```
//Class Customer
using System;
using System.Collections.Generic;
using System.Text;

namespace PTSLibrary
{
    [Serializable]
    public class Customer : User
    {
        public Customer() { }
        public Customer(string name, int id)
        {
            this.name = name;
            this.id = id;
        }
    }
}
//AdminDAO class:
using System;
using System.Collections.Generic;
using System.Text;
using System.Data.SqlClient;
using System.Data;
namespace PTSLibrary.DAO
{
    class AdminDAO : SuperDAO
    {
        public int Authenticate(string username, string password)
        {
            string sql;
            SqlConnection cn;
            SqlCommand cmd;
            SqlDataReader dr;
            sql = String.Format("SELECT UserId FROM Person WHERE
IsAdministrator = 1 AND Username='{0}' AND Password='{1}'", username,
password);
            cn = new
SqlConnection(Properties.Settings.Default.ConnectionString);
            cmd = new SqlCommand(sql, cn);
            int id = 0;
            try
            {
                cn.Open();
                dr = cmd.ExecuteReader(CommandBehavior.SingleRow);
                if (dr.Read())
                {
                    id = (int)dr["UserId"];
                }
                dr.Close();
            }
            catch (SqlException ex)
            {
                throw new Exception("Error Accessing Database", ex);
            }
            finally
            {
```

```
                cn.Close();
            }
            return id;
        }
        public void CreateProject(string name, DateTime startDate, DateTime
endDate, int
        customerId, int administratorId)
        {
            string sql;
            SqlConnection cn;
            SqlCommand cmd;
            Guid projectId = Guid.NewGuid();
            sql = "INSERT INTO Project (ProjectId, Name, ExpectedStartDate,
ExpectedEndDate,CustomerId, AdministratorId)";
            sql += String.Format("VALUES ( '{0}', '{1}', '{2}', '{3}', {4},
{5})", projectId, name, startDate,
            endDate, customerId, administratorId);
            cn = new
SqlConnection(Properties.Settings.Default.ConnectionString);
            cmd = new SqlCommand(sql, cn);
            try
            {
                cn.Open();
                cmd.ExecuteNonQuery();
            }
            catch (SqlException ex)
            {
                throw new Exception("Error Inserting", ex);
            }
            finally
            {
                cn.Close();
            }
        }
        public List<Customer> GetListOfCustomers()
        {
            string sql;
            SqlConnection cn;
            SqlCommand cmd;
            SqlDataReader dr;
            List<Customer> customers;
            customers = new List<Customer>();
            sql = "SELECT * FROM Customer";
            cn = new
SqlConnection(Properties.Settings.Default.ConnectionString);
            cmd = new SqlCommand(sql, cn);
            try
            {
                cn.Open();
                dr = cmd.ExecuteReader();
                while (dr.Read())
                {
                    Customer c = new Customer(dr["Name"].ToString(),
(int)dr["CustomerId"]);
                    customers.Add(c);
                }
                dr.Close();
```

```
            }
            catch (SqlException ex)
            {
                throw new Exception("Error Getting list", ex);
            }
            finally
            {
                cn.Close();
            }
            return customers;
        }
        public List<Project> GetListOfProjects(int adminId)
        {
            string sql;
            SqlConnection cn;
            SqlCommand cmd;
            SqlDataReader dr;
            List<Project> projects;
            projects = new List<Project>();
            sql = "SELECT * FROM Project WHERE AdministratorId = " + adminId;
            cn = new
SqlConnection(Properties.Settings.Default.ConnectionString);
            cmd = new SqlCommand(sql, cn);
            try
            {
                cn.Open();
                dr = cmd.ExecuteReader();
                while (dr.Read())
                {
                    Customer cust = GetCustomer((int)dr["CustomerId"]);
                    Project p = new Project(dr["Name"].ToString(),
(DateTime)dr["ExpectedStartDate"],
                    (DateTime)dr["ExpectedEndDate"], (Guid)dr["ProjectId"],
cust);
                    projects.Add(p);
                }
                dr.Close();
            }
            catch (SqlException ex)
            {
                throw new Exception("Error Getting list", ex);
            }
            finally
            {
                cn.Close();
            }
            return projects;
        }
        public List<Team> GetListOfTeams()
        {
            string sql;
            SqlConnection cn;
            SqlCommand cmd;
            SqlDataReader dr;
            List<Team> teams;
            teams = new List<Team>();
            sql = "SELECT * FROM Team";
```

```
            cn = new
SqlConnection(Properties.Settings.Default.ConnectionString);
            cmd = new SqlCommand(sql, cn);
            try
            {
                cn.Open();
                dr = cmd.ExecuteReader();
                while (dr.Read())
                {
                    Team t = new Team((int)dr["TeamId"],
dr["Location"].ToString(),
                    dr["Name"].ToString(), null);
                    teams.Add(t);
                }
                dr.Close();
            }
            catch (SqlException ex)
            {
                throw new Exception("Error getting team list", ex);
            }
            finally
            {
                cn.Close();
            }
            return teams;
        }
        public void CreateTask(string name, DateTime startDate, DateTime
endDate, int teamId,
        Guid projectId)
        {
            string sql;
            SqlConnection cn;
            SqlCommand cmd;
            Guid taskId = Guid.NewGuid();
            sql = "INSERT INTO Task (TaskId, Name, ExpectedDateStarted,
ExpectedDateCompleted,ProjectId, TeamId, StatusId)";
            sql += String.Format("VALUES ( '{0}', '{1}', '{2}', '{3}', '{4}',
{5}, {6})", taskId, name,
            startDate, endDate, projectId, teamId, 1);
            cn = new
SqlConnection(Properties.Settings.Default.ConnectionString);
            cmd = new SqlCommand(sql, cn);
            try
            {
                cn.Open();
                cmd.ExecuteNonQuery();
            }
            catch (SqlException ex)
            {
                throw new Exception("Error Inserting", ex);
            }
            finally
            {
                cn.Close();
            }
        }
    }
```

```
}

// class ClientDAO
 using System;
using System.Collections.Generic;
using System.Text;
using System.Data.SqlClient;
using System.Data;
namespace PTSLibrary.DAO
{
class ClientDAO : SuperDAO
{
public TeamLeader Authenticate(string username, string password)
{
string sql;
SqlConnection cn;
SqlCommand cmd;
SqlDataReader dr;
TeamLeader leader = null;
sql = String.Format("SELECT DISTINCT Person.Name, UserId, TeamId FROM Person
INNER JOIN Team ON (Team.TeamLeaderId = Person.UserId) WHERE Username='{0}'
AND Password='{1}'", username, password);
cn = new SqlConnection(Properties.Settings.Default.ConnectionString);
cmd = new SqlCommand(sql, cn);
try
{
cn.Open();
dr = cmd.ExecuteReader(CommandBehavior.SingleRow);
if (dr.Read())
{
leader = new TeamLeader(dr["Name"].ToString(), (int)dr["TeamId"],
(int)dr["TeamId"]);
}
dr.Close();
}
catch (SqlException ex)
{
throw new Exception("Error Accessing Database", ex);
}
finally
{
cn.Close();
}
return leader;
}
public List<Project> GetListOfProjects(int teamId)
{
string sql;
SqlConnection cn;
SqlCommand cmd;
SqlDataReader dr;
List<Project> projects;
projects = new List<Project>();
sql = "SELECT P.* FROM Project AS P INNER JOIN Task AS T ON (P.ProjectId =
T.ProjectId)WHERE T.TeamId = " + teamId;
cn = new SqlConnection(Properties.Settings.Default.ConnectionString);
cmd = new SqlCommand(sql, cn);
```

```
try
{
cn.Open();
dr = cmd.ExecuteReader();
while (dr.Read())
{
Customer cust = GetCustomer((int)dr["CustomerId"]);
Project p = new Project(dr["Name"].ToString(),
(DateTime)dr["ExpectedStartDate"],
(DateTime)dr["ExpectedEndDate"], (Guid)dr["ProjectId"], cust);
projects.Add(p);
}
dr.Close();
}
catch (SqlException ex)
{
throw new Exception("Error Getting list", ex);
}
finally
{
cn.Close();
}
return projects;
}
}
}
//Complete code listing for the CustomerDAO class:
using System;
using System.Collections.Generic;
using System.Text;
using System.Data.SqlClient;
using System.Data;
namespace PTSLibrary.DAO
{
class CustomerDAO : SuperDAO
{
public int Authenticate(string username, string password)
{
string sql;
SqlConnection cn;
SqlCommand cmd;
SqlDataReader dr;
sql = String.Format("SELECT CustomerId FROM Customer WHERE Username='{0}' AND
Password='{1}'", username, password);
cn = new SqlConnection(Properties.Settings.Default.ConnectionString);
cmd = new SqlCommand(sql, cn);
int id = 0;
try
{
cn.Open();
dr = cmd.ExecuteReader(CommandBehavior.SingleRow);
if (dr.Read())
{
id = (int)dr["CustomerId"];
}
dr.Close();
}
```

```
catch (SqlException ex)
{
throw new Exception("Error Accessing Database", ex);
}
finally
{
cn.Close();
}
return id;
}
public List<Project> GetListOfProjects(int customerId)
{
string sql;
SqlConnection cn, cn2;
SqlCommand cmd, cmd2;
SqlDataReader dr, dr2;
List<Project> projects;
projects = new List<Project>();
sql = "SELECT * FROM Project WHERE CustomerId = " + customerId;
cn = new SqlConnection(Properties.Settings.Default.ConnectionString);
cmd = new SqlCommand(sql, cn);
try
{
cn.Open();
dr = cmd.ExecuteReader();
while (dr.Read())
{
List<Task> tasks = new List<Task>();
sql = "SELECT * FROM Task WHERE ProjectId = '" + dr["ProjectId"].ToString() +
"'";
cn2 = new SqlConnection(Properties.Settings.Default.ConnectionString);
cmd2 = new SqlCommand(sql, cn2);
cn2.Open();
dr2 = cmd2.ExecuteReader();
while (dr2.Read())
{
Task t = new Task((Guid)dr2["TaskId"], dr2["Name"].ToString(),
(Status)dr2["StatusId"]);
tasks.Add(t);
}
dr2.Close();
Project p = new Project(dr["Name"].ToString(),
(DateTime)dr["ExpectedStartDate"],
(DateTime)dr["ExpectedEndDate"], (Guid)dr["ProjectId"], tasks);
projects.Add(p);
}
dr.Close();
}
catch (SqlException ex)
{
throw new Exception("Error Getting list", ex);
}
finally
{
cn.Close();
}
return projects;
```

```
}
}
}
//Complete code listing for the class SuperDAO:
using System;
using System.Collections.Generic;
using System.Text;
using System.Data.SqlClient;
using System.Data;
namespace PTSLibrary.DAO
{
public class SuperDAO
{
protected Customer GetCustomer(int custId)
{
string sql;
SqlConnection cn;
SqlCommand cmd;
SqlDataReader dr;
Customer cust;
sql = "SELECT * FROM Customer WHERE CustomerId = " + custId;
//cn = new SqlConnection(Properties.Settings.Default.ConnectionString);
cn = new SqlConnection(Properties.Settings.Default.ConnectionString);
cmd = new SqlCommand(sql, cn);
try
{
cn.Open();
dr = cmd.ExecuteReader(CommandBehavior.SingleRow);
dr.Read();
cust = new Customer(dr["Name"].ToString(), (int)dr["CustomerId"]);
dr.Close();
}
catch (SqlException ex)
{
throw new Exception("Error Getting Customer", ex);
}
finally
{
cn.Close();
}
return cust;
}
public List<Task> GetListOfTasks(Guid projectId)
{
string sql;
SqlConnection cn;
SqlCommand cmd;
SqlDataReader dr;
List<Task> tasks;
tasks = new List<Task>();
sql = "SELECT * FROM Task WHERE ProjectId = '" + projectId + "'";
cn = new SqlConnection(Properties.Settings.Default.ConnectionString);
cmd = new SqlCommand(sql, cn);
try
{
cn.Open();
dr = cmd.ExecuteReader();
```

```
while (dr.Read())
{
Task t = new Task((Guid)dr["TaskId"], dr["Name"].ToString(),
(Status)((int)dr["StatusId"]));
tasks.Add(t);
}
dr.Close();
}
catch (SqlException ex)
{
throw new Exception("Error getting taks list", ex);
}
finally
{
cn.Close();
}
return tasks;
}
}
}
```

//PTSClientWebService.cs

```
using System;
using System.Web;
using System.Collections;
using System.Web.Services;
using System.Web.Services.Protocols;
using PTSLibrary;


/// <summary>
/// Summary description for PTSClientWebService
/// </summary>
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class PTSClientWebService : System.Web.Services.WebService
{
    private PTSClientFacade facade;
    public PTSClientWebService()
    {

        //Uncomment the following line if using designed components
        //InitializeComponent();
        facade=new PTSClientFacade();
    }

    [WebMethod]
    public string HelloWorld()
    {
        return "Hello World";
    }
    [WebMethod]
    public TeamLeader Authenticate(string username, string password)
    {
```

```
    return facade.Authenticate(username, password);
    }
    [WebMethod]

    public Project[] GetListOfProjects(int teamId)
    {
    return facade.GetListOfProjects(teamId);
    }

}
```

//PTSCustomerWebService.cs

```
using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;
using PTSLibrary;

[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class Service : System.Web.Services.WebService
{
    private PTSCustomerFacade facade;

    public Service () {

        //Uncomment the following line if using designed components
        //InitializeComponent();
        facade = new PTSCustomerFacade();
    }

    [WebMethod]
    public string HelloWorld() {
        return "Hello World";
    }
    [WebMethod]
    public int Authenticate(string username, string password)
    {
        return facade.Authenticate(username, password);
    }
    [WebMethod]

    public Project[] GetListOfProjects(int customerId)
    {
        return facade.GetListOfProjects(customerId);
    }

}
```

```
//The complete code for the class Project:
using System;
using System.Collections.Generic;
using System.Text;
namespace PTSLibrary
{
    [Serializable]
    public class Project
    {
        private string name;
        private DateTime expectedStartDate;
        private DateTime expectedEndDate;
        private Customer theCustomer;
        private Guid projectId;
        private List<Task> tasks;
        public List<Task> Tasks
        {
            get { return tasks; }
            set { tasks = value; }
        }
        public Project() { }
        public Customer TheCustomer
        {
            get { return theCustomer; }
            set { theCustomer = value; }
        }
        public Guid ProjectId
        {
            get { return projectId; }
        }
        public string Name
        {
            get { return name; }
            set { name = value; }
        }
        public DateTime ExpectedStartDate
        {
            get { return expectedStartDate; }
            set { expectedStartDate = value; }
        }
        public DateTime ExpectedEndDate
        {
            get { return expectedEndDate; }
            set { expectedEndDate = value; }
        }
        public Project(string name, DateTime startDate, DateTime endDate,
Guid projectId,
        Customer customer)
        {
            this.name = name;
            this.expectedStartDate = startDate;
            this.expectedEndDate = endDate;
            this.projectId = projectId;
            this.theCustomer = customer;
        }
        public Project(string name, DateTime startDate, DateTime endDate,
Guid projectId,
```

```
        List<Task> tasks)
        {
            this.name = name;
            this.expectedStartDate = startDate;
            this.expectedEndDate = endDate;
            this.projectId = projectId;
            this.tasks = tasks;
        }
    }
}
//PTS AdminFacade
using System;
using System.Collections.Generic;
using System.Text;
namespace PTSLibrary
{
    public class PTSAdminFacade : PTSSuperFacade
    {
        private DAO.AdminDAO dao;
        public PTSAdminFacade()
            : base(new DAO.AdminDAO())
        {
            dao = (DAO.AdminDAO)base.dao;
        }
        public int Authenticate(string username, string password)
        {
            if (username == "" || password == "")
            {
                throw new Exception("Missing Data");
            }
            return dao.Authenticate(username, password);
        }
        public void CreateProject(string name, DateTime startDate, DateTime
endDate, int
        customerId, int administratorId)
        {
            if (name == null || name == "" || startDate == null || endDate ==
null)
            {
                throw new Exception("Missing Data");
            }
            dao.CreateProject(name, startDate, endDate, customerId,
administratorId);
        }
        public Customer[] GetListOfCustomers()
        {
            return (dao.GetListOfCustomers()).ToArray();
        }
        public Project[] GetListOfProjects(int adminId)
        {
            return (dao.GetListOfProjects(adminId)).ToArray();
        }
        public Team[] GetListOfTeams()
        {
            return (dao.GetListOfTeams()).ToArray();
        }
```

```
        public void CreateTask(string name, DateTime startDate, DateTime
endDate, int teamId,
        Guid projectId)
        {
            if (name == null || name == "" || startDate == null || endDate ==
null)
            {
                throw new Exception("Missing Data");
            }
            dao.CreateTask(name, startDate, endDate, teamId, projectId);
        }
    }
}



//PTSClientFacade Class
using System;
using System.Collections.Generic;
using System.Text;

namespace PTSLibrary
{
    public class PTSClientFacade : PTSSuperFacade
    {
        private DAO.ClientDAO dao;
        public PTSClientFacade() : base(new DAO.ClientDAO())
        {

            dao = (DAO.ClientDAO)base.dao;

        }


        public TeamLeader Authenticate(string username, string password)
        {

            if (username == "" || password == "")
            {
                throw new Exception("Missing Data");
            }
            return dao.Authenticate(username, password);
        }
        public Project[] GetListOfProjects(int teamId)
        {
            return (dao.GetListOfProjects(teamId)).ToArray();
        }
    }
}

//PTSCustomerFacade Class

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

```
namespace PTSLibrary
{
    public class PTSCustomerFacade : PTSSuperFacade
    {
        private DAO.CustomerDAO dao;

        public PTSCustomerFacade() : base(new DAO.CustomerDAO())
        {
            dao = (DAO.CustomerDAO)base.dao;
        }

        public Project[] GetListOfProjects(int customerId)
        {
                return(dao.GetListOfProjects(customerId)).ToArray();
        }
        public int Authenticate(string username, string password)
        {
            return (dao.Authenticate(username, password));
        }
    }
}

//PTSSuperFacade
using System;
using System.Collections.Generic;
using System.Text;

namespace PTSLibrary
{
    public class PTSSuperFacade: MarshalByRefObject
    {
        protected DAO.SuperDAO dao;
        public PTSSuperFacade(DAO.SuperDAO dao)
        {
            this.dao = dao;
        }
        public Task[] GetListOfTasks(Guid projectId)
        {
            //return (dao.GetListOfTasks(projectId)).ToArray();

            return(dao.GetListOfTasks(projectId)).ToArray();
        }
    }
}

using System;
using System.Collections.Generic;
using System.Text;

namespace PTSLibrary
{



    public enum Status
    {
        ReadyToStart = 1,
```

```
        InProgress = 2,
        Completed = 3,
        WaitingForPredecesser = 4,

    }
}


//Subtask class
using System;
using System.Collections.Generic;
using System.Text;

namespace PTSLibrary
{
    class Subtask
    {
    }
}

//Class Task
using System;
using System.Collections.Generic;
using System.Text;

namespace PTSLibrary
{
    [Serializable]
    public class Task
    {
        private Guid taskId;
        private string name;
        private Status status;

        public Task() { }

        public Guid TaskId
        {
            get { return taskId; }
            set { taskId = value; }
        }
        public string Name
        {
            get { return name; }
            set { name = value; }
        }

        public Status theStatus
        {
            get { return status; }
            set { status = value; }

        }

        public Task(Guid id, string name, Status status)
        {
            this.taskId = id;
```

```
            this.name = name;
            this.status = status;
        }
        public string NameAndStatus
        {
            get { return name + "=" + status; }
        }
    }
}

//Class Team
using System;
using System.Collections.Generic;
using System.Text;

namespace PTSLibrary
{
    [Serializable]
    public class Team
    {
        private int id;
        private string location;
        private string name;
        private TeamLeader leader;

        public Team() { }

        public int TeamId
        {
            get { return id; }
            set { id = value; }
        }

        public TeamLeader Leader
        {
            get { return leader; }
            set { leader = value; }
        }

        public string Location
        {
            get { return location; }
            set { location = value; }
        }
        public string Name
        {
            get { return name; }
            set { name = value; }
        }

        public Team(int id, string location, string name, TeamLeader leader)
        {
            this.location = location;
            this.name = name;
            this.id = id;
            this.Leader = leader;
        }
```

```
    }
}

//Class TeamLeader
using System;
using System.Collections.Generic;
using System.Text;

namespace PTSLibrary
{
    [Serializable]
    public class TeamLeader : User
    {
        public TeamLeader() { }
        private int teamId;
        public int TeamId
        {
            get { return teamId; }
            set { teamId = value; }

        }
        public TeamLeader(string name, int id, int teamId)
        {
            this.name = name;
            this.id = id;
            this.teamId = teamId;
        }
    }
}
//Class TeamMember
using System;
using System.Collections.Generic;
using System.Text;

namespace PTSLibrary
{
    class TeamMember
    {
    }
}

//Class User
using System;
using System.Collections.Generic;
using System.Text;

namespace PTSLibrary
{
    [Serializable]
    public class User
    {
        protected string name;
        protected int id;
        public string Name
        {
            get { return name; }
        }
```

```
        public int Id
        {
            get { return id; }
        }
    }
}
```