Assignment 14

Account.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace Assignment14

{

public class Account

{

static readonly object \_object = new object();

public int AccountNumber { get; set; }

public string CustomerName { get; set; }

public string CustomerAddress { get; set; }

private double balance { get; set; }

public Account()

{

}

public Account(int AccountNo,string custname,string custAddress,double amount)

{

AccountNumber = AccountNo;

CustomerName = custname;

CustomerAddress = custAddress;

balance = amount;

}

/// <summary>

/// Peform withdraw of a given amount

/// </summary>

/// <param name="Amount"></param>

public void Withdraw(double Amount)

{

try

{

Monitor.Enter(\_object);

if (balance >= Amount)

{

balance -= Amount;

Console.WriteLine("Amount " + Amount + " debited from the Account " + AccountNumber + " of " + CustomerName + " Remaining balance : " + balance);

}

else

{

Console.WriteLine(" Not a valid Transaction . Account " + AccountNumber + " don't have enough balance ");

}

}

catch (Exception ex)

{

Console.WriteLine("Exception Captured " + ex.Message);

}

finally

{

Monitor.Exit(\_object);

}

}

/// <summary>

/// Perform Deposit of the given Amount

/// </summary>

/// <param name="Amount"></param>

public void Deposit(double Amount)

{

try

{

Monitor.Enter(\_object);

if (Amount > 0d)

{

balance += Amount;

Console.WriteLine("Amount " + Amount + " Credited to the Account " + AccountNumber + " of " + CustomerName + " Remaining balance : " + balance);

}

else

{

Console.WriteLine(" Not a valid Transaction . Amount should be positive value");

}

}

catch (Exception ex)

{

Console.WriteLine("Exception Captured " + ex.Message);

}

finally

{

Monitor.Exit(\_object);

}

}

}

}

Program.cs

using System;

using System.Collections.Concurrent;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace Assignment14

{

class Program

{

static void Main(string[] args)

{

/\*

\* in this below example one account created.

\* Irrespective of transaction any operations can be possible based on the transaction amount.

\* It may be more than a one user performing operation based on the amount.

\* Decision will be taken based on the amount and new transaction will be initiated for the same.

\*

\* \*/

Account a = new Account(1, "Manoj Kumar P", "Cognizant", 5000);

ConcurrentBag<double> cb = new ConcurrentBag<double>();

cb.Add(500);

cb.Add(200);

cb.Add(847);

cb.Add(890);

cb.Add(238);

cb.Add(138);

cb.Add(143);

cb.Add(234);

cb.Add(908);

cb.Add(234);

double amount;

List<Task> tasks = new List<Task>();

int j = 0;

while (!cb.IsEmpty)

{

if (cb.TryPeek(out amount) && amount % 4 == 0)

{

cb.TryTake(out amount);

Thread t1 = new Thread(() => a.Deposit(amount));

t1.Start();

}

else

{

cb.TryTake(out amount);

Thread t2 = new Thread(() => a.Withdraw(amount));

t2.Start();

}

Thread.Sleep(200);

j += 1;

}

Console.ReadKey();

}

}

}

Output :

