BSEF19M012 IQRA SARWAR
OOAD ASSIGNMENT # 03

CLASS DEFINATIONS:

BlogPosts

```
using System;
using Admin;
using Validate;
using System.IO;

namespace blogPosts
{
        public class BlogPosts
        {
                private string Video;
                private readonly DateTime dateTime = DateTime.Now;
                private string Status;
                private string      BlogPostid;
                private string content;
                private string Image;
                private Admin admin;
                private NewsFeed feed;

                BlogPosts();
                public void RequestUpload();
                public void RequestModification();
                public void DeletePost();
                public void ApproveUpload();

        }
};
```

Calls

```
using System;
using System.Collections.Generic;

namespace Calls
{
        class Calls
        {
                private readonly DateTime dateTime = DateTime.Now;
                private string Duration;
                private string SNumber;
                private string Receipent;
                private string Callid;
                private static List<Calls> Log = new List<Calls>();
                private static List<string> blocked = new List<string>();

                Calls();
                public void NewCall();
                public void ReceiveCall(string caller);
                public void AddNewCallToLog();
                public void DeleteCallFromLog(string CallId);
```

```
            public void ClearLog();
            public void BlockCaller(string Callid);
        };
}
```

Communications

```
using System;
using Email;
using Calls;
using BlogPosts;
using Meetings;

namespace communication
{
        public class Communication
        {
                public string SenderId;
                public string ReceiverId;
                public string Type;

                private void CommunicateViaEmail(string TouristId, string
ServiceProviderId, Email emailInstance);
                private void CommunicateViaMessages(string TouristId, string
ServiceProviderId, Message MessagesInstance);
                private void CommunicateViaCalls(string TouristId, string
ServiceProviderId, , Calls CallsInstance);
                private void CommunicateViaMeetings(string TouristId, string
ServiceProviderId, Meetings MeetingsInstance);
        };
}
```

Email

```
using System;
using System.Collections.Generic;
using EmailAPI;

namespace email
{
    class Email
    {
        private readonly DateTime dateTime = DateTime.Now;
        private string Content;
        private string Receipent;
        private readonly string myEmail = "mymail@gmail.com";
        private string EmailId;
        private static List<Email> drafts = new List<Email>();
        private static List<Email> received = new List<Email>();
        private static List<Email> sent = new List<Email>();

        Email();
        public void NewEmail();
        public void MakeDraft();
        public void SendEmail(string Content, string Receipent);
        public void ReceiveEmail();
        public void DiscardDraft(string Emailid);
        public void DeleteEmail(string EmailId);
        public void ReportEmail(string EmailId);
    };
}
```

Meetings

```
using System;
using System.Collections.Generic;
using MeetAPI;

namespace meetings
{
       class Meetings
       {
              private DateTime dateTime;
              private string Duration;
              private string Receipent;
              private List<string> Participant;
              private static List<Meetings> scheduledMeets = new List<Meetings>();
              private string Topic;
              private string MeetId;

              Meetings();
              public void ScheduleNewMeeting();
              public void AttendMeet(string MeetId);
              public void cancelMeet(string MeetId);
              public void SendInvitation(List<string> Participants, string MeetId);
       };
}
```

Message

```
using System;
using System.Collections.Generic;
using MsgAPI;

namespace message
{
       class Message
       {
              private readonly DateTime dateTime = DateTime.Now;
              private string Content;
              private string Receipent;
              private string MsgId;
              private static list<Message> drafts = new list<Message>();
              private static list<Message> sent = new list<Message>();
              private static list<Message> received = new list<Message>();

              Message();
              public void NewMessage();
              public void MakeDraft();
              public void SendMessage(string Content, string Receipent);
              public void ReceiveMessage(string MsgId);
              public void DiscardDraft(string MsgId);
              public void DeleteMsg(string MsgId);
```

```
		public void ReportMsg(string MsgId);
	};
}
```

Pictures

```
using System;
using Admin;
using Validate;
using System.IO;

namespace pictures
{
	class Pictures
	{
		private string Image;
		private string imgId;
		private readonly DateTime dateTime = DateTime.Now;
		private string Status;
		private Admin admin;
		private NewsFeed feed;

		Pictures();
		public void RequestModification();
		public void DeletePicture();
		public void ApproveUpload();
		public void RequestUpload();
	};
}
```

Videos

```
using System;
using Admin;
using Validate;
using System.IO;

namespace Video
{
	class Video
	{

		private string Video;
		private string vidId;
		private readonly DateTime dateTime = DateTime.Now;
		private string Status;
		private Admin admin;
		private NewsFeed feed;

		Video();
		public void DeleteVideo();
		public void ApproveUpload();
		public void RequestModification();
		public void RequestUpload();
	};
}
```

METHOD DEFINATIONS:

BlogPosts

```
using System;
using Admin;
using Validate;
using System.IO;

namespace blogPosts
{
       public class BlogPosts
       {
              private string Video { get; set; }
              private readonly DateTime dateTime = DateTime.Now;
              private string Status { get; set; }
              private string      BlogPostid { get; set; }
              private string content { get; set; }
              private string Image { get; set; }
              private Admin admin { get; set; }
              private NewsFeed feed { get; set; }
              DateTime dateTime { get; }

              BlogPosts()
        {
                     static int id= 1;
                     this.BlogPostid = Convert.ToString(id);
                     id++;
                     Status = false;
                     Console.WriteLine("Enter Video address");
                     Video = Console.ReadLine();
                     Console.WriteLine("Enter Image address");
                     Image = Console.ReadLine();
                     Console.WriteLine("Enter post content");
                     Content = Console.ReadLine();
                     admin = new Admin();
                     feed = new NewsFeed();
              }
              public void RequestUpload()
        {
                     if (Validate.isvalid(this))
                     {
                            admin.ApproveBlogPost(this);
                            if (Status == true)
                                   feed.AddBlogPost(this);
                            else
                            {
                                   Console.WriteLine("Admin Approval denied!");
                                   DeletePost(this);
                            }
                     }
                     else
                     {
                            Console.WriteLine("In appropriate Content! Validation 
falid.");
                            DeletePost(this);
                     }
        }
              public void RequestModification()
        {
```

```
                    BlogPosts temp = new BlogPosts();
                    Console.WriteLine("Enter new video address");
                    temp.Video = Console.ReadLine();
                    Console.WriteLine("Enter new image address");
                    temp.Image = Console.ReadLine();
                    Console.WriteLine("Enter new post content");
                    temp.Content = Console.ReadLine();
                    if (Validate.isvalid(temp))
                    {
                           admin.ApproveBlogPost(temp);
                           if (Status == true)
               {
                                  feed.AddBlogPost(temp);
                                  feed.DeleteBlogPost(this);
                           }
                           else
                           {
                                  Console.WriteLine("Admin Approval denied!");
                                  DeletePost(temp);
                           }
                    }
                    else
                    {
                           Console.WriteLine("In appropriate Content! Validation
falid.");
                           DeletePost(temp);
                    }
             }
             public void DeletePost()
       {
                    feed.DeleteBlogPost(this);
             }
             public void ApproveUpload()
       {
                    this.Status = true;
       }
      }
};
```

Calls

```
using System;
using System.Collections.Generic;
using CallingAPI;

namespace Calls
{
       class Calls
       {
              private readonly DateTime dateTime = DateTime.Now;
              private string Duration { get; set; }
              private string SNumber { get; set; }
              private string Receipent { get; set; }
              private string Callid { get; set; }
              private static List<Calls> Log = new List<Calls>();
              private static List<string> blocked = new List<string>();
              DateTime dateTime { get; }
```

```
Calls()
{
        Duration = "";
        SNumber = "03xx xxxxxxx";
        Receipent = "";
        Callid = "";
        static int id = 1;
        this.Callid = Convert.ToString(id);
        id++;
}
public void NewCall()
{
        Console.WriteLine("enter Contact Number");
        Receipent = Console.ReadLine();
        foreach (string s in blocked)
        {
                if (Receipent == s)
                {
                        Console.Write("Can't make call to blocked contact");
                        return;
                }
        }
        this.AddNewCallToLog();
        CallingAPI.intiateNew(SNumber, Receipent);
}
public void ReceiveCall(string caller)
{
        foreach (string s in blocked)
        {
                if (caller == s)
                {
                        return;
                }
        }
        this.AddNewCallToLog();
        callingAPI.initiateNew(caller, SNumber);
}
public void AddNewCallToLog()
{
        Log.Add(this);
}
public void DeleteCallFromLog(string CallId)
{
        foreach (Calls call in Log)
        {
                if (Log[call].Callid == CallId)
                {
                        Log.Remove(Log[call]);
                }
        }
}
public void ClearLog()
{
        Log.Clear();
}
public void BlockCaller(string Callid)
{
```

```
                    foreach (Calls call in Log)
                    {
                           if (Log[call].Callid == CallId)
                           {
                                  string receient = Log[call].Receipent;
                                  blocked.Add(receient);
                           }
                    }
             }
      };
                                        }
```

Communication

```
using System;
using Email;
using Calls;
using BlogPosts;
using Meetings;

namespace communication
{
       public class Communication
       {
              private string SenderId { get; set; }
              private string ReceiverId { get; set; }
              private string Type { get; set; }

              public void CommunicateViaEmail(string TouristId, string ServiceProviderId,
Email emailInstance, int direction)
        {
                     Type = "Email";
                     if (direction == 1)
                     {
                            SenderId = TouristId;
                            ReceiverId = ServiceProviderId;
                            emailInstance.Recepient = Tourist.getEmail(TouristId);
                            emailInstance.myEmail =
ServiceProvider.getEmail(ServiceProviderId);
                            emailInstance.SendEmail()
                     }
                     else
                     {
                            SenderId = ServiceProviderId;
                            ReceiverId = TouristId;
                            emailInstance.Recepient =
ServiceProvider.getEmail(ServiceProviderId);
                            emailInstance.myEmail = Tourist.getEmail(TouristId);
                            emailInstance.SendEmail()
                     }
              }
              public void CommunicateViaMessages(string TouristId, string
ServiceProviderId, Message MessagesInstance, int direction)
        {
                     Type = "Message";
                     if (direction == 1)
                     {
```

```
                                SenderId = TouristId;
                                ReceiverId = ServiceProviderId;
                                MessagesInstance.Recepient = Tourist.getEmail(TouristId);
                                MessagesInstance.SendMessage();
                        }
                        else
                        {
                                SenderId = ServiceProviderId;
                                ReceiverId = TouristId;
                                MessagesInstance.Recepient =
ServiceProvider.getEmail(ServiceProviderId);
                                MessagesInstance.SendMessage();
                        }
                }
                public void CommunicateViaCalls(string TouristId, string ServiceProviderId,
Calls CallsInstance, int direction)
        {
                        Type = "Call";
                        if (direction == 1)
                        {
                                SenderId = TouristId;
                                ReceiverId = ServiceProviderId;
                                CallsInstance.Recepient = Tourist.getNumber(TouristId);
                                CallsInstance.SNumber =
ServiceProvider.getNumber(ServiceProviderId);
                                CallsInstance.NewCall();
                        }
                        else
                        {
                                SenderId = ServiceProviderId;
                                ReceiverId = TouristId;
                                CallsInstance.Recepient =
ServiceProvider.getNumber(ServiceProviderId);
                                CallsInstance.SNumber = Tourist.getNumber(TouristId);
                                CallsInstance.NewCall();
                        }
                }
                public void CommunicateViaMeetings(string TouristId, string
ServiceProviderId, Meetings MeetingsInstance, int direction)
        {
                        Type = "Meet";
                        if (direction == 1)
                        {
                                SenderId = TouristId;
                                ReceiverId = ServiceProviderId;
                                MeetInstance.Recepient = Tourist.getEmail(TouristId);
                                MeetInstance.ScheduleNewMeeting()
                        }
                        else
                        {
                                SenderId = ServiceProviderId;
                                ReceiverId = TouristId;
                                MeetInstance.Recepient =
ServiceProvider.getEmail(ServiceProviderId);
                                MeetInstance.ScheduleNewMeeting()
                        }
                }
        };
```

```
}
```

Email

```
using System;
using System.Collections.Generic;
using EmailAPI;

namespace email
{
    class Email
    {
        private readonly DateTime dateTime = DateTime.Now;
        private string Content { get; set; }
        private string Receipent { get; set; }
        private readonly string myEmail = "mymail@gmail.com";
        private string EmailId { get; set; }
        private static List<Email> drafts = new List<Email>();
        private static List<Email> received= new List<Email>();
        private static List<Email> sent = new List<Email>();
        DateTime dateTime { get; }
        string myEmail { get; }

        Email()
        {
            static int id = 1;
            this.EmailId = Convert.ToString(id);
            id++;
            Content = "";
            Receipent = "";
        }
        public void NewEmail()
        {
            Console.WriteLine("Enter email content");
            this.Content = Console.ReadLine();
            Console.WriteLine("Enter recepient email");
            this.Receipent = Console.ReadLine();
        }
        public void MakeDraft()
        {
            Email temp;
            Console.WriteLine("Enter email content");
            temp.Content = Console.ReadLine();
            Console.WriteLine("Enter recepient email");
            temp.Receipent = Console.ReadLine();
            drafts.Add(temp);
        }
        public void SendEmail(string Content, string Receipent)
        {
            NewEmail();
            EmailAPI.initiateNew(this.myEmail, Receipent, Content);
            sent.Add(this);
        }
        public void ReceiveEmail()
        {
            NewEmail();
            EmailAPI.initiateNew(Receipent, this.myEmail, Content);
            received.Add(this);
        }
        public void DiscardDraft(string Emailid)
        {
            foreach(Email e in drafts)
            {
                if(drafts[e].EmailId == EmailId)
                {
                    drafts.Remove(drafts[e]);
                }
            }
        }
        public void DeleteEmail(string EmailId)
        {
            foreach (Email e in received)
            {
                if (received[e].EmailId == EmailId)
                {
                    drafts.Remove(received[e]);
                }
            }
            foreach (Email e in sent)
            {
                if (sent[e].EmailId == EmailId)
                {
                    drafts.Remove(sent[e]);
                }
            }
        }
        public void ReportEmail(string EmailId)
        {
            Console.WriteLine("Enter Complain");
            string complain = Console.ReadLine();
            EmailAPI.initiatReport(EmailId, complain);
        }
    };
}
```

Meetings

```
using System;
using System.Collections.Generic;
using MeetAPI;

namespace meetings
{
       class Meetings
       {
              private DateTime dateTime { get; set; }
              private string Duration { get; set; }
              private string Receipent { get; set; }
              private List<string> Participant { get; set; }
              private static List<Meetings> scheduledMeets = new List<Meetings>();
              private string Topic { get; set; }
              private string MeetId { get; set; }

              Meetings()
              {
                     static int id = 1;
                     this.MeetId = Convert.ToString(id);
                     id++;
```

```
                Topic = "";
                Receipent = "";
                Participant = new List<string>();
                Duration = "";
            }
            public void ScheduleNewMeeting()
        {
                Meetings meet;
                Console.WriteLine("Enter meeting time");
                dateTime = Convert.ToDateTime(Console.ReadLine());
                Console.WriteLine("enter the participants or 0 to exist");
                int end = 1;
                while(end != 0)
            {
                    end = Console.ReadLine();
                    if (end != "0")
                        Participant.Add();
                    else
                        break;
            }
                Console.WriteLine("Enter meeting topic");
                Topic = Console.ReadLine();
                Console.WriteLine("Enter expected duration");
                Duration = Console.ReadLine();
                Console.WriteLine("Enter meet guest email");
                Receipent = Console.ReadLine();
                scheduledMeets.Add(meet);
            }
            public void AttendMeet(string MeetId)
        {
                foreach(Meetings m in scheduledMeets)
            {
                    if (scheduledMeets[m].MeetId == MeetId)
                    {
                        if (scheduledMeets[m].dateTime == DateTime.now)
                        {
                            MeetAPI.intiateNew(scheduledMeets[m]);
                        }
                        else
                        {
                            Console.WriteLine("Meet isn't scheduled for this
    time");

                        }
                    }
            }
        }
            public void cancelMeet(string MeetId)
        {
                foreach (Meetings m in scheduledMeets)
                {
                    if (scheduledMeets[m].MeetId == MeetId)
                    {
                        scheduledMeets.Remove(scheduledMeets[m]);
                    }
                }
            }
            public void SendInvitation(List<string> Participants, string MeetId)
        {
```

```
                        this.ScheduleNewMeeting();
                        MeetAPI.initateInvites(this);
                }
        };
}
```

Message

```
using System;
using System.Collections.Generic;
using MsgAPI;

namespace message
{
        class Message
        {
                private readonly DateTime dateTime = DateTime.Now;
                private string Content { get; set; }
                private string Receipent { get; set; }
                private string MsgId { get; set; }
                private static list<Message> drafts = new list<Message>();
                private static list<Message> sent = new list<Message>();
                private static list<Message> received = new list<Message>();
                DateTime dateTime { get; }

                Message()
                {
                        static int id = 1;
                        this.MsgId = Convert.ToString(id);
                        id++;
                        Content = "";
                        Receipent = "";
                }
                public void NewMessage()
          {
                        Console.WriteLine("Enter message content");
                        this.Content = Console.ReadLine();
                        Console.WriteLine("Enter recepient username");
                        this.Receipent = Console.ReadLine();
                }
                public void MakeDraft()
          {
                        Message temp;
                        Console.WriteLine("Enter message content");
                        temp.Content = Console.ReadLine();
                        Console.WriteLine("Enter recepient message");
                        temp.Receipent = Console.ReadLine();
                        drafts.Add(temp);
                }
                public void SendMessage(string Content, string Receipent)
          {
                        NewMessage();
                        MsgAPI.initiateNew(Receipent, Content);
                        sent.Add(this);
                }
                public void ReceiveMessage(string MsgId)
          {
                        NewMessage();
                        MsgAPI.initiateNew(this.MsgId, Content);
                        received.Add(this);
                }
                public void DiscardDraft(string MsgId)
        {
                        foreach (Message m in drafts)
                        {
                                if (drafts[m].MsgId == MsgId)
                                {
                                        drafts.Remove(drafts[m]);
                                }
                        }
                }
                public void DeleteMsg(string MsgId)
        {
                        foreach (Message e in received)
                        {
                                if (received[e].MsgId == MsgId)
                                {
                                        drafts.Remove(received[e]);
                                }
                        }
                        foreach (Message e in sent)
                        {
                                if (sent[e].MsgId == MsgId)
                                {
                                        drafts.Remove(sent[e]);
                                }
                        }
                }
                public void ReportMsg(string MsgId)
        {
                        Console.WriteLine("Enter Complain");
                        string complain = Console.ReadLine();
                        MsgAPI.initiatReport(MsgId, complain);
                }
        };
}
```

Pictures

```
using System;
using Admin;
using Validate;
using System.IO;

namespace pictures
{
        class Pictures
        {
                private string Image { get; set; }
                private string imgId { get; set; }
                private readonly DateTime dateTime = DateTime.Now;
                private string Status { get; set; }
                private Admin admin { get; set; }
                private NewsFeed feed { get; set; }
                DateTime dateTime { get; }

                Pictures()
                {
                    static int id = 1;
                    this.imgId = Convert.ToString(id);
                    id++;
                    Status = false;
                    Console.WriteLine("Enter Image address");
                    Image = Console.ReadLine();
                    admin = new Admin();
                    feed = new NewsFeed();
            }
            public void RequestModification(string imgId)
        {
                    Pictures temp = new Pictures();
                    Console.WriteLine("Enter new image address");
                    temp.Image = Console.ReadLine();
                    if (Validate.isvalid(temp))
                    {
                            admin.ApprovePicture(temp);
                            if (Status == true)
                            {
                                    feed.AddPicture(temp);
                                    feed.DeletePicture(this);
                            }
                            else
                            {
                                    Console.WriteLine("Admin Approval denied!");
                                    DeletePicture(temp);
                            }
                    }
                    else
                    {
                            Console.WriteLine("In appropriate Content! Validation
falid.");

                            DeletePicture(temp);
                    }
            }
            public void DeletePicture(string imgId)
        {
                    feed.DeletePicture(this);
            }
            public void ApproveUpload(string imgId)
        {
                    this.Status = true;
            }
            public void RequestUpload(string imgId)
        {
                    if (Validate.isvalid(this))
                    {
                            admin.ApprovePicture(this);
                            if (Status == true)
                                    feed.AddPicture(this);
                            else
                            {
                                    Console.WriteLine("Admin Approval denied!");
                                    DeletePicture(this);
                            }
                    }
                    else
                    {
```

```
                                Console.WriteLine("In appropriate Content! Validation 
falid.");
                                DeletePicture(this);
                        }
                }
        };
}
```

Videos

```
using System;
using Admin;
using Validate;
using System.IO;

namespace Video
{
        class Video
        {

                private string Video { get; set; }
                private string vidId { get; set; }
                private readonly DateTime dateTime = DateTime.Now;
                private string Status { get; set; }
                private Admin admin { get; set; }
                private NewsFeed feed { get; set; }
                DateTime dateTime { get; }

         Video()
         {
                        static int id = 1;
                        this.vidId = Convert.ToString(id);
                        id++;
                        Status = false;
                        Console.WriteLine("Enter video address");
                        Image = Console.ReadLine();
                        admin = new Admin();
                        feed = new NewsFeed();
                }
                public void DeleteVideo(string vidId)
         {
                        feed.DeleteVideo(this);
                }
                public void ApproveUpload(string vidId)
         {
                        this.Status = true;
                }
                public void RequestModification(string vidId)
                {
                        Video temp = new Video();
                        Console.WriteLine("Enter new video address");
                        temp.Video = Console.ReadLine();
                        if (Validate.isvalid(temp))
                        {
                                admin.ApproveVideo(temp);
                                if (Status == true)
                                {
                                        feed.AddVideo(temp);
                                        feed.DeleteVideo(this);
                        }
                        else
                        {
                            Console.WriteLine("Admin Approval denied!");
                            Deletevideo(temp);
                        }
                    }
                    else
                    {
                        Console.WriteLine("In appropriate Content! Validation 
falid.");
                        DeleteVideo(temp);
                    }
                }
                public void RequestUpload(string vidId)
                {
                    if (Validate.isvalid(this))
                    {
                        admin.ApproveVideo(this);
                        if (Status == true)
                            feed.AddVideo(this);
                        else
                        {
                            Console.WriteLine("Admin Approval denied!");
                            DeleteVideo(this);
                        }
                    }
                    else
                    {
                        Console.WriteLine("In appropriate Content! Validation 
falid.");
                        DeleteVideo(this);
                    }
                }
        };
}
```

<u>CLASS EXCEPTIONS:</u>

BlogPosts

```
using System;
using Admin;
using Validate;
using System.IO;

namespace blogPosts
{
        public class BlogPosts
        {
                //possible exceptions that can occur in the following methods. Refer to 
their code to  see how they can occur.

                BlogPosts()
                {
                        System.IO.IOException;
                        System.BadImageFormatException;
                        System.InvalidCastException;
```

```
              }
              public void RequestUpload()
              {
                     System.IO.IOException;
                     System.FieldAccessException;
                     System.AccessViolationException;
                     System.MethodAccessException;
                     System.ObjectDisposedException;
                     System.UnauthorizedAccessException;
              }

              public void RequestModification()
              {
                     System.IO.IOException;
                     System.FieldAccessException;
                     System.AccessViolationException;
                     System.MethodAccessException;
                     System.ObjectDisposedException;
                     System.UnauthorizedAccessException;
              }
              public void DeletePost()
              {
                     System.AccessViolationException;
                     System.MethodAccessException;
                     System.ObjectDisposedException;
              }
              public void ApproveUpload()
              {
                     System.AccessViolationException;
                     System.MethodAccessException;
                     System.ObjectDisposedException;
                     System.MemberAccessException;
                     System.MissingFieldException;
              }
       }
};
```

Calls

```
using System;
using System.Collections.Generic;

namespace Calls
{
       class Calls
       {
              //possible exceptions that can occur in the following methods. Refer to
their code to  see how they can occur.

              Calls()
        {
                     System.InvalidCastException;
                     System.ArithmeticException;
              }
              public void NewCall()
        {
                     System.IndexOutOfRangeException;
```

```
                    System.IO.IOException;
                    System.MethodAccessException;
                    System.AccessViolationException;
                    System.UnauthorizedAccessException;
                }
                public void ReceiveCall(string caller)
        {
                    System.ArgumentException;
                    System.ArgumentNullException;
                    System.ArgumentOutOfRangeException;
                    System.IndexOutOfRangeException;
                    System.IO.IOException;
                    System.AccessViolationException;
                    System.MethodAccessException;
                    System.UnauthorizedAccessException;
                }
                public void AddNewCallToLog()
        {
                    //No Exception Thrown
                }
                public void DeleteCallFromLog(string CallId)
        {
                    System.IndexOutOfRangeException;
                    System.ArgumentException;
                    System.ArgumentNullException;
                    System.ArgumentOutOfRangeException;
                }
                public void ClearLog()
        {
                    //No Exception Thrown
        }
                public void BlockCaller(string Callid)
        {
                    System.IndexOutOfRangeException;
                    System.ArgumentException;
                    System.ArgumentNullException;
                    System.ArgumentOutOfRangeException;
                }
       };
                                    }
```

Communication

```
using System;
using Email;
using Calls;
using BlogPosts;
using Meetings;

namespace communication
{
       public class Communication
       {
              //possible exceptions that can occur in the following methods. Refer to
their code to  see how they can occur.

              private void CommunicateViaEmail(string TouristId, string
ServiceProviderId, Email emailInstance)
        {
```

```
                    System.MethodAccessException;
                    System.AccessViolationException;
                    System.UnauthorizedAccessException;
                    System.MissingMethodException;
                    System.NotImplementedException;
                    System.ArgumentException;
                    System.ArgumentNullException;
                    System.ArgumentOutOfRangeException;
            }
            private void CommunicateViaMessages(string TouristId, string
ServiceProviderId, Message MessagesInstance)
        {
                    System.MethodAccessException;
                    System.AccessViolationException;
                    System.UnauthorizedAccessException;
                    System.MissingMethodException;
                    System.NotImplementedException;
                    System.ArgumentException;
                    System.ArgumentNullException;
                    System.ArgumentOutOfRangeException;
            }
            private void CommunicateViaCalls(string TouristId, string
ServiceProviderId, , Calls CallsInstance)
        {
                    System.ArgumentException;
                    System.ArgumentNullException;
                    System.ArgumentOutOfRangeException;
                    System.MethodAccessException;
                    System.AccessViolationException;
                    System.UnauthorizedAccessException;
                    System.MissingMethodException;
                    System.NotImplementedException;
            }
            private void CommunicateViaMeetings(string TouristId, string
ServiceProviderId, Meetings MeetingsInstance)
        {
                    System.ArgumentException;
                    System.ArgumentNullException;
                    System.ArgumentOutOfRangeException;
                    System.MethodAccessException;
                    System.AccessViolationException;
                    System.UnauthorizedAccessException;
                    System.MissingMethodException;
                    System.NotImplementedException;
            }
       };
}
```

Email

```
using System;
using System.Collections.Generic;
using EmailAPI;

namespace email
{
    class Email
    {
        //possible exceptions that can occur in the following methods. Refer to their
code to  see how they can occur.

        Email()
        {
            System.InvalidCastException;
            System.ArithmeticException;
        }
        public void NewEmail()
        {
            System.IO.IOException;
        }
        public void MakeDraft()
        {
            System.IO.IOException;
        }
        public void SendEmail(string Content, string Receipent)
        {
            System.ArgumentException;
            System.ArgumentNullException;
            System.ArgumentOutOfRangeException;
            System.MethodAccessException;
            System.AccessViolationException;
            System.UnauthorizedAccessException;
        }
        public void ReceiveEmail()
        {
            System.MethodAccessException;
            System.AccessViolationException;
            System.UnauthorizedAccessException;
        }
        public void DiscardDraft(string Emailid)
        {
            System.ArgumentException;
            System.ArgumentNullException;
            System.ArgumentOutOfRangeException;
            System.IndexOutOfRangeException;
        }
        public void DeleteEmail(string EmailId)
        {
            System.ArgumentException;
            System.ArgumentNullException;
            System.ArgumentOutOfRangeException;
            System.IndexOutOfRangeException;
        }
        public void ReportEmail(string EmailId)
        {
            System.ArgumentException;
            System.ArgumentNullException;
            System.ArgumentOutOfRangeException;
            System.IO.IOException;
            System.MethodAccessException;
            System.AccessViolationException;
            System.UnauthorizedAccessException;
        }
    };
}
```

Meetings

```
using System;
using System.Collections.Generic;
using MeetAPI;

namespace meetings
{
	class Meetings
	{
		//possible exceptions that can occur in the following methods. Refer to
their code to  see how they can occur.

		Meetings()
	 {
			System.InvalidCastException;
			System.ArithmeticException;
		}
		public void ScheduleNewMeeting()
	 {
			System.IO.IOException;
		}
		public void AttendMeet(string MeetId)
	 {
			System.ArgumentException;
			System.ArgumentNullException;
			System.ArgumentOutOfRangeException;
			System.IO.IOException;
			System.IndexOutOfRangeException;
		}
		public void cancelMeet(string MeetId)
	 {
			System.ArgumentException;
			System.ArgumentNullException;
			System.ArgumentOutOfRangeException;
			System.IndexOutOfRangeException;
		}
		public void SendInvitation(List<string> Participants, string MeetId)
	 {
			System.ArgumentException;
			System.ArgumentNullException;
			System.ArgumentOutOfRangeException;
			System.IndexOutOfRangeException;
			System.MethodAccessException;
			System.AccessViolationException;
			System.UnauthorizedAccessException;
		}
	};
}
```

Message

```
using System;
using System.Collections.Generic;
using MsgAPI;

namespace message
{
	class Message
```

```
        {
                //possible exceptions that can occur in the following methods. Refer to
    their code to  see how they can occur.

                Message()
         {
                        System.IO.IOException;
                        System.InvalidCastException;
                }
                public void NewMessage()
         {
                        System.IO.IOException;
                }
                public void MakeDraft()
         {
                        System.IO.IOException;
                }
                public void SendMessage(string Content, string Receipent)
         {
                        System.ArgumentException;
                        System.ArgumentNullException;
                        System.ArgumentOutOfRangeException;
                        System.MethodAccessException;
                        System.AccessViolationException;
                        System.UnauthorizedAccessException;
                }
                public void ReceiveMessage(string MsgId)
         {
                        System.ArgumentException;
                        System.ArgumentNullException;
                        System.ArgumentOutOfRangeException;
                        System.MethodAccessException;
                        System.AccessViolationException;
                        System.UnauthorizedAccessException;
                }
                public void DiscardDraft(string MsgId)
         {
                        System.ArgumentException;
                        System.ArgumentNullException;
                        System.ArgumentOutOfRangeException;
                        System.IndexOutOfRangeException;
                }
                public void DeleteMsg(string MsgId)
         {
                        System.ArgumentException;
                        System.ArgumentNullException;
                        System.ArgumentOutOfRangeException;
                        System.IndexOutOfRangeException;
                }
                public void ReportMsg(string MsgId)
         {
                        System.ArgumentException;
                        System.ArgumentNullException;
                        System.ArgumentOutOfRangeException;
                        System.IO.IOException;
                        System.MethodAccessException;
                        System.AccessViolationException;
                        System.UnauthorizedAccessException;
```

```
            }
        };
}
```

Pictures

```
using System;
using Admin;
using Validate;
using System.IO;

namespace pictures
{
        class Pictures
        {
                //possible exceptions that can occur in the following methods. Refer to
their code to  see how they can occur.


                Pictures()
         {
                        System.IO.IOException;
                        System.BadImageFormatException;
                        System.InvalidCastException;
                }
                public void RequestModification()
         {
                        System.IO.IOException;
                        System.FieldAccessException;
                        System.AccessViolationException;
                        System.MethodAccessException;
                        System.ObjectDisposedException;
                        System.UnauthorizedAccessException;
                }
                public void DeletePicture()
         {
                        System.AccessViolationException;
                        System.MethodAccessException;
                        System.ObjectDisposedException;
                }
                public void ApproveUpload()
         {
                        System.AccessViolationException;
                        System.MethodAccessException;
                        System.ObjectDisposedException;
                        System.MemberAccessException;
                        System.MissingFieldException;
                }
                public void RequestUpload()
         {
                        System.IO.IOException;
                        System.FieldAccessException;
                        System.AccessViolationException;
                        System.MethodAccessException;
                        System.ObjectDisposedException;
                        System.UnauthorizedAccessException;
                }
```

```
	};
}
```

Videos

```
using System;
using Admin;
using Validate;
using System.IO;

namespace Video
{
	class Video
	{

		//possible exceptions that can occur in the following methods. Refer to
their code to  see how they can occur.


		Video()
         {
			System.IO.IOException;
			System.BadImageFormatException;
			System.InvalidCastException;
		}
		public void DeleteVideo()
         {
			System.AccessViolationException;
			System.MethodAccessException;
			System.ObjectDisposedException;
		}
		public void ApproveUpload()
         {
			System.AccessViolationException;
			System.MethodAccessException;
			System.ObjectDisposedException;
			System.MemberAccessException;
			System.MissingFieldException;
		}
		public void RequestModification()
         {
			System.IO.IOException;
			System.FieldAccessException;
			System.AccessViolationException;
			System.MethodAccessException;
			System.ObjectDisposedException;
			System.UnauthorizedAccessException;
		}
		public void RequestUpload()
         {
			System.IO.IOException;
			System.FieldAccessException;
			System.AccessViolationException;
			System.MethodAccessException;
			System.ObjectDisposedException;
			System.UnauthorizedAccessException;
		}
	};
}
```