

Jonathan Binns Coursework 1 Person Class Listing and Form Screenshots

Person Class Listing

The using statements and namespace have been omitted.

```
public class Customer
{
    //initializes the variables to be used as private
    private int _customerID;
    private string _FirstName;
    private string _Surname;
    private string _EmailAddress;
    private string _SkypeID;
    private string _Telephone;
    private string _PreferredContact;
    private int _Matric;

    //method to get/set the customers ID
    public int ID
    {
        get
        {
            return _customerID;
        }
        set
        {
            //if-else statement to throw an exception if the ID is going to be set
            //to something outside of the specified range
            if (value >= 10001 && value <= 50000)
            {
                _customerID = value;
            }
            else
            {
                throw new ArgumentException("ID is out of range. The range is
10001 - 50000");
            }
        }
    }

    //method to get/set the customers first name
    public string FirstName
    {
        get
        {
            return _FirstName;
        }
        set
        {
            //if-else statement to throw an exception if the first name is going to
            //be set to a blank string
            if (value != string.Empty)
            {
                _FirstName = value;
            }
            else
            {
                throw new ArgumentException("First name cannot be empty");
            }
        }
    }
}
```

```

        {
            throw new ArgumentException("First Name cannot be empty");
        }
    }
}

//method to get/set the customers Surname
public string Surname
{
    get
    {
        return _Surname;
    }
    set
    {
        //if-else statement to throw an exception if the surname is going to
        be set to a blank string
        if (value != string.Empty)
        {
            _Surname = value;
        }
        else
        {
            throw new ArgumentException("Surname cannot be empty");
        }
    }
}

//method to get/set the customers email address
public string EmailAddress
{
    get
    {
        return _EmailAddress;
    }
    set
    {
        //if-else statement to throw an exception if the string the email
        address will be set to a string that doesnt contain an @
        if (value.Contains("@"))
        {
            _EmailAddress = value;
        }
        else
        {
            throw new ArgumentException("Please enter a valid email");
        }
    }
}

//method to get/set the customers skype ID
public string SkypeID
{
    get
    {
        return _SkypeID;
    }
    set
    {
        _SkypeID = value;
    }
}

```

```

    }
}

//method to get/set the customers telephone number
public string Telephone
{
    get
    {
        return _Telephone;
    }
    set
    {
        //if-else statement to throw an exception if the telephone number is
going to be set to a blank string
        if (value != string.Empty)
        {
            _Telephone = value;
        }
        else
        {
            throw new ArgumentException("Telephone cannot be empty");
        }
    }
}

//method to get/set the customers preferred contact
public string PreferredContact
{
    get
    {
        return _PreferredContact;
    }
    set
    {
        //if-else statement to throw an exception if the preferred contact is
going to be set to something that isnt 'Skype', 'Telephone' or 'Email'.
        if (value == "Skype" || value == "Telephone" || value == "Email")
        {
            _PreferredContact = value;
        }
        else
        {
            throw new ArgumentException("Please enter either Email, Telephone
or Skype");
        }
    }
}

//method to get/set the customers Matric
public int Matric
{
    get
    {
        return _Matric;
    }
    set
    {
        _Matric = value;
    }
}

```

```

        //method to return a string with the method of preferred contact with that
method of contacts data
        public string getPreferredContact()
        {
            //if statements to return a string with the preferred method of contact
and that methods data
            if (_PreferredContact == "Skype")
            {
                return "Skype: " + _SkypeID;
            }

            if (_PreferredContact == "Email")
            {
                return "Email: " + _EmailAddress;
            }

            if (_PreferredContact == "Telephone")
            {
                return "Telephone: " + _Telephone;
            }
            //otherwise it returns null
            else
            {
                return null;
            }
        }

        //method to return all of the customers attributes
        public string Display()
        {
            return _customerID + " Name: " + _FirstName + " " + _Surname + " Email
Address: " + _EmailAddress + " Skype ID: " + _SkypeID + " Telephone: " + _Telephone +
" Preferred Contact: " + _PreferredContact + Environment.NewLine;

        }
    }
}

```

Screenshots of Form

Blank

The screenshot shows a web form titled "Please enter the customers data" with a dark header bar. The form contains several input fields: ID (with the value "10001"), First Name, Surname, Email Address, Skype ID, Telephone, and Preferred Contact. To the right of the form is a large empty box labeled "Previous Customers". Below the form, there is a prompt "Please enter either, Email, Skype or Telephone" and a row of five buttons: "Get Preferred Contact", "Delete Customer", "Find Customer", "Save Customer", and "Clear Form". Below these buttons is a single button labeled "List All".

With Data Filled Out

Please enter the customers data

ID:10001

First Name:Test

Surname:McTestyson

Email Address:T.Testyson@gmail.com

Skype ID:T.Testyson

Telephone:01189998819991197253

Preferred Contact:Email

Please enter either, Email, Skype or Telephone

Get Preferred Contact

Delete Customer

Find Customer

Save Customer

Clear Form

List All

Previous Customers

With one Customer Already Added

Please enter the customers data

ID:10002

First Name:Test2

Surname:Test2

Email Address:Test2@email.com

Skype ID:Test2

Telephone:1234567890

Preferred Contact:Skype

Please enter either, Email, Skype or Telephone

Get Preferred Contact

Delete Customer

Find Customer

Save Customer

Clear Form

List All

Previous Customers

With one Customer in List

All Customers

10001 Name: Test McTestyson Email Address: T.Testyson@gmail.com Skype ID: T.Testyson Telephone: 01189998819991197253 Preferred Contact: Email