Bahria University,

Karachi Campus



LAB EXPERIMENT NO.

**09**

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| **01** | **You are working on a project where you get data from third party**  **webapi/service in XML format. You are creating charts from that data for creating a chart you need data in JSON format in order to make them compatiable you need to convert xml data into json but limitation is you can’t modify third party webservice/api you have**  **to make Adapter that returns data in JSON by using adapter pattern.** |
|  |  |
|  |  |

Submitted On:

**19 – 05 -2023**

(Date: DD/MM/YY)

**Task No. 1:** Implement Chain of Responsibility Design pattern on Processing Scenario. Consider that there are multiple processors having multiple responsibilities to handle (ex: media player file is played by media player handler, calculation is carried out by calculation handler, etc…) initiate handling for opening internet browser and notepad by creating multiple handlers with different responsibilities.

**Solution:**

**Abstract Class AbstractHandler**

abstract class AbstractHandler

{

protected AbstractHandler ah;

public void setsuccessor(AbstractHandler ah)

{

this.ah = ah;

}

public abstract void process\_request(ProcesserDirection pd);

}

**Class Processer Direction**

class ProcesserDirection

{

String processName;

String firstCharacter;

public ProcesserDirection(String processName, String firstCharacter)

{

this.firstCharacter = firstCharacter;

this.processName = processName;

}

public string pname

{

get

{

return processName;

}

set

{

this.processName = value;

}

}

public string fchar

{

get

{

return firstCharacter;

}

set

{

this.firstCharacter = value;

}

}

}

**Class NotePadHandler**

class NotePadHandler : AbstractHandler

{

public override void process\_request(ProcesserDirection pd)

{

if (pd.fchar == "N")

{

Console.WriteLine("{0}Request Handeled By {1}", pd.pname, this.GetType().Name);

}

else if (ah != null)

{

ah.process\_request(pd);

}

}

}

**Class Browserhandler**

class browserHandler : AbstractHandler

{

public override void process\_request(ProcesserDirection pd)

{

if (pd.fchar == "B")

{

Console.WriteLine("{0} Request Handeled By {1}", pd.pname, this.GetType().Name);

}

else if (ah != null)

{

ah.process\_request(pd);

}

}

}

**Class MemoryFull**

class MemoryFull : AbstractHandler

{

public override void process\_request(ProcesserDirection pd)

{

Console.WriteLine("{0} Request not Handeled Error {1}", pd.pname, this.GetType().Name);

}

}

**Main Method**

static void Main(string[] args)

{

AbstractHandler abstracthandler1 = new NotePadHandler();

AbstractHandler abstracthandler2 = new browserHandler();

AbstractHandler abstracthandler3 = new MemoryFull();

abstracthandler1.setsuccessor(abstracthandler2);

abstracthandler2.setsuccessor(abstracthandler3);

Console.WriteLine("Enter process first char: \n");

string choice = Console.ReadLine();

ProcesserDirection pd;

if (choice == "N")

{

pd = new ProcesserDirection("NotePad Started", choice);

abstracthandler1.process\_request(pd);

}

else if (choice == "B")

{

pd = new ProcesserDirection("browser Started", choice);

abstracthandler1.process\_request(pd);

}

else

{

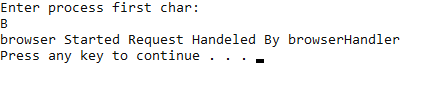
pd = new ProcesserDirection("Error", choice);

abstracthandler3.process\_request(pd);

}

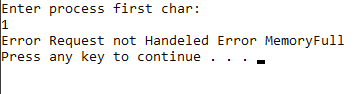
}

**Output:**

****

A black text on a white background

Description automatically generated with low confidence



**Task No. 2:** Implement COR on call center scenario.

**Solution:**

**Class Approver**

abstract class Approver

{

public Approver approver;

public void setSuccessor(Approver approver)

{

this.approver = approver;

}

public abstract void processRequest(callChecker purchase);

}

**Class CallChecker**

class callChecker

{

int call\_ID;

public callChecker(int call\_ID)

{

this.call\_ID = call\_ID;

}

public int id

{

get { return call\_ID; }

set { call\_ID = value; }

}

}

**Class ServiceMan1**

class ServiceMan1 : Approver

{

public override void processRequest(callChecker CC)

{

if (CC.id < 1001)

{

Console.WriteLine("{0} will attend call request No. {1} ", this.GetType().Name, CC.id);

}

else if (approver != null)

{

approver.processRequest(CC);

}

}

} **Class ServiceMan2**

class ServiceMan2 : Approver

{

public override void processRequest(callChecker CC)

{

if (CC.id < 2001)

{

Console.WriteLine("{0} will attend call request No. {1} ", this.GetType().Name, CC.id);

}

else if (approver != null)

{

approver.processRequest(CC);

}

}

}

**Class ServiceMan3**

class ServiceMan3 : Approver

{

public override void processRequest(callChecker CC)

{

if (CC.id < 5001)

{

Console.WriteLine("{0} will attend call request No. {1}", this.GetType().Name, CC.id);

}

else

{

Console.WriteLine("Request {0} requires an Special member to approve", CC.id);

}

}

}

**Main Method**

static void Main(string[] args)

{

Approver s1 = new ServiceMan1();

Approver s2 = new ServiceMan2();

Approver s3 = new ServiceMan3();

s1.setSuccessor(s2);

s2.setSuccessor(s3);

callChecker cC = new callChecker(901);

s1.processRequest(cC);

cC = new callChecker(1051);

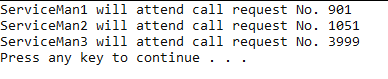
s1.processRequest(cC);

cC = new callChecker(3999);

s1.processRequest(cC);

}

**Output:**

****