# COS20007 - Object Oriented Programming

Pass Task 2.2: Counter Class and Arithmetic Overflow-checking

Show Wai Yan/105293041

Program.cs  
namespace CounterTask

{

internal class Program

{

private static void PrintCounters(Counter[] coutners)

{

foreach (Counter c in coutners)

{

Console.WriteLine("{0} is {1}",c.Name,c.Ticks);

}

}

public static void Main(string[] args)

{

Counter[] myCounters = new Counter[3];

myCounters[0] = new Counter("Counter 1");

myCounters[1] = new Counter("Counter 2");

myCounters[2] = myCounters[1];

for (int i = 1; i <= 9; i++)

{

myCounters[0].Increment();

}

for (int i = 1; i <= 14; i++)

{

myCounters[1].Increment();

}

Program.PrintCounters(myCounters);

myCounters[2].Reset();

Program.PrintCounters(myCounters);

}

}

}

### Counter.cs

namespace CounterTask

{

public class Counter

{

private int \_count;

private string \_name;

public Counter(string name)

{

\_name = name;

\_count = 0;

}

public void Increment()

{

\_count += 1;

}

public void Reset()

{

\_count = 0;

}

public void ResetByDefault()

{

// Answer for 13

// \_count = 2147483647041;

// above value create the crash(error) for the compile time

// due to overflow of integer which is \_count

// to run safetly (However, \_count will get unexcpeted value), we can use

// uncheckd block, which neglecte the overflow exception throw and

// use explictly type cast to int

// becuase 2147483647041 is bigger than int max value

unchecked

{

\_count = (int)2147483647041;

}

}

public string Name

{

get { return \_name; }

set { \_name = value; }

}

public int Ticks

{

get { return \_count; }

}

}

}

Screenshot of the console output

# A screenshot of a computer program AI-generated content may be incorrect.