**Exercise: Design a workflow engine**

**Design a workflow engine that takes a workflow object and runs it. A workflow is a series of steps**

**or activities. The workflow engine class should have one method called Run() that takes a**

**workflow, and then iterates over each activity in the workflow and runs it.**

**We want our workflows to be extensible, so we can create new activities without impacting the existing activities.**

**Educational tip: we should represent the concept of an activity using an interface. Each activity should have a method called Execute(). The workflow engine does not care about the concrete implementation of activities. All it cares about is that these activities have a common interface: they provide a method called Execute(). The engine simply calls this method and this way it executes a series of activities in sequence.**

**The aim of this exercise is to help you understand how you can use interfaces to design**

**extensible applications. You change the behaviour of your application by creating new classes, rather than changing the existing classes. You’ll also see polymorphic behaviour of interfaces.**

**Ans:**

public interface IActivity

{

void Execute();

}

public interface IWorkflow

{

void RegisterActivity(IActivity activityObj);

List<IActivity> GetAllActivities(); //return all the activities registered using the above

}

public class uploadvideo : IActivity //this should be repeated for all the activites

{

public void Execute()

{

Console.WriteLine("Uploading video...");

}

}

public class callingWebService : IActivity //this should be repeated for all the activites

{

public void Execute()

{

Console.WriteLine("callingWebService...");

}

}

public class emailSending : IActivity //this should be repeated for all the activites

{

public void Execute()

{

Console.WriteLine("sending email....");

}

}

public class changeStatus : IActivity //this should be repeated for all the activites

{

public void Execute()

{

Console.WriteLine("Changing status...Processing Video....");

}

}

class workflow : IWorkflow

{

private List<IActivity> aList = new List<IActivity>();

public void RegisterActivity(IActivity activityObj)

{

aList.Add(activityObj);

}

public List<IActivity> GetAllActivities()

{

//retun all the activities registerd. Activity object

return aList;

}

}

class WorkFLowEngine

{

public void Run(IWorkflow obj)

{

List<IActivity> aList = obj.GetAllActivities();

//iterate the workflow obj and call the corresponding activites Execute method

foreach (IActivity activity in aList)

{

activity.Execute();

}

}

//private List<IActivity> aList = obj.GetAllActivities();

//foreach (var act in )

}

class Program

{

public static void Main(string[] args)

{

//create object for workflowengine

WorkFLowEngine wrk = new WorkFLowEngine();

//create objcet for workflow

workflow wrkf = new workflow();

//call the registeractivity method of workflow by passing the activity as a parameter

uploadvideo act1 = new uploadvideo();

callingWebService act2 = new callingWebService();

emailSending act3 = new emailSending();

changeStatus act4 = new changeStatus();

wrkf.RegisterActivity(act1);

wrkf.RegisterActivity(act2);

wrkf.RegisterActivity(act3);

wrkf.RegisterActivity(act4);

//call register for all the activity

List<IActivity> activities = wrkf.GetAllActivities();

/\*foreach (IActivity activity in activities)

{

Console.WriteLine(activity);

}\*/

//call the workflowengine class run method by passing the workflow object created above

wrk.Run(wrkf);

}

}

**Text

Description automatically generated**