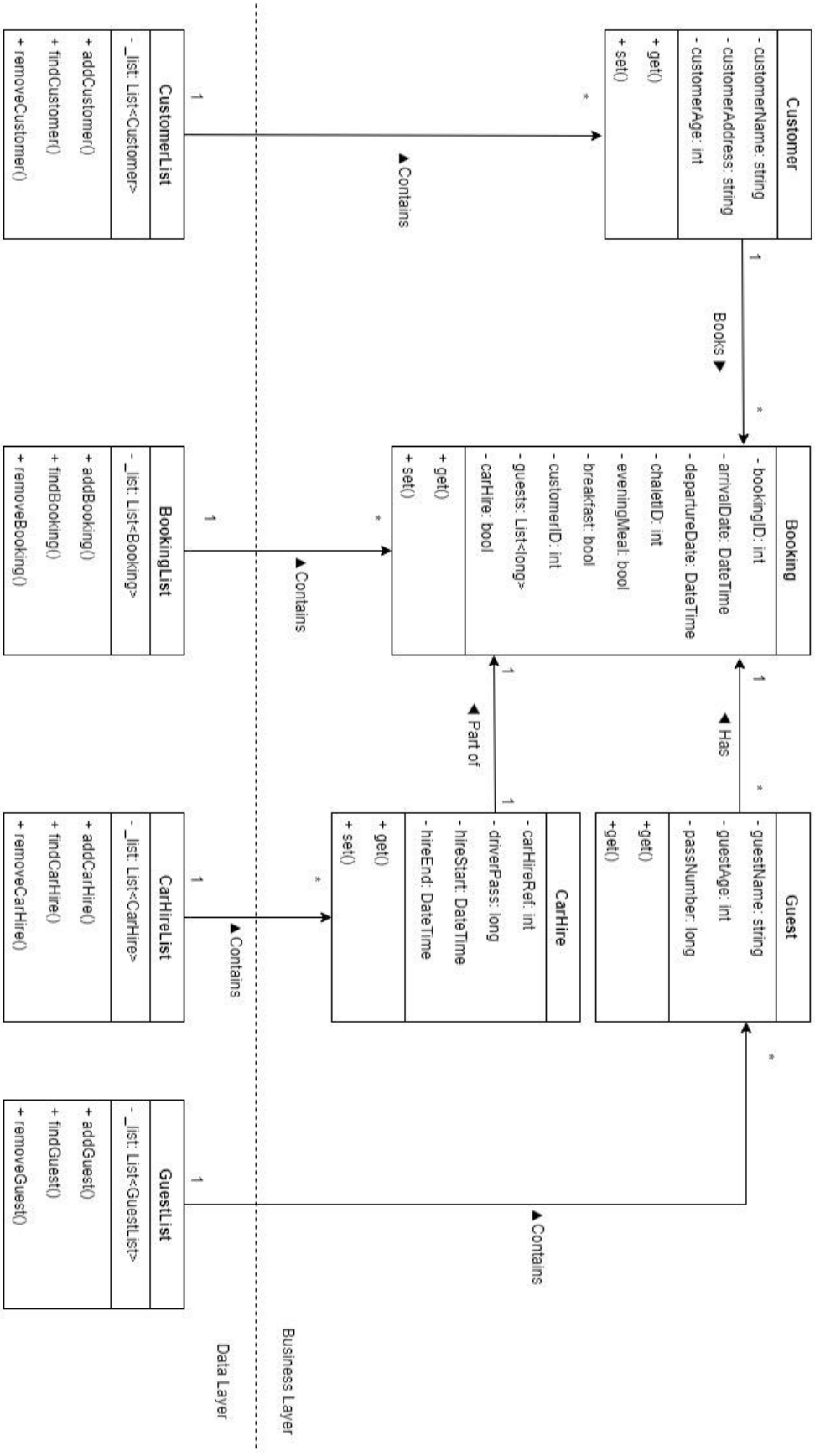
40275286 - Christopher Johnson



What advantages are of the 3-layered approach to building applications?

Three levelled architecture is a design approach which incorporates a Data, Business and Presentation layer to the software being developed. Isolation of these separate layers allows the modification of one layer with minimal impact to another, allowing separate groups of people with different skill sets/specialities to work on a specific aspect of the program.

When designing a business layer, it is important that business rules are taken into consideration. Three levelled architecture has all the business logic is in one place (business layer) which allows a business rule to be changed/updated without needing a rework of the data or presentation layers, saving man hours and making the software more flexible.

Reusability is an advantage of three levelled architecture. A class within the business layer of a piece of software can be reused at the presentation layer many times, meaning less overall code and reducing the chance of error.

The isolation of the layers is also advantageous when it comes to upgrading the system. The database layer, for example could be updated to a new system without affecting other areas of the application.

With an example, explain why using design patterns can make the design of an OO system easier to understand?

Design patterns are solutions in the form of templates that are used to solve problems encountered whilst designing an object-oriented system. They can make the design of a system easier to understand as they are recognisable to developers. When working in a group the use of design patterns makes communication between developers easier.

If presented with a specific problem, for example wanting to ensure an object is not instantiated more than once (this may be due to it's large size, resulting in a waste of resources), a design pattern can act as an easy to use template that is not tied to any specific language. In this case a Singleton Pattern would be used as it restricts the instantiation of a class to one object.

Design patterns are tried and tested solutions, reducing the chance of developer error. A developer can be sure that a design pattern will work as planned. This does not mean that design patterns are not flexible, they can be altered for efficiency in a specific scenario.

Documentation of code is easier using design patterns as they are already well documented and just by naming the pattern used in your code will help anyone reading your code to fully understand what your code is achieving.

Booking.cs

```
// Author: Christopher Johnson [40275286]
// Class Purpose: Booking class to store booking information
// Date Last Modified: 10/12/2017

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Business
{
    public class Booking
    {
        // Declaring attributes for the booking class
        private int _bookingID;
        private DateTime _arrivalDate;
        private DateTime _departureDate;
        private int _chaletID;
        private bool _eveningMeal;
        private bool _breakfast;
        private int _customerID;
        private List<long> _guests;
        private bool _carHire;


        public int bookingID
        {
            get
            {
                return _bookingID;
            }
            set
            {
                _bookingID = value;
            }
        }

        public DateTime arrivalDate
        {
            get
            {
                return _arrivalDate;
            }
            set
            {
                // Validate
                if (value == null)
                {
                    throw new ArgumentException("Arrival Date cannot be blank");
                }
                _arrivalDate = value;
            }
        }

        public DateTime departureDate
        {
            get
            {
                return _departureDate;
            }
            set
            {
                // Validate
                if (value == null)
                {
                    throw new ArgumentException("Departure Date cannot be blank");
                }
                _departureDate = value;
            }
        }
```

```
public int chaletID
{
    get
    {
        return _chaletID;
    }
    set
    {
        // Validate
        if(value < 0 || value > 10)
        {
            throw new ArgumentException("Chalet ID must be between 1 and 10");
        }
        _chaletID = value;
    }
}

public bool eveningMeal
{
    get
    {
        return _eveningMeal;
    }
    set
    {
        // Does not need validating as value comes from checkbox
        _eveningMeal = value;
    }
}

public bool breakfast
{
    get
    {
        return _breakfast;
    }
    set
    {
        // Does not need validating as value comes from checkbox
        _breakfast = value;
    }
}

public int custID
{
    get
    {
        return _customerID;
    }
    set
    {
        _customerID = value;
    }
}

public List<long> guestPassList
{
    get
    {
        return _guests;
    }
    set
    {
        if(value == null)
        {
            throw new ArgumentException("Booking must contain at least 1 guest");
        }
        _guests = value;
    }
}

public bool carHire
{
    get
    {
        return _carHire;
```

```
            }
            set
            {
                // Does not need validating as value comes from checkbox
                _carHire = value;
            }
        }
    }
}
```

Guest.cs

```
// Author: Christopher Johnson [40275286]
// Class Purpose: Guest class to store guest information
// Date Last Modified: 03/12/2017

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Business
{
    public class Guest
    {
        // Declare attributes for the guest class
        private string _guestName;
        private long _passNumber;
        private int _guestAge;

        public string guestName
        {
            get
            {
                return _guestName;
            }
            set
            {
                // Validate
                if (string.IsNullOrEmpty(value))
                {
                    throw new ArgumentException("Guest name cannot be blank!");
                }
                _guestName = value;
            }
        }

        public long passNumber
        {
            get
            {
                return _passNumber;
            }
            set
            {
                // Validate
                if(value.ToString().Length != 10)
                {
                    throw new ArgumentException("Invalid Passport Number!");
                }
                _passNumber = value;
            }
        }

        public int guestAge
        {
            get
            {
                return _guestAge;
            }
            set
            {
                // Validate
                if (value < 0 || value > 110)
                {
                    throw new ArgumentException("Age must be between 0 and 101 years!");
                }
                _guestAge = value;
            }
        }
    }
}
```

Customer.cs

```
// Author: Christopher Johnson [40275286]
// Class Purpose: Customer class to store customer information
// Date Last Modified: 10/12/2017

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

namespace Business
{
    public class Customer
    {
        // Declare the attributes for the customer class
        private string _customerName;
        private string _customerAddress;
        private int _customerRefNumber;

        public string customerName
        {
            get
            {
                return _customerName;
            }
            set
            {
                // Validate
                if(string.IsNullOrEmpty(value))
                {
                    throw new ArgumentException("Name must not be blank!");
                }
                _customerName = value;
            }
        }

        public string customerAddress
        {
            get
            {
                return _customerAddress;
            }
            set
            {
                // Validate
                if (string.IsNullOrEmpty(value))
                {
                    throw new ArgumentException("Address must not be blank!");
                }
                _customerAddress = value;
            }
        }

        public int customerRefNumber
        {
            get
            {
                return _customerRefNumber;
            }
            set
            {
                // Does not need to be validated because it's auto generated
                _customerRefNumber = value;
            }
        }
    }
}
```

CarHire.cs

```
// Author: Christopher Johnson [40275286]
// Class Purpose: Car Hire class to store car hire information
// Date Last Modified: 10/12/2017

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Business
{
    public class CarHire
    {
        // Declare the attributes for the car hire class
        private DateTime _hireStart;
        private DateTime _hireEnd;
        private long _driverPass;
        private int _carHireRef;

        public DateTime hireStart
        {
            get
            {
                return _hireStart;
            }
            set
            {
                // Validate
                if (value == null)
                {
                    throw new ArgumentException("Arrival Date cannot be blank");
                }
                _hireStart = value;
            }
        }

        public DateTime hireEnd
        {
            get
            {
                return _hireEnd;
            }
            set
            {
                // Validate
                if (value == null)
                {
                    throw new ArgumentException("Arrival Date cannot be blank");
                }
                _hireEnd = value;
            }
        }

        public long driverPass
        {
            get
            {
                return _driverPass;
            }
            set
            {
                _driverPass = value;
            }
        }

        public int carHireRef
        {
            get
            {
                return _carHireRef;
            }
            set
            {
                // Is autogenerated no need for validation
                _carHireRef = value;
            }
        }
    }
}
```

BookingList.cs

```
// Author: Christopher Johnson [40275286]
// Class Purpose: Booking List to store booking objects
// Date Last Modified: 10/12/2017

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Business
{
    public class BookingList
    {
        private List<Booking> _list = new List<Booking>();

        public void addBooking(Booking newBooking)
        {
            _list.Add(newBooking);
        }

        public Booking findBooking(int bookingID)
        {
            foreach (Booking p in _list)
            {
                if (bookingID == p.bookingID)
                {
                    return p;
                }
            }

            return null;

        }

        public void deleteBooking(int bookingID)
        {
            Booking p = this.findBooking(bookingID);
            if (p != null)
            {
                _list.Remove(p);
            }

        }

        public List<int> bookingIDs
        {
            get
            {
                List<int> res = new List<int>();
                foreach (Booking p in _list)
                    res.Add(p.bookingID);
                return res;
            }

        }
    }
}
```

CarHireList.cs

```
// Author: Christopher Johnson [40275286]
// Class Purpose: Car Hire list to store car hire objects
// Date Last Modified: 10/12/2017

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Business
{
    public class CarHireList
    {
        private List<CarHire> _list = new List<CarHire>();

        public void addCarHire(CarHire newCarHire)
        {
            _list.Add(newCarHire);
        }

        public CarHire find(int carHireRef)
        {
            foreach (CarHire p in _list)
            {
                if (carHireRef == p.carHireRef)
                {
                    return p;
                }
            }

            return null;

        }

        public void deleteCarHire(int carHireRef)
        {
            CarHire p = this.find(carHireRef);
            if (p != null)
            {
                _list.Remove(p);
            }

        }

        public List<int> carHireRefs
        {
            get
            {
                List<int> res = new List<int>();
                foreach (CarHire p in _list)
                    res.Add(p.carHireRef);
                return res;
            }
        }
    }
}
```

CustomerList.cs

```
// Author: Christopher Johnson [40275286]
// Class Purpose: Customer list to store customer objects
// Date Last Modified: 10/12/2017

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Business
{
    public class CustomerList
    {
        private List<Customer> _list = new List<Customer>();

        public void addCustomer(Customer newCustomer)
        {
            _list.Add(newCustomer);
        }

        public Customer find(int customerRefNumber)
        {
            foreach (Customer p in _list)
            {
                if (customerRefNumber == p.customerRefNumber)
                {
                    return p;
                }
            }

            return null;

        }

        public void deleteCustomer(int customerRefNumber)
        {
            Customer p = this.find(customerRefNumber);
            if (p != null)
            {
                _list.Remove(p);
            }

        }

        public List<int> refNumbers
        {
            get
            {
                List<int> res = new List<int>();
                foreach (Customer p in _list)
                    res.Add(p.customerRefNumber);
                return res;
            }

        }
    }
}
```

GuestList.cs

```
// Author: Christopher Johnson [40275286]
// Class Purpose: Guest list to store guest objects
// Date Last Modified: 10/12/2017

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Business
{
    public class GuestList
    {
        private List<Guest> _list = new List<Guest>();

        public void addGuest(Guest newGuest)
        {
            _list.Add(newGuest);
        }

        public Guest find(int passNumber)
        {
            foreach (Guest p in _list)
            {
                if (passNumber == p.passNumber)
                {
                    return p;
                }
            }

            return null;

        }

        public void deleteGuest(int passNumber)
        {
            Guest p = this.find(passNumber);
            if (p != null)
            {
                _list.Remove(p);
            }

        }

        public List<long> passNumbers
        {
            get
            {
                List<long> res = new List<long>();
                foreach (Guest p in _list)
                    res.Add(p.passNumber);
                return res;
            }

        }
    }
}
```