```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace AlgorithmsHW3
{
    // Exercise 4.4 #10/A (cont.)
    class Pile
    {
        public int weight // property returns weight, cannot be set by user
        {
            get
            {
                return GetWeight(this.coins);
            }
            private set
            {
                weight = value;
            }
        }
        public List<Coin> coins { get; set; } // holds coins

        // initial pile constructor with fake coin creator
        public Pile(int size)
        {
            Random fakeCoinGen = new Random();
            int fakeCoinIndex = fakeCoinGen.Next(0, size - 1);
            this.coins = new List<Coin>();

            for (int i = 0; i < size; i++)
            {
                if (i == fakeCoinIndex)
                {
                    this.coins.Add(new Coin(false));
                }
                else
                {
                    this.coins.Add(new Coin(true));
                }
            }
        }
        // sub pile constructor, takes in parent pile and creates new pile given upper and lower bound
        public Pile(Pile parentPile, int upperBound, int lowerBound)
        {
            this.coins = new List<Coin>();

            for (int i = lowerBound; i < upperBound; i++)
            {
                this.coins.Add(parentPile.coins[i]);
            }
        }

        // Finds fake coin Guid
        public Guid FindCoin(Pile pile, int size)
        {
            if (pile.coins.Count == 1)
            {
                return pile.coins[0].id;
            }
            else
            {
                return ReturnPileContainingFake(pile).coins[0].id;
            }
        }
```

```
        // returns pile with fake coin
        private Pile ReturnPileContainingFake(Pile parent)
        {
            int lowerBound = 0;
            int upperBound = 0;
            Pile pileOne = null;
            Pile pileTwo = null;
            Pile pileThree = null;

            // first pile creation
            upperBound = parent.coins.Count / 3;
            pileOne = new Pile(parent, upperBound, lowerBound);
            lowerBound = upperBound;
            upperBound += upperBound;
            pileTwo = new Pile(parent, upperBound, lowerBound);
            // checks if third pile creation neccesary
            if (upperBound < parent.coins.Count)
            {
                lowerBound = upperBound;
                upperBound = parent.coins.Count;
                pileThree = new Pile(parent, upperBound, lowerBound);
            }

            if (pileOne.weight == pileTwo.weight)
            {
                return pileThree;
            }
            else if (pileTwo.weight < pileOne.weight)
            {
                return pileTwo;
            }

            return pileOne;
        }

        // counts coin weight to generate pile weight
        private int GetWeight(List<Coin> coins)
        {
            int weight = 0;

            for (int i = 0; i < coins.Count; i++)
            {
                weight += coins[i].weight;
            }

            return weight;
        }
    }
}
```