних изузетака (***ArithmeticException*** – неисправан резултат аритметичке операције, дељење са нулом) ка општим – основним изузецима (***Exception*** **–** обрађује све изузетке)
- Блок **try** може али и не мора да има **finally** блок

Генерисање изузетака (бацање изузетака) – throw

Преко кључне речи **throw**, изузетак се може изазвати програмски – ручно.

Пример:

У следећем примеру анализирати програмски код и објаснити шта ће бити приказано на екрану.

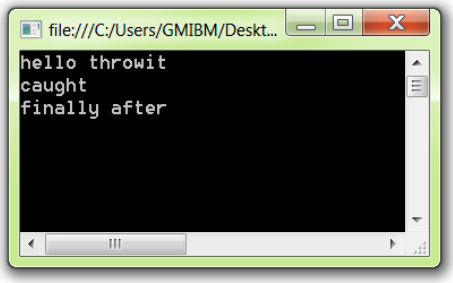
```
using System;

namespace Zadatak
{
    class Program
    {

        public static void throwit(){
            Console.WriteLine("throwit");
            throw new ApplicationException();
        }

        static void Main(string[] args)
        {
            try{
                Console.Write("hello ");
                throwit();
            }
            catch(Exception re){
                Console.WriteLine("caught ");
            }
            finally{
                Console.Write("finally ");
            }

            Console.Write("after");
            Console.ReadKey();
        }
    }
}
```



Покретањем апликације, програм изврашава метод **Main().** Исписује се **“hello”** и позива метод **throwit().** У овој методи, исписује се **“throwit”** и баца (генерише) изузетак. Извршава се **catch** и исписује “**caught**”. Затим се извршава **finally** и исписује “**finally**”, после чега се исписује “**after**”. Као што видимо редослед приказивања ће бити: **hello throwit caught finally after**

ЗА НАВЕДЕНЕ ПРИМЕРЕ НАПИСАТИ ТАЧАН ОДГОВОР:

213.	Дати су искази који се односе на правила писања try-catch-finally блокова за руковање изузецима. Који искази су тачни: 1. Блок try мора имати бар један catch блок 2. Блок try може имати више catch блокова 3. Ако блок try има више catch блокова, изузетак основне Exception класе мора се хватати у првом catch блоку 4. Ако блок try има више catch блокова, битан је редослед њиховог писања 5. Блок try мора имати бар један finally блок 6. Блок try не сме да има више catch блокова	**1,5**

229.	Започете су изјаве које се односе на делове кода за обраду изузетака. Довршити започете реченице: Наредбе које се извршавају у случају настанка грешке, стављају се унутар блока _____ Наредбе које се извршавају и ако се деси и ако се не деси грешка, стављају се унутар блока _____ Наредбе које могу изазвати грешку стављају се унутар блока _____	**1,5**

178. Дат је део кода који је написан у C# програмском језику. Одредити шта ће бити на излазу:

```
publicclassRTExcept
{
publicstaticvoid throwit() {
Console.WriteLine("throwit ");
thrownewApplicationException();
      }
publicstaticvoid Main(String[] args) {
      try {
      Console.Write("hello ");
               throwit();
           }
      catch (Exception re ) {Console.WriteLine("caught ");}
      finally { Console.Write("finally "); }
Console.Write("after ");
      }
}
```

1. hello throwit caught
2. Грешка приликом компајлирања
3. hello throwit *RuntimeException* caught after
4. hello throwit caught finally after

1

177. Дат је део кода који је написан уC# програмском језику. Одредити шта ће се приказати на излазу:

```
try
{
int x = 0;
int y = 5 / x;
}
catch (Exception e)
{
Console.WriteLine("Exception");
}
catch (ArithmeticException ae)
{
Console.WriteLine(" Arithmetic Exception");
}
Console.WriteLine("finished");
```

1. Приказује се текст: finished
2. Приказује се текст: Exception
3. Ништа. Дешава се грешка приликом компајлирања
4. Приказује се текст: Arithmetic Exception

1

НАПОМЕНА: Зашто је тачан одговор под 3. Зашто се јавља грешка при компајлирању? Погледати друго правило, где смо навели да приликом дефинисања више catch блокова мора бити испоштован редослед навођења изузетака**: од специфичних ка основним**. Овде то није испоштовано и зато се јавља грешка. У следећем питању иста је логика: изузетак основне **Exception** класе мора се "хватати" у последњем **catch** блоку.

Заокружити број испред исказа који представља исправан наставак дате реченице:

Ако try-catch наредба има више catch блокова у којима "хватамо" изузетак основне ***Exception*** класе, заједно са изузецима других класа изведених из класе ***Exceptions***...

1. онда се изузетак основне Exception класе може „хватати" у било ком catch блоку (редослед није битан, битно је да се наведу све могуће грешке)
2. онда се изузетак основне Exception класе мора „хватати" у последњем catch блоку
3. онда се изузетак основне Exception класе мора „хватати" у првом catch блоку
4. основна Exception класа се не комбинује у истој наредби са класама изведеним из ње јер их оснавна класа „маскира"

2