Bahria University,

Karachi Campus



LAB EXPERIMENT NO.

08

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 1 | Create simple application for chain of responsibility among multiple threads. |
| 2 | Create an application for Task management in different USERS . it is noted that the tasks/workload on each user will be only 4 tasks at a time(Implement Chain of Responsibility to solve the allocation of tasks on different resources). |
|  |  |
|  |  |
|  |  |

Submitted On:

**Task No. 1:** Create simple application for chain of responsibility among multiple threads.

**Solution:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace COR

{

class LoanRequest

{

public string Customer { get; set; }

public decimal Amount { get; set; }

}

interface IRequestHandler

{

string Name { get; set; }

void HandleRequest(LoanRequest req);

IRequestHandler Successor { get; set; }

}

static class RequestHandlerExtension

{

public static void TrySuccessor(this IRequestHandler current, LoanRequest req)

{

if (current.Successor != null)

{

Console.WriteLine("{0} Can't approve - Passing request to {1}", current.Name, current.Successor.Name);

current.Successor.HandleRequest(req);

}

else

{

Console.WriteLine("Amount invaid, no approval given");

}

}

}

class Cashier : IRequestHandler

{

public string Name { get; set; }

public void HandleRequest(LoanRequest req)

{

Console.WriteLine("\n----\n{0} Rs Loan Requested by {1}",

req.Amount, req.Customer);

if (req.Amount < 1000)

Console.WriteLine("{0} Rs Loan approved for {1} - Approved by {2}",

req.Amount, req.Customer, this.Name);

else

this.TrySuccessor(req);

}

public IRequestHandler Successor { get; set; }

}

class Manager : IRequestHandler

{

public string Name { get; set; }

public void HandleRequest(LoanRequest req)

{

if (req.Amount < 10000)

Console.WriteLine("{0} Rs Loan approved for {1} - Approved by {2}",

req.Amount, req.Customer, this.Name);

else

this.TrySuccessor(req);

}

public IRequestHandler Successor { get; set; }

}

class Program

{

static void Main(string[] args)

{

var request1 = new LoanRequest() { Amount = 800, Customer = "Asad" };

var request2 = new LoanRequest() { Amount = 5000, Customer = "Bilal" };

var request3 = new LoanRequest() { Amount = 200000, Customer = "Hassan" };

var manager = new Manager() { Name = "Talha, Manager" };

var cashier = new Cashier() { Name = "Jalal, Cashier", Successor = manager };

cashier.HandleRequest(request1);

cashier.HandleRequest(request2);

cashier.HandleRequest(request3);

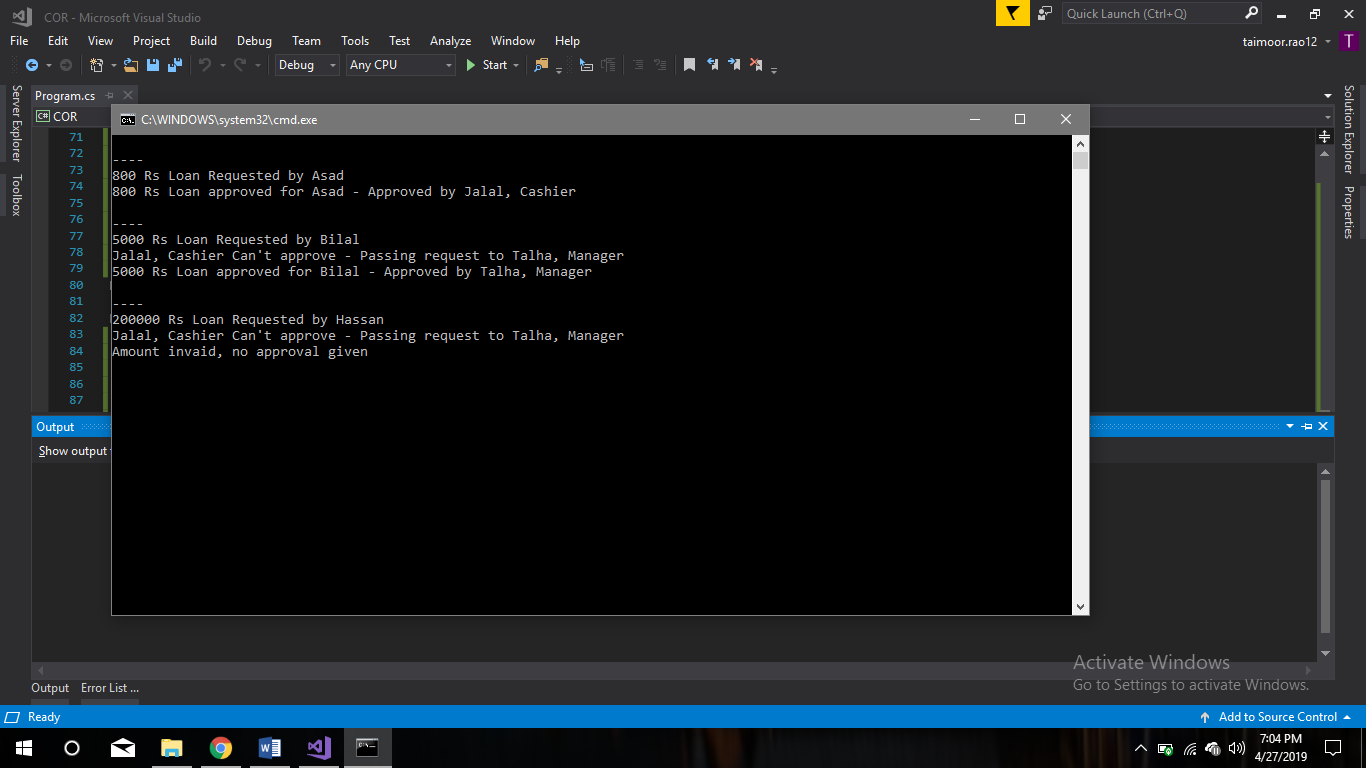
Console.ReadLine();

}

}

}

**Output:**



**Task No. 2:** Create an application for Task management in different USERS . it is noted that the tasks/workload on each user will be only 4 tasks at a time(Implement Chain of Responsibility to solve the allocation of tasks on different resources).

**Solution:**

using System;

namespace COR

{

class Program

{

static void Main(string[] args)

{

RequestHandler h1 = new Admin1();

RequestHandler h2 = new Admin2();

RequestHandler h3 = new Admin3();

RequestHandler h4 = new Admin4();

h1.SetSuccessor(h2);

h2.SetSuccessor(h3);

h3.SetSuccessor(h4);

int[] requests = { 2, 5, 14, 21, 12, 30, 8, 20 ,3};

foreach (int request in requests)

{

h1.HandleRequest(request);

}

Console.ReadKey();

}

}

abstract class RequestHandler

{

protected RequestHandler successor;

public void SetSuccessor(RequestHandler successor)

{

this.successor = successor;

}

public abstract void HandleRequest(int request);

}

class Admin1 : RequestHandler

{

public override void HandleRequest(int request)

{

if (request >= 0 && request < 5)

{

Console.WriteLine("{0} handled request {1}",

this.GetType().Name, request);

}

else if (successor != null)

{

successor.HandleRequest(request);

}

}

}

class Admin2 : RequestHandler

{

public override void HandleRequest(int request)

{

if (request >= 5 && request < 10)

{

Console.WriteLine("{0} handled request {1}",

this.GetType().Name, request);

}

else if (successor != null)

{

successor.HandleRequest(request);

}

}

}

class Admin3 : RequestHandler

{

public override void HandleRequest(int request)

{

if (request >= 10 && request < 15)

{

Console.WriteLine("{0} handled request {1}",

this.GetType().Name, request);

}

else if (successor != null)

{

successor.HandleRequest(request);

}

}

}

class Admin4 : RequestHandler

{

public override void HandleRequest(int request)

{

if (request >= 15 || request < 30)

{

Console.WriteLine("{0} handled request {1}",

this.GetType().Name, request);

}

else if (successor != null)

{

successor.HandleRequest(request);

}

}

}

}

**Output:**

