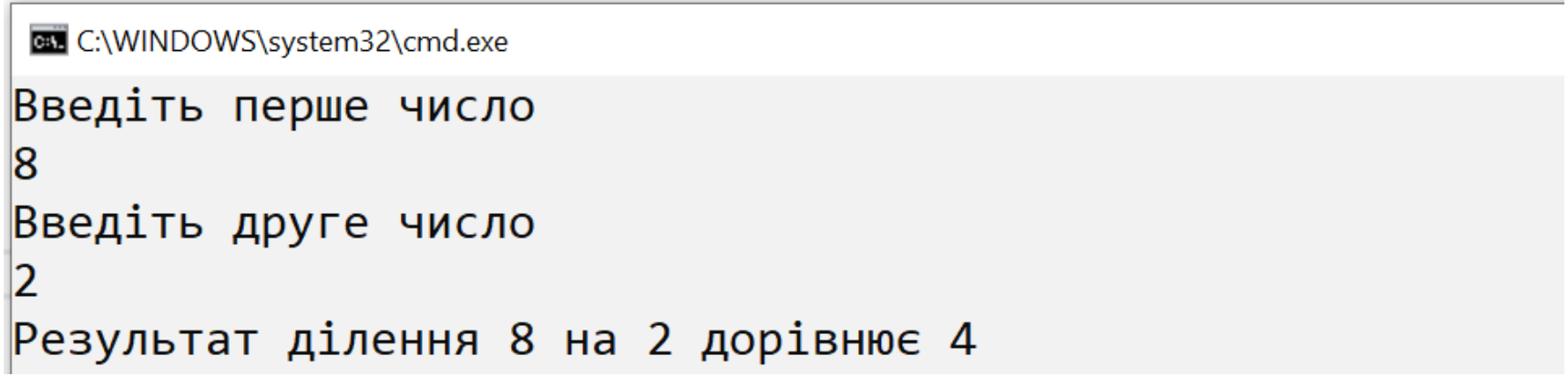
Знайомство з ООП. Поглиблене вивчення мови C#

Тема 5. Виняткові ситуації

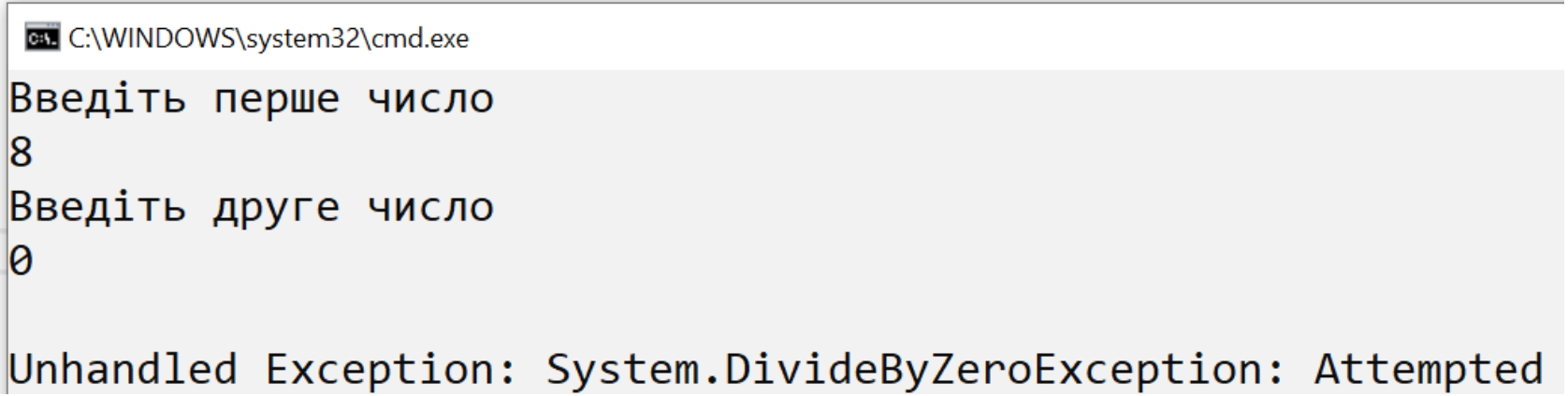
Програма виконалася без помилок

```
Console.WriteLine("Введіть перше число");
int i = Int32.Parse(Console.ReadLine());
Console.WriteLine("Введіть друге число");
int j = Int32.Parse(Console.ReadLine());
int k = i / j;
Console.WriteLine("Результат ділення {0} на {1} дорівнює {2}", i, j, k);
```



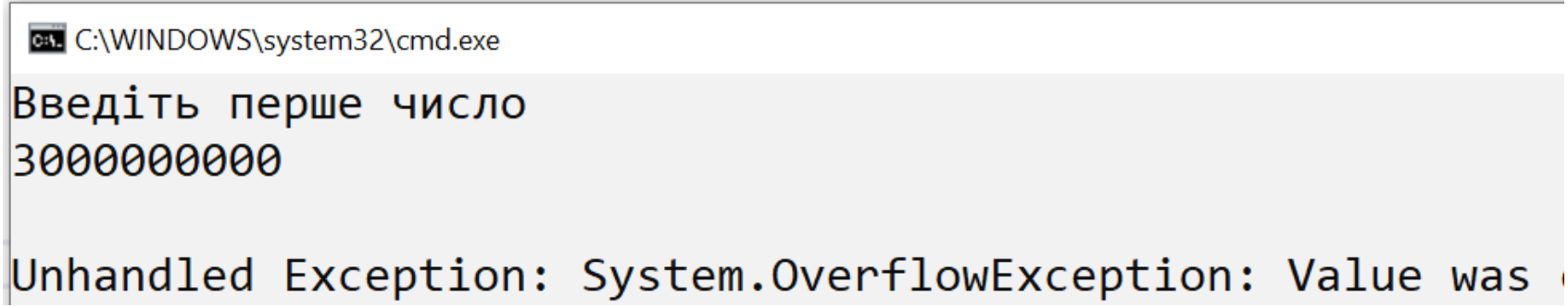
Помилка «Ділення на нуль»

```
Console.WriteLine("Введіть перше число");
int i = Int32.Parse(Console.ReadLine());
Console.WriteLine("Введіть друге число");
int j = Int32.Parse(Console.ReadLine());
int k = i / j;
Console.WriteLine("Результат ділення {0} на {1} дорівнює {2}", i, j, k);
```



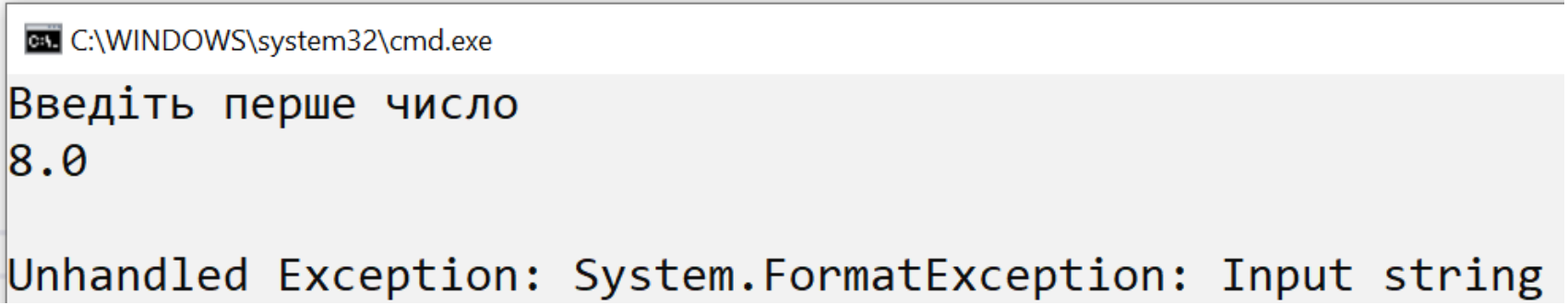
Помилка «Переповнення»

```
Console.WriteLine("Введіть перше число");
int i = Int32.Parse(Console.ReadLine());
Console.WriteLine("Введіть друге число");
int j = Int32.Parse(Console.ReadLine());
int k = i / j;
Console.WriteLine("Результат ділення {0} на {1} дорівнює {2}", i, j, k);
```



Помилка «Некоректний формат даних»

```
Console.WriteLine("Введіть перше число");
int i = Int32.Parse(Console.ReadLine());
Console.WriteLine("Введіть друге число");
int j = Int32.Parse(Console.ReadLine());
int k = i / j;
Console.WriteLine("Результат ділення {0} на {1} дорівнює {2}", i, j, k);
```



Виняткові ситуації

Виняткова ситуація – це помилка у програмі, яка виникає внаслідок порушення системних або програмних обмежень. Виняткова ситуація (її ще називають винятком) викликає переривання або припинення виконання програми. Обробка виняткової ситуації полягає у нейтралізації динамічної помилки, яка її визвала. Виняткова ситуація залишається активною, доки не буде оброблена. Якщо програма не містить коду для обробки виняткової ситуації, то система припинить виконання програми.
Коли виникає виняткова ситуація, система автоматично генерує об’єкт виняткової ситуації, який містить дані про неї.

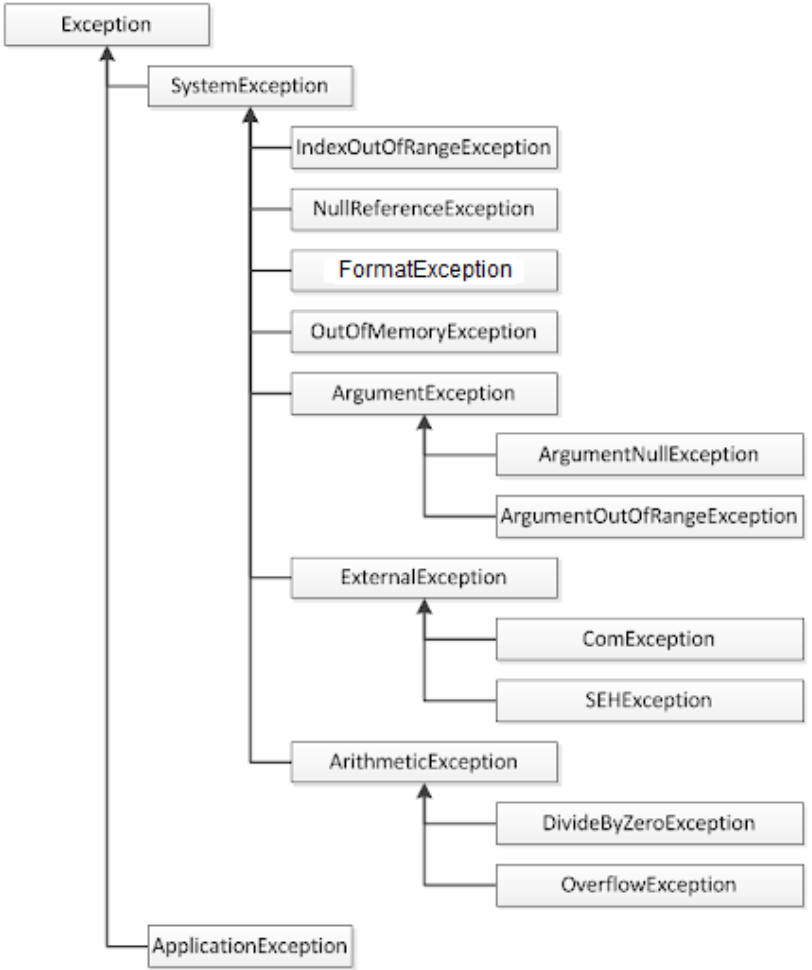
Класи виняткових ситуацій

- У програмах можуть траплятися винятки багатьох різних типів.
- Бібліотека базових класів .NET Framework (BCL) визначає велику кількість класів, які представляють виняткові ситуації конкретного виду. Коли трапляється виняткова ситуація, середовище виконання .NET (CLR) створює об'єкт виняткової ситуації певного типу і шукає прийнятний пункт catch для її обробки.
- Всі винятки походять від класу Exception, оголошеного у просторі імен System. Цей клас інкапсулює загальні характеристики виняткової ситуації.
- Об'єкт виняткової ситуації містить властивості (доступні лише для читання) з інформацією про виняток.

Класи виняткових ситуацій

Всі винятки походять від класу Exception

На основі класу Exception створено велику кількість класів, які описують виняткові ситуації певного виду: помилки при роботі з числами із плаваючою комою, помилки при роботі з пам'яттю, помилки дискового вводу-виводу тощо.



Найвживаніші властивості класу Exception

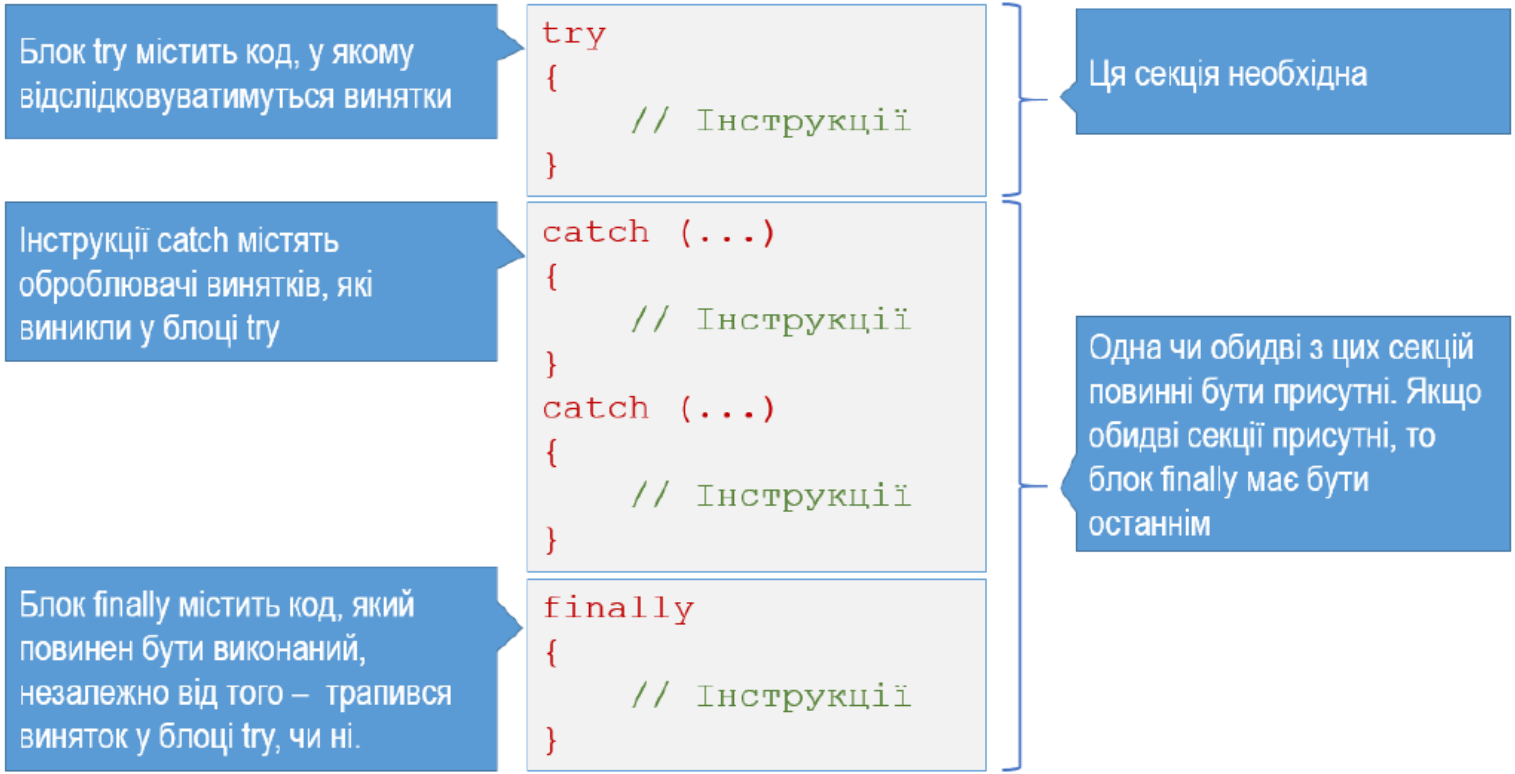
Властивість	Тип	Опис
Message	string	Містить повідомлення про помилку, яке пояснює причину винятку
StackTrace	string	Містить інформацію, яка описує, де сталась помилка
InnerException	Exception	Якщо поточний виняток згенерований іншим винятком, ця властивість містить посилання на попередній виняток
HelpLink	string	Винятки, описані програмно, можуть задати у цій властивості посилання (URN, URL) на інформацію, яка пояснює причину винятку
Source	string	Якщо ця властивість не встановлена програмно, то вона містить назву збірки, де виник виняток

Обробка виняткових ситуацій

Для обробки виняткових ситуацій використовують інструкцію try. Вона дозволяє позначити блок коду, в якому відслідковуватимуться виняткові ситуації, а також забезпечує код для їх обробки. Інструкція try може складатися з таких трьох секцій:

- Блок try містить код, у якому відслідковуватимуться винятки.
- Секція catch містить одну чи кілька інструкцій catch, кожна з яких має свій блок коду для обробки винятків. Інструкції catch називають ще оброблювачами виняткових ситуацій.
- Блок finally містить код, який виконається при будь-яких умовах: трапиться виняток, чи ні.

Структура конструкції try



Один catch для всіх винятків

```
try
{
    Console.WriteLine("Введіть перше число");
    int i = Int32.Parse(Console.ReadLine());
    Console.WriteLine("Введіть друге число");
    int j = Int32.Parse(Console.ReadLine());
    int k = i / j;
    Console.WriteLine("Результат ділення {0} на {1} дорівнює {2}", i, j, k);
}
catch (Exception e)
{
    Console.WriteLine("Помилка: {0} ", e.Message);
}
```

Три catch
для різних
типів помилок

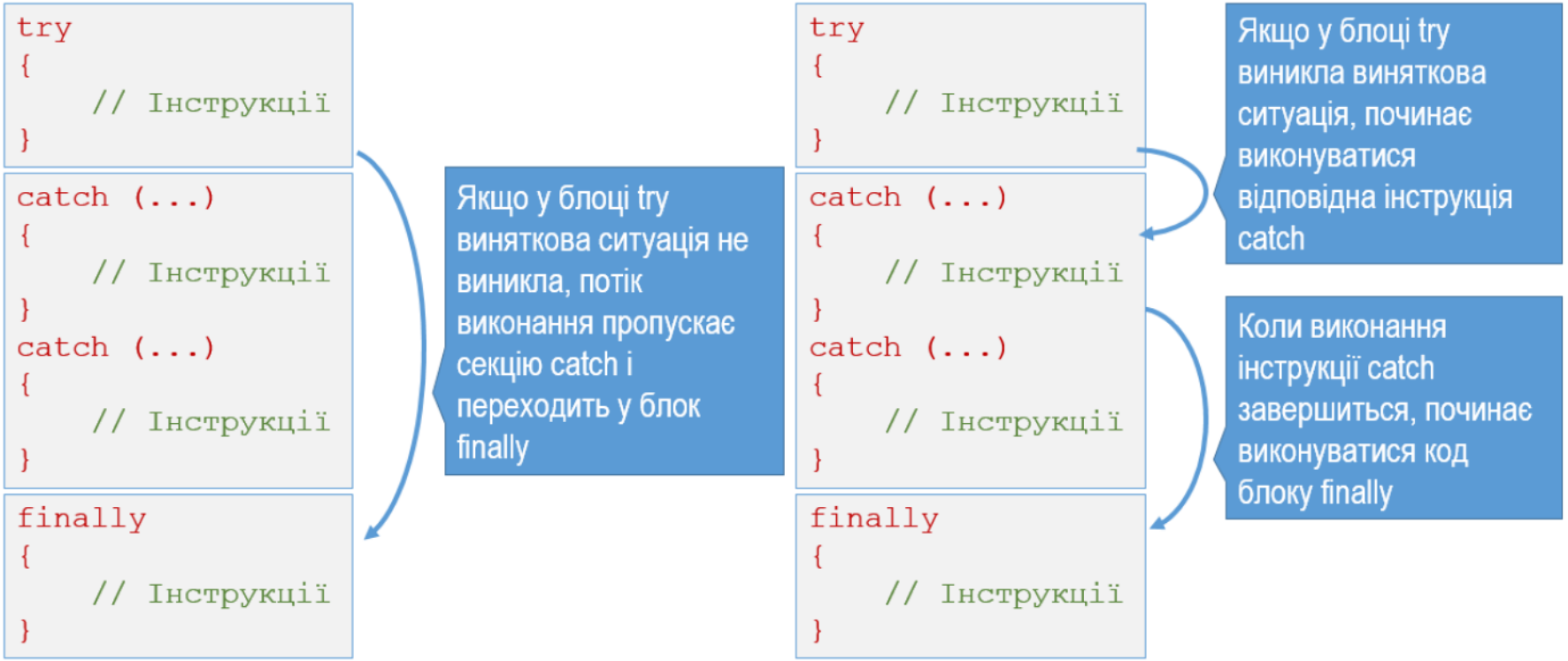
```
try
{
    Console.WriteLine("Введіть перше число");
    int i = Int32.Parse(Console.ReadLine());
    Console.WriteLine("Введіть друге число");
    int j = Int32.Parse(Console.ReadLine());
    int k = i / j;
    Console.WriteLine("Результат ділення {0} на {1} дорівнює {2}", i, j, k);

}
catch (FormatException)
{
    Console.WriteLine("Некоректний формат вхідних даних");
    Console.WriteLine("Вхідний рядок містить інші символи окрім цифр");
}
catch (OverflowException)
{
    Console.WriteLine("Переповнення");
    Console.WriteLine("Число більше ніж {0}", int.MaxValue);
}
catch (DivideByZeroException)
{
    Console.WriteLine("Ділення на нуль");
}
```

Блок finally виконується завжди

```
try
{
    Console.WriteLine("Введіть перше число");
    int i = Int32.Parse(Console.ReadLine());
    Console.WriteLine("Введіть друге число");
    int j = Int32.Parse(Console.ReadLine());
    int k = i / j;
    Console.WriteLine("Результат ділення {0} на {1} дорівнює {2}", i, j, k);
}
catch (Exception e)
{
    Console.WriteLine("Помилка: {0} ", e.Message);
}
finally
{
    Console.WriteLine("Виконання програми завершено!!!");
}
```

Виконання блоку finally



Генерування винятків

За потреби в коді можна явно генерувати виняткові ситуації. Для цього використовують інструкцію throw.
Синтаксис її використання такий:
throw exceptionObject;
де exceptionObject – посилання на об'єкт винятку (класу Exception або похідного класу).
Наприклад, можна генерувати виняткову ситуацію, коли перше число не ділиться без залишку на друге.

Якщо залишок не дорівнює 0, то генеруємо помилку

```
try
{
    Console.WriteLine("Введіть перше число");
    int i = Int32.Parse(Console.ReadLine());
    Console.WriteLine("Введіть друге число");
    int j = Int32.Parse(Console.ReadLine());
    if (i % j != 0)
    {
        string s = String.Format("Число {0} не ділиться без залишку на {1}", i, j);
        throw new Exception(s);
    }
    int k = i / j;
    Console.WriteLine("Результат ділення {0} на {1} дорівнює {2}", i, j, k);
}
catch (Exception e)
{
    Console.WriteLine("Помилка: {0} ", e.Message);
}
```

```
C:\WINDOWS\system32\cmd.exe
Введіть перше число
27
Введіть друге число
6
Помилка: Число 27 не ділиться без залишку на 6
```

Домашнє завдання

Доповнити проєкт з попереднього домашнього завдання можливістю виконувати перевірку коректності вхідних даних при введені значення максимальної швидкості автомобіля. Генерувати виняткову ситуацію якщо вводиться від’ємне значення швидкості або якщо значення швидкості перевищує 300 км/год.