**Event -Delegate Example**

using System;

namespace DelegateEventExpl

{

public delegate void PayamentNotification(object sender, EventArgs e);

public class SSKPayament

{

public event PayamentNotification onPaymentCompleted;

public void doPayment()

{

Console.WriteLine("Do payament");

if(onPaymentCompleted!=null)

{

onPaymentCompleted(this, EventArgs.Empty);

}

}

}

public class SMSNotifier

{

public void sendSMS(object sender, EventArgs e)

{

Console.WriteLine("Send Sms");

}

}

public class EmailNotifier

{

public void sendEmail(object sender, EventArgs e)

{

Console.WriteLine("Send Email");

}

}

class Program

{

static void Main(string[] args)

{

SSKPayament sSKPayament = new SSKPayament();

EmailNotifier emailNotifier = new EmailNotifier();

SMSNotifier sMSNotifier = new SMSNotifier();

sSKPayament.onPaymentCompleted += emailNotifier.sendEmail;

sSKPayament.onPaymentCompleted += sMSNotifier.sendSMS;

Console.WriteLine("Hello World!");

sSKPayament.doPayment();

}

}

}

// Here we don’t need to add code in SSKpayament class if further we want to mode functionality , only we need to attach event of same signature.

**Only Delegate**

using System;

using System.Collections.Generic;

namespace DelegateEventExpl

{

// Suppose we have Employee class (IBM and there are no of verndor and we want to promote those employee

// but final check is with IBM , so we will not add condition in Employee class to promote employee of each types of vendor

// condition of promotion will depends upon the Vendors and they will pass the list of employee to IBM's mail Employee

// class where will trigger condition and prmote those employee as per differnt vendor 's condition defined in differnt vendor

// class

class Program

{

public delegate bool isPromotionApplicable(Employee e);

public class Employee

{

public int ID { get; set; }

public string Name { get; set; }

public int Salary { get; set; }

public int Exp { get; set; }

public void Promoteemployee(List<Employee> empList , isPromotionApplicable isPromotion)

{

foreach(Employee e in empList)

{

if(isPromotion(e))

{

Console.WriteLine("Promoted >" + e.Name);

}

}

}

}

public class outworks

{

public List<Employee> outworksEmployees;

public outworks()

{

outworksEmployees = new List<Employee>()

{

new Employee { ID=1, Exp=10, Name="A", Salary=1000},

new Employee { ID=1, Exp=10, Name="b", Salary=2000},

new Employee { ID=1, Exp=10, Name="c", Salary=5000},

new Employee { ID=1, Exp=10, Name="d", Salary=4000},

};

}

public bool isPromotionApplication(Employee e)

{

return e.Salary < 4000 ? true : false;

}

}

public class WNS

{

public List<Employee> WNSEmployees;

public WNS()

{

WNSEmployees = new List<Employee>()

{

new Employee { ID=1, Exp=10, Name="WA", Salary=1000},

new Employee { ID=1, Exp=11, Name="Wb", Salary=2000},

new Employee { ID=1, Exp=12, Name="Wc", Salary=5000},

new Employee { ID=1, Exp=13, Name="Wd", Salary=4000},

};

}

public bool isPromotionApplication(Employee e)

{

return e.Exp >11 ? true : false;

}

}

static void Main(string[] args)

{

outworks ou = new outworks();

Employee employee = new Employee();

employee.Promoteemployee(ou.outworksEmployees, ou.isPromotionApplication);

WNS wNS = new WNS();

employee.Promoteemployee(wNS.WNSEmployees, wNS.isPromotionApplication);

}

}

}