Esercitazione di Fine Settimana – Week 6

Nome _____

Cognome _____

Data _____

Consegnare su GitHub. Nome del repository: NomeCognome_Prova6

ATTENZIONE: Le domande a risposta multipla possono contenere più risposte corrette.

1. Cosa ci si aspetta dal seguente codice?

```
try
{
        //codice che può scatenare o non scatenare un'eccezione
 }
 finally
 {
        //altro codice
 }
```

A) Indipendentemente dal verificarsi o meno di un'eccezione, il codice nel blocco finally non verrà mai raggiunto.
B) Se dovesse verificarsi un'eccezione, il codice nel blocco finally non verrà mai raggiunto.
C) Indipendentemente dal verificarsi o meno di un'eccezione, le istruzioni nella clausola finally verranno eseguite.
D) Se dovesse verificarsi un'eccezione, la clausola finally verrà raggiunta e potrebbe essere utilizzata per eseguire operazioni di 'pulizia' di eventuali risorse allocate in precedenza.

2. Dopo aver osservato il seguente codice, selezionare le affermazioni vere.

```
try
{

}
catch (Exception e)
{

}
catch (ArithmeticException a)
{

}
```

A) Il secondo blocco catch viene raggiunto solo se viene sollevata un'ArithmeticException
B) Il primo blocco catch verrà raggiunto per primo e intercetterà eccezioni di diversa natura, tra cui un'eventuale ArithmeticException
C) Qualora venga scatenata un'eccezione ArithmeticException, il secondo blocco catch non sarà mai raggiunto.
D) Il primo blocco catch non è in grado di gestire la ArithmeticException

3. Quali affermazioni, riguardante il codice seguente, sono vere?

```
using System;
namespace Test6
{
   class Program
   {
      static void Main(string[] args)
      {
         int index = 6;
```

```
            int[] myArray = new int[8];
            try
            {
               myArray[index] = 10;
            }
            catch (IndexOutOfRangeException e)
            {
               Console.WriteLine("Fuori dal range");
            }
        Console.WriteLine("Fine del programma");
          }
       }
    }
```

A) Il valore 10 sarà assegnato a myArray[6].
B) Verrà scatenata un'eccezione IndexOutOfRangeException.
C) Non ci sarà nessun output.
D) L'output sarà: Fine del programma.

Esercitazione Pratica

Realizzare un sistema di gestione di un'agenzia di assicurazione che si basi su:

- Un database Assicurazione (SQL Server), costituito dalle tabelle

o Clienti
Id (int, PK, auto-incrementale)
Codice fiscale (nchar(16), not nullable)
Nome (varchar(30))
Cognome (varchar(20))

o Polizze
Id (int, PK, auto-incrementale)
NumeroPolizza (int, not nullable)
DataScadenza (date)
RataMensile (decimal)
Tipo (RCAuto, Furto, Vita)

Il cliente può stipulare più polizze ma ogni polizza è 'personalizzata' per un cliente.

- Una Console app che consenta all'assicuratore di:

- Inserire un nuovo cliente
- Inserire una polizza per un cliente già esistente
- Visualizzare le polizze di un cliente
- Posticipare la data di scadenza (per es. quando si rinnova una polizza)
- Visualizzare i clienti che hanno una polizza vita (OPZIONALE)

VINCOLI TECNICI

- Utilizzare Entity Framework Core
- Utilizzare l'approccio Code-First e le Migrations

OPZIONALE: Implementare una o più delle funzionalità utilizzando ADO.NET (Connected mode)