```
#drop database if exists propertydb;
#create database propertyDB;
use propertyDB;

DROP TABLE IF EXISTS property;
#Create property table. Add property name.
CREATE TABLE `property` (
        `property_no` int(11) auto_increment,
    `property_name` varchar(255) default null,
    `property_address` text default null,
    `post_code` varchar(255) default null,
    `property_type` varchar(255) default null,
    `property_price` varchar(255) default null,
    `max_no_occupancy` int default null,
    `multiple_occupancy` tinyint(1) not null,
    `resident_status` varchar(255) not null,
    `pets_allowed` varchar(25) not null,
    `children_allowed` varchar(25) not null,
    PRIMARY KEY (`property_no`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

#insert values
INSERT INTO property (property_name, property_address, post_code, property_type,
property_price, resident_status, max_no_occupancy, multiple_occupancy, pets_allowed,
children_allowed)
VALUES ("Block A, Flat14", "Nixon Court, 33 Putney Road, Leicester", "LE2 7TG", "Student
Flat", "Â£430 pcm", "Student", 6, 1, "no", "no");
INSERT INTO property (property_name, property_address, post_code, property_type,
property_price, resident_status, max_no_occupancy, multiple_occupancy, pets_allowed,
children_allowed)
VALUES ("Block B, Flat12", "Nixon Court, 33 Putney Road, Leicester", "LE2 7TG", "Student
Flat", "Â£430 pcm", "Student", 6, 1, "no", "no");
INSERT INTO property (property_name, property_address, post_code, property_type,
property_price, resident_status, max_no_occupancy, multiple_occupancy, pets_allowed,
children_allowed)
VALUES ("Flat 11, Regent Place", "44 Church Road, Luton", "LU4 3JP", "Flat", "Â£500 pcm",
"Non-Student", 4, 1, "yes", "yes");
INSERT INTO property (property_name, property_address, post_code, property_type,
property_price, resident_status, max_no_occupancy, multiple_occupancy, pets_allowed,
children_allowed)
VALUES ("Chris Wilkinsons House", "23 Gramby Street, Moss Side, Manchester", "M44 3JG",
"House", "Â£650 pcm", "Non-Student", 4, 1, "yes", "yes");
INSERT INTO property (property_name, property_address, post_code, property_type,
property_price, resident_status, max_no_occupancy, multiple_occupancy, pets_allowed,
children_allowed)
VALUES ("2 Bedroom Flat", "Steel Grove, 33 Jubely Road, Leicester", "LE1 5PB",
"Appartment", "Â£751 pcm", "Non-Student", 2, 1, "yes", "yes");
INSERT INTO property (property_name, property_address, post_code, property_type,
property_price, resident_status, max_no_occupancy, multiple_occupancy, pets_allowed,
children_allowed)
VALUES ("Bourmount Gardends", "17 Narbourough Road, Leicester", "LE3 5JQ", "Appartment",
"Â£850 pcm", "Non-Student", 1, 0, "no", "no");
INSERT INTO property (property_name, property_address, post_code, property_type,
property_price, resident_status, max_no_occupancy, multiple_occupancy, pets_allowed,
children_allowed)
VALUES ("Block E, Flat506", "Nixon Court, 33 Putney Road, Leicester", "LE2 7TG", "Student
Flat", "Â£430 pcm", "Student", 6, 1, "no", "no");
INSERT INTO property (property_name, property_address, post_code, property_type,
property_price, resident_status, max_no_occupancy, multiple_occupancy, pets_allowed,
children_allowed)
VALUES ("Slough Gardens", "Nixon Court, 30 Putney Road, Leicester", "LE2 7TP", "House",
"Â£330 pcm", "Student", 4, 1, "no", "no");
```

T1 6/6
T3 10/10
T4. 6/6
22/22

```
/*
INSERT INTO property (property_name, property_address, post_code, property_type,
property_price, resident_status, max_no_occupancy, multiple_occupancy, pets_allowed,
children_allowed)
VALUES ("Cherry Gardens", "Nixon Court, 30 Putney Road, Leicester", "LE2 7TP", "House",
"Â£330 pcm", "Student", 4, 1, "no", "no");
*/
/*

INSERT INTO property (property_name, property_address, post_code, property_type,
property_price, resident_status, max_no_occupancy, multiple_occupancy, pets_allowed,
children_allowed)
VALUES ("Block A, Flat14", "Nixon Court, 33 Putney Road, Leicester", "LE2 7TG", "Student
Flat", "Â£430 pcm", "Student", 6, 1, "no", "no");
INSERT INTO property (property_name, property_address, post_code, property_type,
property_price, resident_status, max_no_occupancy, multiple_occupancy, pets_allowed,
children_allowed)
VALUES ("Block B, Flat12", "Nixon Court, 33 Putney Road, Leicester", "LE2 7TG", "Student
Flat", "Â£430 pcm", "Student", 6, 1, "no", "no");
INSERT INTO property (property_name, property_address, post_code, property_type,
property_price, resident_status, max_no_occupancy, multiple_occupancy, pets_allowed,
children_allowed)
VALUES ("Block A, Flat14", "Nixon Court, 33 Putney Road, Leicester", "LE2 7TG", "Student
Flat", "Â£430 pcm", "Student", 6, 1, "no", "no");
INSERT INTO property (property_name, property_address, post_code, property_type,
property_price, resident_status, max_no_occupancy, multiple_occupancy, pets_allowed,
children_allowed)
VALUES ("Block B, Flat12", "Nixon Court, 33 Putney Road, Leicester", "LE2 7TG", "Student
Flat", "Â£430 pcm", "Student", 6, 1, "no", "no");
*/

DROP TABLE IF EXISTS `lettings`;
#Create the lettings table
CREATE TABLE `lettings` (
        `rental_no` int(11) auto_increment,
    `current_status` tinyint(1) default NULL, #0 for empty, 1 for let
    `lease_arrangement` date default NULL,
    `is_available` tinyint(1) default NULL,
    `property_no` int(11) NOT NULL,
    PRIMARY KEY (`rental_no`),
    FOREIGN KEY (`property_no`) REFERENCES property(`property_no`)

    #reference provided to primary key.
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

#Insert Data
INSERT INTO lettings (current_status, lease_arrangement, is_available, property_no)
VALUES (1, '2019-07-14', 0, 1);
INSERT INTO lettings (current_status, lease_arrangement, is_available, property_no)
VALUES (1, '2019-07-14', 0, 2);
INSERT INTO lettings (current_status,lease_arrangement, is_available, property_no)
VALUES (1, '2019-07-14', 0, 3);

INSERT INTO lettings (current_status,lease_arrangement, is_available, property_no)
VALUES (0, NULL, 1, 4);
INSERT INTO lettings (current_status,lease_arrangement, is_available, property_no)
VALUES (0, NULL, 1, 5);
INSERT INTO lettings (current_status, lease_arrangement, is_available, property_no)
VALUES (0, NULL, 1, 6);
INSERT INTO lettings (current_status, lease_arrangement, is_available, property_no)
VALUES (0, NULL, 1, 7);
INSERT INTO lettings (current_status,lease_arrangement, is_available, property_no)
VALUES (0, NULL, 1, 8);
```

```
#Display values stored in each entity
select * from property;
select * from lettings;

/*
select property.property_name, property.property_address, property.property_price,
property.max_no_occupancy,
property.multiple_occupancy,property.resident_status, property.pets_allowed,
lettings.current_status, lettings.is_available,
property.children_allowed from property inner join lettings on lettings.property_no =
property.property_no where property.resident_status = 'Student' and lettings.is_available =
1;
*/

#select * from property inner join lettings on lettings.property_no = property.property_no
where property.resident_status = 'Student' and lettings.is_available = 1;
```

```
ï»¿using System;
using System.Collections.Generic;
using MySql.Data.MySqlClient;

namespace PropertyService
{

    // NOTE: In order to launch WCF Test Client for testing this service, please select
PropertyService.svc or PropertyService.svc.cs at the Solution Explorer and start debugging.
    public class PropertyService : IPropertyService
    {


        //Search property via postcode
        public List<Property> SearchPropertyByPostCode(string postCode)
        {
            List<Property> propertyList = new List<Property>();

            try
            {
                MySqlConnection connection = GetConnection();
                connection.Open();

                //This variable is responsible for running the sql query.
                MySqlCommand sqlCommand = new MySqlCommand("", connection);
                sqlCommand.Prepare();

                //sql query for getting the properties based on the postcode.
                sqlCommand.CommandText = String.Format("select * from property where
property.post_code like '" + postCode + '%' + "' ");

                MySqlDataReader mySqlData = sqlCommand.ExecuteReader();

                //read the database and retrieve the data
                //data is set apporpriately and accordingly
                while (mySqlData.Read())
                {
                    if (mySqlData.HasRows)
                    {
                        Property property = new Property()
                        {
                            PropertyName = mySqlData.GetString("property_name"),
                            Address = mySqlData.GetString("property_address"),
                            PostCode = mySqlData.GetString("post_code"),
                            PropertyType = mySqlData.GetString("property_type"),
                            PropertyPrice = mySqlData.GetString("property_price"),
                            MaxNoOfOccupancy = mySqlData.GetInt32("max_no_occupancy"),
                            MultipleOccupancy = mySqlData.GetBoolean("multiple_occupancy"),
                            ResidentStatus = mySqlData.GetString("resident_status"),
                            Pets_allowed = mySqlData.GetString("pets_allowed"),
                            Children_allowed = mySqlData.GetString("children_allowed")
                        };

                        propertyList.Add(property);
                    }
                }

                //close the data reader and connection
                mySqlData.Close();
                connection.Close();
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }

            return propertyList;
        }

        //Method for inserting a new property into the database.
        public bool EnterNewProperty(Property p)
        {
            bool isAdded = false;
            try
            {
                MySqlConnection connection = GetConnection();
                connection.Open();
                MySqlCommand command = new MySqlCommand("", connection);
                command.Prepare();
                command.CommandText = String.Format("INSERT INTO property (property_name,
property_address, " +
                    "post_code, property_type, property_price, max_no_occupancy,
multiple_occupancy, resident_status, pets_allowed, children_allowed)"
                    + "values
(?name_param,?address_param,?post_code_param,?property_type_param,?property_price_param," +

"?max_no_occupancy_param,?multiple_occupancy_param,?resident_status_param,?pets_allowed_para
m,?children_param)");

                command.Parameters.AddWithValue("?name_param", p.PropertyName);
                command.Parameters.AddWithValue("?address_param", p.Address);
                command.Parameters.AddWithValue("?post_code_param", p.PostCode);
                command.Parameters.AddWithValue("?property_type_param", p.PropertyType);
                command.Parameters.AddWithValue("?property_price_param", p.PropertyPrice);
                command.Parameters.AddWithValue("?max_no_occupancy_param",
p.MaxNoOfOccupancy);
                command.Parameters.AddWithValue("?multiple_occupancy_param",
p.MultipleOccupancy);
                command.Parameters.AddWithValue("?resident_status_param", p.ResidentStatus);
                command.Parameters.AddWithValue("?pets_allowed_param", p.Pets_allowed);
                command.Parameters.AddWithValue("?children_param", p.Children_allowed);

                //Execute insertion query to add the following records to the property table
                command.ExecuteNonQuery();

                //Execute sql command for lettings table.

                int pID = Convert.ToInt32(command.LastInsertedId);
                //command.CommandText = String.Format("INSERT INTO lettings values
(?cs_param, ?lease_arrange_param, ?is_available_param, ?prop_no_param)");
                string sqlQuery = "INSERT INTO lettings (current_status, lease_arrangement,
is_available, property_no) values (?cs_param, ?lease_arrange_param, ?is_available_param,
?prop_no_param)";
                command = new MySqlCommand(sqlQuery, connection);
                command.Parameters.AddWithValue("?cs_param", p.CurrentStatus);
                command.Parameters.AddWithValue("?lease_arrange_param", p.LeaseArrDate);
                command.Parameters.AddWithValue("?is_available_param", p.IsAvailable);
                command.Parameters.AddWithValue("?prop_no_param", pID);


                //Execute insertion query to add the following records to the lettings
table.
                command.ExecuteNonQuery();

                //once the new property is added successfully boolean variable is set to
true.
```

```
                isAdded = true;
                //close connection
                connection.Close();
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
                return isAdded;
            }

            return isAdded;
        }

        //This method gets a list of all available properties that can acommodate students.
        public List<Property> GetAllPropertiesForStudents()
        {
            List<Property> availableProperties = new List<Property>();

            try
            {
                MySqlConnection connection = GetConnection();
                //open connection
                connection.Open();


                //mysql command for the property table
                MySqlCommand myscom = new MySqlCommand("", connection);

                myscom.Prepare();
                myscom.CommandText = String.Format("select * from property where
resident_status = 'Student'");


                MySqlDataReader reader = myscom.ExecuteReader();

                while (reader.Read())
                {
                    if (reader.HasRows)
                    {
                        Property p = new Property
                        {
                            PropertyID = reader.GetInt32("property_no"),
                            PropertyName = reader.GetString("property_name"),
                            Address = reader.GetString("property_address"),
                            PropertyPrice = reader.GetString("property_price"),
                            PostCode = reader.GetString("post_code"),
                            MaxNoOfOccupancy = reader.GetInt32("max_no_occupancy"),
                            MultipleOccupancy = reader.GetBoolean("multiple_occupancy"),
                            ResidentStatus = reader.GetString("resident_status"),
                            Pets_allowed = reader.GetString("pets_allowed"),
                            Children_allowed = reader.GetString("children_allowed"),
                            PropertyType = reader.GetString("property_type")
                            //CurrentStatus = reader.GetBoolean("current_status"),
                            //LeaseArrDate = reader.GetString("lease_arrangement"),
                            //IsAvailable = reader.GetBoolean("is_available")
                        };

                        availableProperties.Add(p);
                    }
                }

                //mysql command for the lettings table
                MySqlCommand command = new MySqlCommand("", connection);
                command.Prepare();
                command.CommandText = String.Format("select * from lettings where
is_available = 1");
                reader = command.ExecuteReader();

                while (reader.Read())
                {
                    if (reader.HasRows)
                    {
                        Property p = new Property();
                        p.CurrentStatus = reader.GetBoolean("current_status");
                        p.LeaseArrDate = reader.GetString("lease_arrangement");
                        p.IsAvailable = reader.GetBoolean("is_available");

                        availableProperties.Add(p);
                    }
                }


                reader.Close();
                connection.Close();


            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message + "\n" + ex.HelpLink);
            }

            return availableProperties;
        }


        //Method for establishing connection to the database.
        public static MySqlConnection GetConnection()
        {
            //NOTE - that I am connecting to mysql database locally!
            string connectionString = "server=localhost;user
id=root;database=propertyDB;port=3307;password=superman;CharSet=utf8;";

            //Instance for establishing connection
            MySqlConnection myconnection = new MySqlConnection(connectionString);

            return myconnection;
        }


    }
}
```

```
ï»¿<%@ ServiceHost Language="C#" Debug="true" Service="PropertyService.PropertyService"
CodeBehind="PropertyService.svc.cs" %>
```

```
ï»¿
using System.Collections.Generic;
using System.Runtime.Serialization;
using System.ServiceModel;


namespace PropertyService
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to change the
interface name "IPropertyService" in both code and config file together.
    [ServiceContract]
    public interface IPropertyService
    {

        [OperationContract]
        List<Property> SearchPropertyByPostCode(string postCode);

        [OperationContract]
        bool EnterNewProperty(Property p);

        [OperationContract]
        List<Property> GetAllPropertiesForStudents();
    }

    [DataContract]
    public class Property
    {

        //Encapsulate Fields
        [DataMember]
        public int PropertyID
        {
            get;
            set;
        }

        [DataMember]
        public string PropertyName
        {
            get;
            set;
        }

        [DataMember]
        public string Address
        {
            get;
            set;
        }

        [DataMember]
        public string PostCode
        {
            get;
            set;
        }

        [DataMember]
        public string PropertyType
        {
            get;
            set;
        }

        [DataMember]
        public string PropertyPrice
        {
            get;
            set;
        }

        [DataMember]
        public string ResidentStatus
        {
            get;
            set;
        }

        [DataMember]
        public int MaxNoOfOccupancy
        {
            get;
            set;
        }

        [DataMember]
        public bool MultipleOccupancy
        {
            get;
            set;
        }

        [DataMember]
        public string Pets_allowed
        {
            get;
            set;
        }

        [DataMember]
        public string Children_allowed
        {
            get;
            set;
        }

        [DataMember]
        public bool CurrentStatus
        {
            get;
            set;
        }

        [DataMember]
        public string LeaseArrDate
        {
            get;
            set;
        }

    }

        [DataMember]
        public bool IsAvailable

        {
            get;
```

```
            set;
        }

        //This method will be responsible for printing the values.

        override
        public string ToString()
        {
            return "Property information retrieved from the database: "
                + "\n" + "Property ID: " + PropertyID + "\tProperty Name: " + PropertyName
                + "\tAddress: " + Address + "\nPost Code: " + PostCode + "\n"
                + "Property Type: " + PropertyType + "\tProperty Price: " + PropertyPrice +
"\tResidence Status: " + ResidentStatus
                + "\n" + "Maximum Number Of Occupants Allowed: " + MaxNoOfOccupancy +
"\tMultiple Occupancy: " + MultipleOccupancy
                + "\nAre Pets Allowed? " + Pets_allowed + "\tAre Children Allowed To Live
In This Property? " + Children_allowed + "\n"
                + "Current Status (Whether if it is let/empty): " + CurrentStatus +
"\tLease Arrangement Date: " + LeaseArrDate
                + "\tIs the property available? " + IsAvailable;
        }
    }

}
```

```
ï»¿using System;
//using PropertyClient.ServiceReference1;
using PropertyClient.UOServiceReference;

namespace PropertyClient
{
    class Client
    {
        static void Main(string[] args)
        {

            //This is a client object, that will be used to invoke the service methods
created in web service.
            ServiceReference1.PropertyServiceClient rentServiceClient = new
ServiceReference1.PropertyServiceClient();

            //Web service object for converting postcode to uppercase
            UpperOperatorPortTypeClient clientUpper = new
UOServiceReference.UpperOperatorPortTypeClient("UpperOperatorSOAP12port_http");

            //while loop then have a switching case for the following options.
            do
            {
                Console.WriteLine("Welcome to Mike's Rental Service \n Please choose the
following options:");

                Console.WriteLine("Press 'a' to search a property");
                Console.WriteLine("Press 'b' to enter a new property");
                Console.WriteLine("Press 'c' to show available student lets");
                Console.WriteLine("Type stop to exit this program");
                Console.WriteLine("Write a letter to the corresponding option you want to
pick.");
                string choice = Console.ReadLine();

                switch (choice)
                {
                    case "a":
                          Console.WriteLine("Enter the postcode to search for properties:
");

                          string post_code = Console.ReadLine();
                          post_code = clientUpper.upper(post_code);
                          Console.WriteLine("Upper Case Post code -> " + post_code);

                        ServiceReference1.Property[] propertyList =
rentServiceClient.SearchPropertyByPostCode(post_code);

                       Console.WriteLine("Here are the list of properties with the
postcode you have provided!");
                       foreach (ServiceReference1.Property prop in propertyList)
//Iterating through the list and assigning all the values to the properties accordingly.
                       {
                           Console.WriteLine("Property Number: " + prop.PropertyID + "\n");
                           Console.WriteLine("Property Name: " + prop.PropertyName + "\n");
                           Console.WriteLine("Property Price: " + prop.PropertyPrice +
'\n');

                           Console.WriteLine("Property Address: " + prop.Address + "\n");
                           Console.WriteLine("Property Post Code: " + prop.PostCode +
"\n");

                           Console.WriteLine("Property Type: " + prop.PropertyType + "\n");
                           Console.WriteLine("Resident Status: " + prop.ResidentStatus +
"\n");

                           Console.WriteLine("Multiple Occupancy? " +
prop.MultipleOccupancy + "\n");
                           Console.WriteLine("Maximum Number Of Occupants: " +
prop.MaxNoOfOccupancy + "\n");
                           Console.WriteLine("Is property available to Students?" +
prop.IsAvailable + "\n");
                           Console.WriteLine("Are pets allowed in the property? " +
prop.Pets_allowed + "\n");
                           Console.WriteLine("Are children allowed in that property? " +
prop.Children_allowed + "\n");
                           Console.WriteLine("Is the property leased? " +
prop.CurrentStatus + "\n");
                           Console.WriteLine("Lease Arrangement Date: " +
prop.LeaseArrDate + "\n");

                       }
                       break;

                    case "b":
                        ServiceReference1.Property p = new ServiceReference1.Property();

                        Console.WriteLine("Enter a new property.\n");

                        Console.WriteLine("\nEnter Property Name");
                        p.PropertyName = Console.ReadLine();
                        Console.WriteLine("\nEnter Property Address");
                        p.Address = Console.ReadLine();
                        Console.WriteLine("\nEnter Property Postcode");
                        string  p_code = Console.ReadLine();
                        p.PostCode = clientUpper.upper(p_code);
                        Console.WriteLine("\nEnter Property Type");
                        p.PropertyType = Console.ReadLine();
                        Console.WriteLine("\nEnter Property Price");
                        p.PropertyPrice = Console.ReadLine();
                        Console.WriteLine("\nEnter the Max Number Of Occupancy for this
property");
                        p.MaxNoOfOccupancy = int.Parse(Console.ReadLine());
                        Console.WriteLine("\nIs this property allowed to have multiple
occupancy? (True/False)");
                        p.MultipleOccupancy = Boolean.Parse(Console.ReadLine());
                        Console.WriteLine("\nEnter the allowed resident status for this
property. (Student/Non-Student)");
                        p.ResidentStatus = Console.ReadLine();
                        Console.WriteLine("\nAre pets allowed in this property? (yes/no)");
                        p.Pets_allowed = Console.ReadLine();
                        Console.WriteLine("\nAre children allowed to live in this property?
(yes/no)");
                        p.Children_allowed = Console.ReadLine();
                        Console.WriteLine("\nWhat is the current status of this property?
Has it been leased? (empty - false/let - true)");
                        p.CurrentStatus = Boolean.Parse(Console.ReadLine());
                        Console.WriteLine("\nEnter the lease arrangement date for this
property? Otherwise, specify n/a");
                        p.LeaseArrDate = Console.ReadLine();
                        Console.WriteLine("\nIs this property available to students? (false
- no, true - yes)");
                        p.IsAvailable = Boolean.Parse(Console.ReadLine());

                        Console.WriteLine(rentServiceClient.EnterNewProperty(p)); //calling
the service method to add a new property.
                        Console.WriteLine("The property which you provided the information
for has been added successfully to the database.");
                        break;

                    case "c":
```

```
                        Console.WriteLine("Here are all the available student
properties\n");
                        ServiceReference1.Property[] studentPropList =
rentServiceClient.GetAllPropertiesForStudents();

                        foreach (ServiceReference1.Property studentProp in studentPropList)
                        {
                            Console.WriteLine("Property Number: " + studentProp.PropertyID
+ "\n");
                            Console.WriteLine("Property Name: " + studentProp.PropertyName
+ "\n");
                            Console.WriteLine("Property Price: " +
studentProp.PropertyPrice + '\n');
                            Console.WriteLine("Property Address: " + studentProp.Address +
"\n");
                            Console.WriteLine("Property Post Code: " + studentProp.PostCode
+ "\n");
                            Console.WriteLine("Property Type: " + studentProp.PropertyType
+ "\n");
                            Console.WriteLine("Resident Status: " +
studentProp.ResidentStatus + "\n");
                            Console.WriteLine("Multiple Occupancy? " +
studentProp.MultipleOccupancy + "\n");
                            Console.WriteLine("Maximum Number Of Occupants: " +
studentProp.MaxNoOfOccupancy + "\n");
                            Console.WriteLine("Is property available to Students?" +
studentProp.IsAvailable + "\n");
                            Console.WriteLine("Are pets allowed in the property? " +
studentProp.Pets_allowed + "\n");
                            Console.WriteLine("Are children allowed in that property? " +
studentProp.Children_allowed + "\n");
                            Console.WriteLine("Is the property leased? " +
studentProp.CurrentStatus + "\n");
                            Console.WriteLine("Lease Arrangement Date: " +
studentProp.LeaseArrDate + "\n");

                        }

                        break;

                    default:
                            Console.WriteLine("Mike's Rental Company, find the property for
you!");
                        break;
                }

            }
            while (!Console.ReadLine().Equals("stop"));

        }
    }
}
```

```
ï»¿//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:4.0.30319.42000
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace PropertyClient.ServiceReference1 {
    using System.Runtime.Serialization;
    using System;


    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.CodeDom.Compiler.GeneratedCodeAttribute("System.Runtime.Serialization",
"4.0.0.0")]
    [System.Runtime.Serialization.DataContractAttribute(Name="Property",
Namespace="http://schemas.datacontract.org/2004/07/PropertyService")]
    [System.SerializableAttribute()]
    public partial class Property : object,
System.Runtime.Serialization.IExtensibleDataObject,
System.ComponentModel.INotifyPropertyChanged {

        [System.NonSerializedAttribute()]
        private System.Runtime.Serialization.ExtensionDataObject extensionDataField;

        [System.Runtime.Serialization.OptionalFieldAttribute()]
        private string AddressField;

        [System.Runtime.Serialization.OptionalFieldAttribute()]
        private string Children_allowedField;

        [System.Runtime.Serialization.OptionalFieldAttribute()]
        private bool CurrentStatusField;

        [System.Runtime.Serialization.OptionalFieldAttribute()]
        private bool IsAvailableField;

        [System.Runtime.Serialization.OptionalFieldAttribute()]
        private string LeaseArrDateField;

        [System.Runtime.Serialization.OptionalFieldAttribute()]
        private int MaxNoOfOccupancyField;

        [System.Runtime.Serialization.OptionalFieldAttribute()]
        private bool MultipleOccupancyField;

        [System.Runtime.Serialization.OptionalFieldAttribute()]
        private string Pets_allowedField;

        [System.Runtime.Serialization.OptionalFieldAttribute()]
        private string PostCodeField;

        [System.Runtime.Serialization.OptionalFieldAttribute()]
        private int PropertyIDField;

        [System.Runtime.Serialization.OptionalFieldAttribute()]
        private string PropertyNameField;

        [System.Runtime.Serialization.OptionalFieldAttribute()]
        private string PropertyPriceField;

        [System.Runtime.Serialization.OptionalFieldAttribute()]
        private string PropertyTypeField;

        [System.Runtime.Serialization.OptionalFieldAttribute()]
        private string ResidentStatusField;

        [global::System.ComponentModel.BrowsableAttribute(false)]
        public System.Runtime.Serialization.ExtensionDataObject ExtensionData {
            get {
                return this.extensionDataField;
            }
            set {
                this.extensionDataField = value;
            }
        }

        [System.Runtime.Serialization.DataMemberAttribute()]
        public string Address {
            get {
                return this.AddressField;
            }
            set {
                if ((object.ReferenceEquals(this.AddressField, value) != true)) {
                    this.AddressField = value;
                    this.RaisePropertyChanged("Address");
                }
            }
        }

        [System.Runtime.Serialization.DataMemberAttribute()]
        public string Children_allowed {
            get {
                return this.Children_allowedField;
            }
            set {
                if ((object.ReferenceEquals(this.Children_allowedField, value) != true)) {
                    this.Children_allowedField = value;
                    this.RaisePropertyChanged("Children_allowed");
                }
            }
        }

        [System.Runtime.Serialization.DataMemberAttribute()]
        public bool CurrentStatus {
            get {
                return this.CurrentStatusField;
            }
            set {
                if ((this.CurrentStatusField.Equals(value) != true)) {
                    this.CurrentStatusField = value;
                    this.RaisePropertyChanged("CurrentStatus");
                }
            }
        }

        [System.Runtime.Serialization.DataMemberAttribute()]
        public bool IsAvailable {
            get {
                return this.IsAvailableField;
            }
            set {
                if ((this.IsAvailableField.Equals(value) != true)) {
```

```
            this.IsAvailableField = value;
            this.RaisePropertyChanged("IsAvailable");
        }
    }
}

[System.Runtime.Serialization.DataMemberAttribute()]
public string LeaseArrDate {
    get {
        return this.LeaseArrDateField;
    }
    set {
        if ((object.ReferenceEquals(this.LeaseArrDateField, value) != true)) {
            this.LeaseArrDateField = value;
            this.RaisePropertyChanged("LeaseArrDate");
        }
    }
}

[System.Runtime.Serialization.DataMemberAttribute()]
public int MaxNoOfOccupancy {
    get {
        return this.MaxNoOfOccupancyField;
    }
    set {
        if ((this.MaxNoOfOccupancyField.Equals(value) != true)) {
            this.MaxNoOfOccupancyField = value;
            this.RaisePropertyChanged("MaxNoOfOccupancy");
        }
    }
}

[System.Runtime.Serialization.DataMemberAttribute()]
public bool MultipleOccupancy {
    get {
        return this.MultipleOccupancyField;
    }
    set {
        if ((this.MultipleOccupancyField.Equals(value) != true)) {
            this.MultipleOccupancyField = value;
            this.RaisePropertyChanged("MultipleOccupancy");
        }
    }
}

[System.Runtime.Serialization.DataMemberAttribute()]
public string Pets_allowed {
    get {
        return this.Pets_allowedField;
    }
    set {
        if ((object.ReferenceEquals(this.Pets_allowedField, value) != true)) {
            this.Pets_allowedField = value;
            this.RaisePropertyChanged("Pets_allowed");
        }
    }
}

[System.Runtime.Serialization.DataMemberAttribute()]
public string PostCode {
    get {
        return this.PostCodeField;
    }
    set {
        if ((object.ReferenceEquals(this.PostCodeField, value) != true)) {
            this.PostCodeField = value;
            this.RaisePropertyChanged("PostCode");
        }
    }
}

[System.Runtime.Serialization.DataMemberAttribute()]
public int PropertyID {
    get {
        return this.PropertyIDField;
    }
    set {
        if ((this.PropertyIDField.Equals(value) != true)) {
            this.PropertyIDField = value;
            this.RaisePropertyChanged("PropertyID");
        }
    }
}

[System.Runtime.Serialization.DataMemberAttribute()]
public string PropertyName {
    get {
        return this.PropertyNameField;
    }
    set {
        if ((object.ReferenceEquals(this.PropertyNameField, value) != true)) {
            this.PropertyNameField = value;
            this.RaisePropertyChanged("PropertyName");
        }
    }
}

[System.Runtime.Serialization.DataMemberAttribute()]
public string PropertyPrice {
    get {
        return this.PropertyPriceField;
    }
    set {
        if ((object.ReferenceEquals(this.PropertyPriceField, value) != true)) {
            this.PropertyPriceField = value;
            this.RaisePropertyChanged("PropertyPrice");
        }
    }
}

[System.Runtime.Serialization.DataMemberAttribute()]
public string PropertyType {
    get {
        return this.PropertyTypeField;
    }
    set {
        if ((object.ReferenceEquals(this.PropertyTypeField, value) != true)) {
            this.PropertyTypeField = value;
            this.RaisePropertyChanged("PropertyType");
        }
    }
}

[System.Runtime.Serialization.DataMemberAttribute()]
public string ResidentStatus {
    get {
```

```
            return this.ResidentStatusField;
        }
        set {
            if ((object.ReferenceEquals(this.ResidentStatusField, value) != true)) {
                this.ResidentStatusField = value;
                this.RaisePropertyChanged("ResidentStatus");
            }
        }
    }

    public event System.ComponentModel.PropertyChangedEventHandler PropertyChanged;

    protected void RaisePropertyChanged(string propertyName) {
        System.ComponentModel.PropertyChangedEventHandler propertyChanged =
this.PropertyChanged;
        if ((propertyChanged != null)) {
            propertyChanged(this, new
System.ComponentModel.PropertyChangedEventArgs(propertyName));
        }
    }
}

[System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel", "4.0.0.0")]

[System.ServiceModel.ServiceContractAttribute(ConfigurationName="ServiceReference1.IProperty
Service")]
public interface IPropertyService {


[System.ServiceModel.OperationContractAttribute(Action="http://tempuri.org/IPropertyService/
SearchPropertyByPostCode",
ReplyAction="http://tempuri.org/IPropertyService/SearchPropertyByPostCodeResponse")]
    PropertyClient.ServiceReference1.Property[] SearchPropertyByPostCode(string
postCode);


[System.ServiceModel.OperationContractAttribute(Action="http://tempuri.org/IPropertyService/
SearchPropertyByPostCode",
ReplyAction="http://tempuri.org/IPropertyService/SearchPropertyByPostCodeResponse")]
    System.Threading.Tasks.Task<PropertyClient.ServiceReference1.Property[]>
SearchPropertyByPostCodeAsync(string postCode);


[System.ServiceModel.OperationContractAttribute(Action="http://tempuri.org/IPropertyService/
EnterNewProperty",
ReplyAction="http://tempuri.org/IPropertyService/EnterNewPropertyResponse")]
    bool EnterNewProperty(PropertyClient.ServiceReference1.Property p);


[System.ServiceModel.OperationContractAttribute(Action="http://tempuri.org/IPropertyService/
EnterNewProperty",
ReplyAction="http://tempuri.org/IPropertyService/EnterNewPropertyResponse")]
    System.Threading.Tasks.Task<bool>
EnterNewPropertyAsync(PropertyClient.ServiceReference1.Property p);


[System.ServiceModel.OperationContractAttribute(Action="http://tempuri.org/IPropertyService/
GetAllPropertiesForStudents",
ReplyAction="http://tempuri.org/IPropertyService/GetAllPropertiesForStudentsResponse")]
    PropertyClient.ServiceReference1.Property[] GetAllPropertiesForStudents();


[System.ServiceModel.OperationContractAttribute(Action="http://tempuri.org/IPropertyService/
GetAllPropertiesForStudents",
ReplyAction="http://tempuri.org/IPropertyService/GetAllPropertiesForStudentsResponse")]
    System.Threading.Tasks.Task<PropertyClient.ServiceReference1.Property[]>
GetAllPropertiesForStudentsAsync();
}

[System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel", "4.0.0.0")]
public interface IPropertyServiceChannel :
PropertyClient.ServiceReference1.IPropertyService, System.ServiceModel.IClientChannel {
}

[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel", "4.0.0.0")]
public partial class PropertyServiceClient :
System.ServiceModel.ClientBase<PropertyClient.ServiceReference1.IPropertyService>,
PropertyClient.ServiceReference1.IPropertyService {

    public PropertyServiceClient() {
    }

    public PropertyServiceClient(string endpointConfigurationName) :
            base(endpointConfigurationName) {
    }

    public PropertyServiceClient(string endpointConfigurationName, string
remoteAddress) :
            base(endpointConfigurationName, remoteAddress) {
    }

    public PropertyServiceClient(string endpointConfigurationName,
System.ServiceModel.EndpointAddress remoteAddress) :
            base(endpointConfigurationName, remoteAddress) {
    }

    public PropertyServiceClient(System.ServiceModel.Channels.Binding binding,
System.ServiceModel.EndpointAddress remoteAddress) :
            base(binding, remoteAddress) {
    }

    public PropertyClient.ServiceReference1.Property[] SearchPropertyByPostCode(string
postCode) {
        return base.Channel.SearchPropertyByPostCode(postCode);
    }

    public System.Threading.Tasks.Task<PropertyClient.ServiceReference1.Property[]>
SearchPropertyByPostCodeAsync(string postCode) {
        return base.Channel.SearchPropertyByPostCodeAsync(postCode);
    }

    public bool EnterNewProperty(PropertyClient.ServiceReference1.Property p) {
        return base.Channel.EnterNewProperty(p);
    }

    public System.Threading.Tasks.Task<bool>
EnterNewPropertyAsync(PropertyClient.ServiceReference1.Property p) {
        return base.Channel.EnterNewPropertyAsync(p);
    }

    public PropertyClient.ServiceReference1.Property[] GetAllPropertiesForStudents() {
        return base.Channel.GetAllPropertiesForStudents();
    }

    public System.Threading.Tasks.Task<PropertyClient.ServiceReference1.Property[]>
```

```
GetAllPropertiesForStudentsAsync() {
            return base.Channel.GetAllPropertiesForStudentsAsync();
        }
    }
}
```

```
ï»¿//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:4.0.30319.42000
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace PropertyClient.UOServiceReference {


    [System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel", "4.0.0.0")]
    [System.ServiceModel.ServiceContractAttribute(Namespace="http://service.bpel",
ConfigurationName="UOServiceReference.UpperOperatorPortType")]
    public interface UpperOperatorPortType {

        // CODEGEN: Parameter 'return' requires additional schema information that cannot
be captured using the parameter mode. The specific attribute is
'System.Xml.Serialization.XmlElementAttribute'.
        [System.ServiceModel.OperationContractAttribute(Action="urn:upper",
ReplyAction="urn:upperResponse")]
        [System.ServiceModel.XmlSerializerFormatAttribute(SupportFaults=true)]
        [return: System.ServiceModel.MessageParameterAttribute(Name="return")]
        PropertyClient.UOServiceReference.upperResponse
upper(PropertyClient.UOServiceReference.upperRequest request);

        [System.ServiceModel.OperationContractAttribute(Action="urn:upper",
ReplyAction="urn:upperResponse")]
        System.Threading.Tasks.Task<PropertyClient.UOServiceReference.upperResponse>
upperAsync(PropertyClient.UOServiceReference.upperRequest request);
    }

    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel", "4.0.0.0")]

[System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.A
dvanced)]
    [System.ServiceModel.MessageContractAttribute(WrapperName="upper",
WrapperNamespace="http://service.bpel", IsWrapped=true)]
    public partial class upperRequest {

        [System.ServiceModel.MessageBodyMemberAttribute(Namespace="http://service.bpel",
Order=0)]
        [System.Xml.Serialization.XmlElementAttribute(IsNullable=true)]
        public string str;

        public upperRequest() {
        }

        public upperRequest(string str) {
            this.str = str;
        }
    }

    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel", "4.0.0.0")]

[System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.A
dvanced)]
    [System.ServiceModel.MessageContractAttribute(WrapperName="upperResponse",
WrapperNamespace="http://service.bpel", IsWrapped=true)]
    public partial class upperResponse {

        [System.ServiceModel.MessageBodyMemberAttribute(Namespace="http://service.bpel",
Order=0)]
        [System.Xml.Serialization.XmlElementAttribute(IsNullable=true)]
        public string @return;

        public upperResponse() {
        }

        public upperResponse(string @return) {
            this.@return = @return;
        }
    }

    [System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel", "4.0.0.0")]
    public interface UpperOperatorPortTypeChannel :
PropertyClient.UOServiceReference.UpperOperatorPortType, System.ServiceModel.IClientChannel
{
    }

    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel", "4.0.0.0")]
    public partial class UpperOperatorPortTypeClient :
System.ServiceModel.ClientBase<PropertyClient.UOServiceReference.UpperOperatorPortType>,
PropertyClient.UOServiceReference.UpperOperatorPortType {

        public UpperOperatorPortTypeClient() {
        }

        public UpperOperatorPortTypeClient(string endpointConfigurationName) :
                base(endpointConfigurationName) {
        }

        public UpperOperatorPortTypeClient(string endpointConfigurationName, string
remoteAddress) :
                base(endpointConfigurationName, remoteAddress) {
        }

        public UpperOperatorPortTypeClient(string endpointConfigurationName,
System.ServiceModel.EndpointAddress remoteAddress) :
                base(endpointConfigurationName, remoteAddress) {
        }

        public UpperOperatorPortTypeClient(System.ServiceModel.Channels.Binding binding,
System.ServiceModel.EndpointAddress remoteAddress) :
                base(binding, remoteAddress) {
        }


[System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.A
dvanced)]
        PropertyClient.UOServiceReference.upperResponse
PropertyClient.UOServiceReference.UpperOperatorPortType.upper(PropertyClient.UOServiceRefere
nce.upperRequest request) {
            return base.Channel.upper(request);
        }

        public string upper(string str) {
            PropertyClient.UOServiceReference.upperRequest inValue = new
PropertyClient.UOServiceReference.upperRequest();
            inValue.str = str;
            PropertyClient.UOServiceReference.upperResponse retVal =
```

```
((PropertyClient.UOServiceReference.UpperOperatorPortType)(this)).upper(inValue);
            return retVal.@return;
        }


[System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.A
dvanced)]
        System.Threading.Tasks.Task<PropertyClient.UOServiceReference.upperResponse>
PropertyClient.UOServiceReference.UpperOperatorPortType.upperAsync(PropertyClient.UOServiceR
eference.upperRequest request) {
            return base.Channel.upperAsync(request);
        }

        public System.Threading.Tasks.Task<PropertyClient.UOServiceReference.upperResponse>
upperAsync(string str) {
            PropertyClient.UOServiceReference.upperRequest inValue = new
PropertyClient.UOServiceReference.upperRequest();
            inValue.str = str;
            return
((PropertyClient.UOServiceReference.UpperOperatorPortType)(this)).upperAsync(inValue);
        }
    }
}
```

In this project, I have attempted and completed all of the four questions.
For question one, I made a console application called 'PropertySearch', which consists of the following classes: Flat, StudentFlat, House and Property.
I made an abstract class called 'Property' where all of the classes inherit from.
I did this in order to capture the criteria described in the task.

For task 2, I created my database, the following tables called lettings and property and some test data for testing purposes.
NOTE - Please consult the property.sql file to see how I have done this.

For task 3 and 4, I made a separate solution from the console application in task 1.
The service I created is called 'PropertyService' and the corresponding client app is called 'PropertyClient'.
For task four I used a switch case to display the text choice. And in the textual menu, I have assigned letters 'a','b','c' as choices for the user to select from.
Each choice correspond to the following operations as described in task 3 and 4 of the assignment.

```
ï»¿using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PropertySearch
{

    class House : Property
    {
        private bool multiple_occupancy;
        private bool available_to_students;

        public bool Multiple_occupancy { get => multiple_occupancy; set =>
multiple_occupancy = value; }
        public bool Available_to_students { get => available_to_students; set =>
available_to_students = value; }

        //Overriding of the ToString method - print all of the values.
       override
       public string ToString()
        {
            return "Property information retrieved from the database: "
                + "\n" + "Property ID: " + PropertyID + "\tProperty Name: " + PropertyName
                + "\tAddress: " + Address + "\nPost Code: " + PostCode + "\n"
                + "Property Type: " + PropertyType + "\tProperty Price: " + PropertyPrice +
"\tResidence Status: " + ResidentStatus
                + "\n" + "Multiple Occupancy: " + this.multiple_occupancy + "\tIs it
available to students? " + this.available_to_students
                + "\tCurrent Status: " + CurrentStatus + "\tCurrent Lease Arrangement Date:
" + LeaseArrangement;
        }

    }
}
```

```
ï»¿using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PropertySearch
{
    //Class for a non student flat
    class Flat : Property
    {
        private bool pets_allowed;
        private bool children_allowed;

        public bool PetsAllowed
        {
            get => pets_allowed;
            set => pets_allowed = value;
        }

        public bool ChildrenAllowed
        {
            get => children_allowed;
            set => children_allowed = value;
        }


        //This method will be responsible for printing the values.
        override
        public string ToString()
        {
            return "Property information for Non Student Flats: "
                + "\n" + "Property ID: " + PropertyID + "\tProperty Name: " + PropertyName
                + "\tAddress: " + Address + "\nPost Code: " + PostCode + "\n"
                + "Property Type: " + PropertyType + "\tProperty Price: " + PropertyPrice +
"\tResidence Status: " + ResidentStatus
                + "\nAre Pets Allowed? " + this.pets_allowed + "\tAre Children Allowed To
Live In This Property? " + this.children_allowed + "\n"
                 + "\tCurrent Status: " + CurrentStatus + "\tCurrent Lease Arrangement
Date: " + LeaseArrangement;

        }

    }
}
```

```
ï»¿using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PropertySearch
{

    abstract class Property
    {

        //Data members
        private int id;
        private string name;
        private string address;
        private string postCode;
        private string type;
        private string resident_status;
        private string price;

        private string currentStatus; //indicates whether a property is leased or not
(empty/let).
        private string lease_arrangement;

        //Encapsulate Fields
        public int PropertyID
        {
            get => id;
            set => id = value;
        }

        public string PropertyName
        {
            get => name;
            set => name = value;
        }

        public string Address
        {
            get => address;
            set => address = value;
        }

        public string PostCode
        {
            get => postCode;
            set => postCode = value;
        }

        public string PropertyType
        {
            get => type;
            set => type = value;
        }

        public string PropertyPrice
        {
            get => price;
            set => price = value;
        }

        public string ResidentStatus
        {
            get => resident_status;
            set => resident_status = value;
        }


        public string CurrentStatus
        {
            get => currentStatus;
            set => currentStatus = value;
        }
        public string LeaseArrangement
        {
            get => lease_arrangement;
            set => lease_arrangement = value;
        }
    }
}
```

```
ï»¿using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;



namespace PropertySearch
{
    class PropertyMain
    {

        static void Main(string[] args)
        {
            //Object initialisation for each property type.
            StudentFlat student = new StudentFlat
            {
                PropertyID = 1,
                PropertyName = "Nixon Court Accomodation",
                PropertyType = "Student Flat",
                PropertyPrice = "Â£430 pcm",
                Address = "33 Putney Road, Leicester",
                PostCode = "LE2 7GH",
                ResidentStatus = "Student",
                MaxNoOfOccupancy = 6,
                LeaseArrangement = "n/a",
                CurrentStatus = "empty",
            };

            House house = new House
            {
                PropertyID = 01,
                PropertyName = "Regent House",
                PropertyPrice = "480pcm",
                PropertyType = "House",
                ResidentStatus = "Non-Student",
                Available_to_students = false,
                CurrentStatus = "let",
                PostCode = "LE3 7PT",
                Address = "Postman Way, Welford Road, Leicester",
                LeaseArrangement = "14-07-2018 until 14-07-2019"
            };

            House stHouse = new House
            {
                PropertyID = 02,
                PropertyName = "Regent House",
                PropertyPrice = "480pcm",
                PropertyType = "House",
                Available_to_students = true,
                CurrentStatus = "empty",
                PostCode = "LU8 2BP",
                Address = "Trueman Street, Beeston, Luton",
                LeaseArrangement = "n/a",
                ResidentStatus = "Student"
            };

            Flat nonStudentFlat = new Flat
            {
                PropertyID = 1,
                PropertyName = "Merion Heights",
                PropertyPrice = "500pcm",
                PropertyType = "Flat",
                ResidentStatus = "Non-student",
                PetsAllowed = false,
                ChildrenAllowed = true,
                LeaseArrangement = "n/a",
                CurrentStatus = "empty",
                Address = "39 Shallow Avenue, Woodgreen, London",
                PostCode = "N9 8GH"

            };

            //Display the following properties
            Console.WriteLine("Here is some information on a student flat");
            Console.WriteLine(student.ToString() + "\n");
            Console.WriteLine("Information on a house (which is not offered to students!)");
            Console.WriteLine(house.ToString() + "\n");
            Console.WriteLine("Information on a student house");
            Console.WriteLine(stHouse.ToString() + "\n");
            Console.WriteLine("Here is some information for a non-student flat");
            Console.WriteLine(nonStudentFlat.ToString() + "\n");



            String s = Console.ReadLine();
        }
    }
}
```

```
ï»¿using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PropertySearch
{
    //Class for the Student flat
    class StudentFlat : Property
    {
        private int maximum_no_of_occupancy;

        public int MaxNoOfOccupancy
        {
            get => maximum_no_of_occupancy;
            set => maximum_no_of_occupancy = value;
        }

        override
        public string ToString()
        {
            return "Property information for Student Flats: "
                + "\n" + "Property ID: " + PropertyID + "\tProperty Name: " + PropertyName
                + "\tAddress: " + Address + "\nPost Code: " + PostCode + "\n"
                + "Property Type: " + PropertyType + "\tProperty Price: " + PropertyPrice +
"\tResidence Status: " + ResidentStatus
                + "\n" + "Maximum Number Of Occupants Allowed: " +
this.maximum_no_of_occupancy
                + "\tCurrent Status: " + CurrentStatus + "\nCurrent Lease Arrangement Date:
" + LeaseArrangement;
        }
    }
}
```

Name: Mike [redacted] (ms853)
Assignment: Coursework 1
Date Submitted: Thursday, 8 November 2018 21:03:18 o'clock GMT
Current Grade: Needs Grading

Submission Field:

For task 1, I made a console application called 'PropertySearch', which consists of the following classes: Flat, StudentFlat, House and Property.
I made an abstract class called 'Property' where all of the classes inherit from.Â I did this in order to capture the criteria described in the task.
For task 2, I created my database, the following tables called lettings and property and some test data for testing purposes.
NOTE - Please consult the property.sql file to see how I have done this.
For task 3 and 4, I made a separate solution from the console application in task 1.Â The service I created is called 'PropertyService' and the corresponding client application is called 'PropertyClient'.
For task four I used a switch case to display the text choice.Â And in the textual menu, I have assigned letters 'a','b','c' as choices for the user to select from. Each choice corresponds to the following operations as described in task 3 and 4 of the assignment.Â
My Assignment1.zip folder contains my property.sqlÂ which contains MySQLÂ script for creating my schema and tables, two separate solutions and a solution_readme.txt file which outlines what I have done for this assignment.Â

Comments:
There are no student comments for this assignment.

Files:
	Original filename: Assignment1.zip
	Filename: Coursework 1_ms853_attempt_2018-11-08-21-03-18_Assignment1.zip