Holyday Booking System

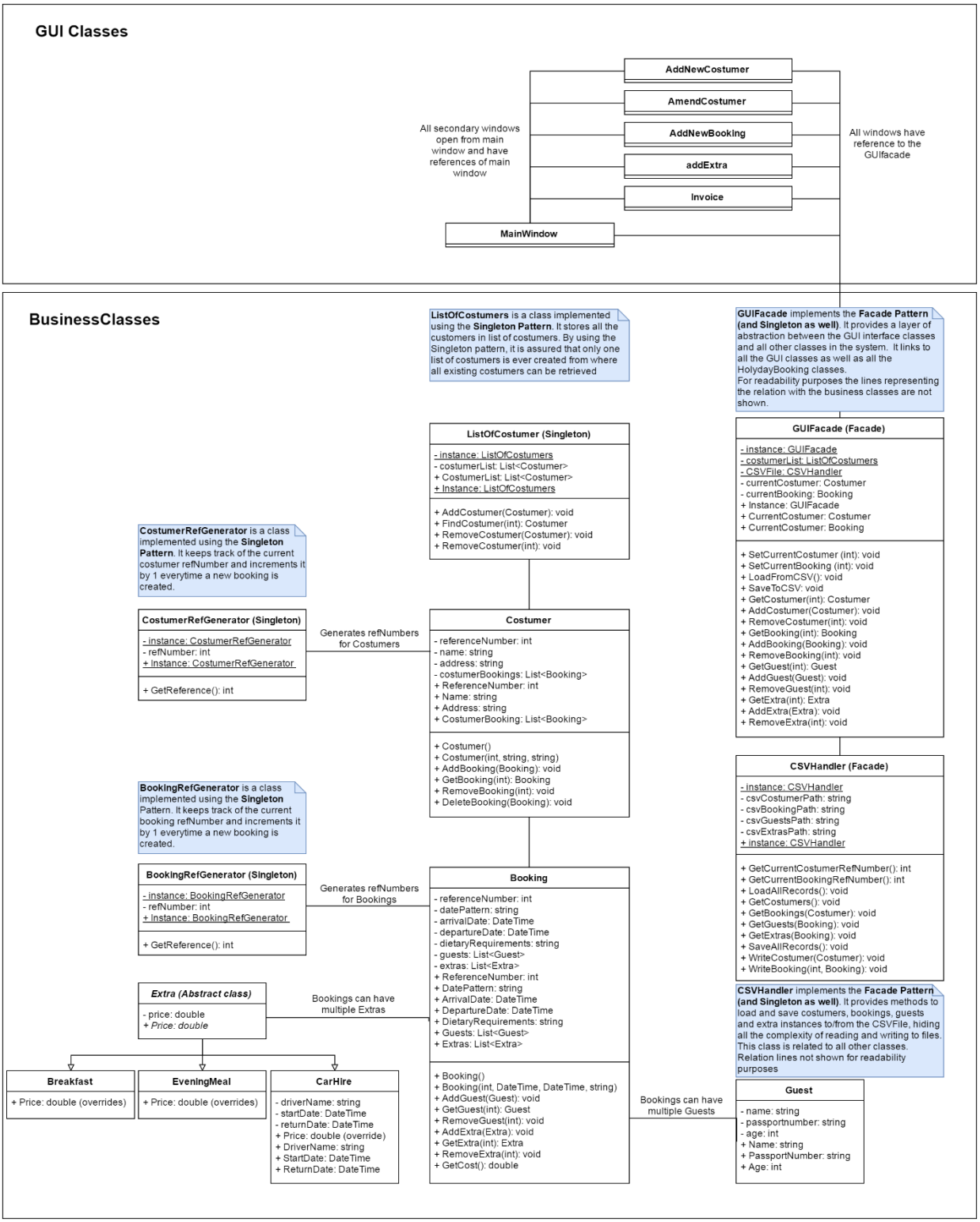
Software Development 2 – coursework

Matric number: 40218056

Name: Pedro Mendes

Date: 10-12-2016

UML Class Diagram



Contents

Business Classes

ListOfCustomers (Singleton)

```
//Author: Pedro Mendes     MatricNum:40218056
//Description: Class used to store a unique list of costumer (containing a list with
references to all the existing customers).
//Date last modified: 2016-11-26
//Class uses the Singleton Design Pattern

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

namespace HollydayBooking
{
    // Singleton Class ListofCostumers represent a list of all the costumers of the
Hollyday booking System
    public class ListOfCostumers
    {
        private static ListOfCostumers instance; //create a static instance of the class
        private List<Costumer> costumerList = new List<Costumer>(); //create a list of
customers

        // Returns always the same instance of the class. Creates a new instance if one has
not yet been created
        public static ListOfCostumers Instance()
        {
            if (instance == null)
            {
                instance = new ListOfCostumers();
            }
            return instance;
        }

        //Property used to get the entire list of customers
        public List<Costumer> CostumerList
        {
            get
            {
                return costumerList;
            }
        }

        //Method adds a costumer object to the list of costumers
        public void AddCostumer(Costumer costumer)
        {
            this.costumerList.Add(costumer);
        }

        //Return the costumer given its reference number (return null if costumer with the
given reference number does not exist)
        public Costumer FindCostumer(int costumerReference)
        {
            foreach(Costumer costumer in costumerList)
            {
                if (costumer.ReferenceNumber == costumerReference)
                {
                    return costumer;
                }
```

```
            }
            return null;
        }

        // Remove a costumer from a list of costumer given the costumer object
        public void RemoveCostumer(Costumer costumer)
        {
            costumerList.Remove(costumer);
        }

        // Remove a costumer from a list of costumer given its reference number
        public void RemoveCostumer(int refNumber)
        {
            foreach (Costumer aCostumer in this.costumerList)
            {
                if (aCostumer.ReferenceNumber == refNumber)
                {
                    costumerList.Remove(aCostumer);
                    break;
                }
            }
        }
    }
}
```

Costumer

```
//Author: Pedro Mendes     MatricNum:40218056
//Description: Class used to represent a Costumer. Constains Costumer properties and its
set and get methods (with validation checking) and some other useful methods
//Date last modified: 2016-12-07
//

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

namespace HollydayBooking
{
    // Class represents a Costumer
    public class Costumer
    {
        private int referenceNumber;//holds the reference number of the custumer which
uniquely identifies it
        private string name; //Holds the custumer name
        private string address;//holds the customer address
        private List<Booking> costumerBookings; //hold a list of references to all the
bookings of the customer

        //Constructer used to create new Customers
        public Costumer()
        {
            CostumerRefGenerator costRefGen = CostumerRefGenerator.Instance();
            referenceNumber = costRefGen.GetReference(); //use the CostumerRefGenerator
class to generate the right reference number for the booking
            costumerBookings = new List<Booking>(); //Creates an empty list of bookings
        }

        //Constructor used to create pre-existing custumers (details of custumers loaded
from CSV when the application starts (including custumer reference numbers))
        public Costumer(int refNumber, string name, string address)
        {
            this.referenceNumber = refNumber;
            this.name = name;
            this.address = address;
            costumerBookings = new List<Booking>();
        }


        //ReferenceNumber can not be set after the object has been created.
        public int ReferenceNumber
        {
            get
            {
                return referenceNumber;
            }
        }

        //Get set methods for the Name property. Throw exception is the empty string is
assigned to name.
        public string Name
        {
            get
            {
                return name;
            }
            set
```

```
            {
                if (value == "")
                {
                    throw new ArgumentException("Customer name can not be left blank");
                }
                name = value;
            }
        }

        //Get set methods for the Address property. Throw exception is the empty string is
assigned to Address.
        public string Address
        {
            get
            {
                return address;
            }
            set
            {
                if (value == "")
                {
                    throw new ArgumentException("Customer address can not be left blank");
                }
                address = value;
            }
        }

        //Get set methods for the CostumerBookings property
        public List<Booking> CostumerBookings
        {
            get
            {
                return costumerBookings;
            }
            set
            {
                costumerBookings = value;
            }
        }

        //Adds a new booking to costumer given a Booking object
        public void AddBooking(Booking booking)
        {
            costumerBookings.Add(booking);
        }

        // Returns a booking from this custumer's booking list given a Booking reference
        public Booking GetBooking(int refNumber)
        {
            foreach (Booking aBooking in costumerBookings)
            {
                if (aBooking.ReferenceNumber == refNumber)
                {
                    return aBooking;
                }
            }
            return null; //Returns null instead if custumer does not exist
        }

        // Removes a booking from this custumer's booking list given a Booking reference
(if booking exist)
        public void RemoveBooking(int refNum)
        {
            foreach (Booking aBooking in this.CostumerBookings)
            {
                if (aBooking.ReferenceNumber == refNum)
```

```
                {
                    aBooking.Guests.Clear();
                    aBooking.Extras.Clear();
                    this.CostumerBookings.Remove(aBooking);
                    break;//Break out of loop when booking is deleted (no point in
continuing iterating through the list)
                }
            }
        }

        // Removes a booking from the customer's list of bookings given a booking
        public void DeleteBooking(Booking aBooking)
        {
            aBooking.Guests.Clear(); //Clear the Booking list of guests
            aBooking.Extras.Clear(); //Clear the Booking list of extras
            this.CostumerBookings.Remove(aBooking); //Remove the booking from the costumer
list of bookings
        }
    }
}
```

Booking

```
//Author: Pedro Mendes     MatricNum:40218056
//Description: Class used to represent a Booking. Contains booking properties (with
validation checking) and some useful methods
//Date last modified: 2016-12-07
//

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

namespace HollydayBooking
{
    public class Booking
    {
        private int referenceNumber; //holds the booking reference number that uniquely
identifies the booking
        private string datePattern; //holds the string pattern used to convert DateTime
fields to string
        private DateTime arrivalDate; //holds the booking arrival date
        private DateTime departureDate; //holds the booking departuredate
        private string dietaryRequirements; //holds possible nutritional requirements
associated with the booking
        private List<Guest> guests = new List<Guest>(); //Holds references to all the
guests associated with the booking
        private List<Extra> extras = new List<Extra>(); //Holds references to all the
extras associated with the booking

        //contructer used to create new bookings
        public Booking()
        {
            //Booking reference number automatically generated using the
BookingRefGenerator class
            BookingRefGenerator bookingRefGen = BookingRefGenerator.Instance();
            referenceNumber = bookingRefGen.GetReference();
            datePattern = "yyyy-MM-dd"; //pattern for dates used in DateTime fields
        }

        // Contructer used to create (exesting) Bookings loaded from the CSVFile
        public Booking(int refNumber, DateTime arrivalDate, DateTime departureDate, string
dietaryRequirements)
        {
            this.referenceNumber = refNumber;
            this.arrivalDate = arrivalDate;
            this.departureDate = departureDate;
            this.dietaryRequirements = dietaryRequirements;
            datePattern = "yyyy-MM-dd";
        }

        //ReferenceNumber property containing the get method. Does not contain a set method
since the referenceNumber cannot be changed after the object creation
        public int ReferenceNumber
        {
            get
            {
                return referenceNumber;
            }
        }

        //DatePattern property the get method. Value is assigned when object is created.
(no need for a set method)
```

```
        public string DatePattern
        {
            get
            {
                return datePattern;
            }
        }

        //get and set methods for the ArrivalDate property
        public DateTime ArrivalDate
        {
            get
            {
                return arrivalDate;
            }
            set
            {
                arrivalDate = value;
            }
        }

        //get and set methods for the DepartureDate property
        public DateTime DepartureDate
        {
            get
            {
                return departureDate;
            }
            set
            {
                //Departure date must be after the arrival date
                if (value < arrivalDate)
                {
                    throw new ArgumentException("Error: departure date cannot be before the
arrival date");
                }
                departureDate = value;
            }
        }

        //get and set methods for the DietaryRequirements property
        public string DietaryRequirements
        {
            get
            {
                return dietaryRequirements;
            }
            set
            {
                dietaryRequirements = value;
            }
        }

        //get and set methods for the Guests property
        public List<Guest> Guests
        {
            get
            {
                return guests;
            }
            set
            {
                guests = value;
            }
        }
```

```
        //get and set methods for the Extras property
        public List<Extra> Extras
        {
            get
            {
                return extras;
            }
            set
            {
                extras = value;
            }
        }

        // Method to add a guest to the list of guests accepting a guest object as only
argument
        public void AddGuest(Guest guest)
        {
            //Add guest to list if there is less than 4 guests in the list
            if (guests.Count < 4)
            {
                guests.Add(guest);
            }
            //throw exception if the list is full (4 guests already)
            else
            {
                throw new ArgumentException("Error: A booking can have a maximum of 4
guests!");
            }
        }

        //Method used to get a certain guest from the list of guests given its index
        public Guest GetGuest(int index)
        {
            // return the guest correspondent to the index provided
            if (index < this.Guests.Count() && index >= 0)
            {
                return this.Guests.ElementAt(index);
            }
            // return null if index provided does not exist
            else
            {
                return null;
            }
        }

        //Method to remove a guest from the list of guests given the guest index
        public void RemoveGuest(int index)
        {
            if (index > Guests.Count || index < 0)
            {
                throw new ArgumentException("Guest with index provided does not exist");
            }
            else
            {
                Guests.RemoveAt(index);
            }
        }

        //Adds an extra to the list of extras given an extra object
        public void AddExtra(Extra extra)
        {
            extras.Add(extra);
        }

        //Returns an extra object given its index
        public Extra GetExtra(int index)
```

```
        {
            // return the extra correspondent to the index provided
            if (index < this.Extras.Count())
            {
                return this.Extras.ElementAt(index);
            }
            //return null if index does not exist
            else
            {
                return null;
            }
        }

        //Removes an extra from the list given its index
        public void RemoveExtra(int index)
        {
            if (index < 0 || index > Extras.Count)
            {
                throw new ArgumentException("Extra with index provided does not exist");
            }
            else
            {
                Extras.RemoveAt(index);
            }

        }

        //Gets the total cost for a booking (adding the total stay charge + extras)
        public double GetCost()
        {
            double totalCost = 0;
            // Get the total amount (for the stay) for each of the guests guests
            foreach (Guest guest in guests)
            {
                //If guest is child
                if (guest.Age <= 18)
                {
                    // Then, cost for the guest is £30.00 (child rate) * number of nights
                    totalCost += (30.00 * (this.departureDate -
this.arrivalDate).TotalDays); //add value to totalCost
                }
                else
                {
                    // otherwise, cost for the guest is £50.00 (adult rate) * number of
nights
                    totalCost += (50.00 * (this.departureDate -
this.arrivalDate).TotalDays); //add value to totalCost
                }
            }
            // Get the total amount (for each extra) for all extras
            foreach (Extra extra in extras)
            {
                // If extra is of type brekfast
                if (extra is Breakfast)
                {
                    //Add to totalCost the price of a Brekfast * the number of nights *
number of Guests
                    totalCost += extra.Price * (this.DepartureDate - this.ArrivalDate).Days
* guests.Count();
                }
                else if (extra is EveningMeal)
                {
                    //Add to totalCost the price of an Evening Meal * the number of nights
* number of Guests
                    totalCost += extra.Price * (this.DepartureDate - this.ArrivalDate).Days
* guests.Count();
```

```
                }
                else if (extra is CarHire)
                {
                    CarHire aCarHire = (CarHire)extra; // Cast from extra to CarHire object
in order to access CarHire specific properties
                    //Add to totalCost the price of Hiring a car per night * the number of
days Hired
                    totalCost += aCarHire.Price * (aCarHire.ReturnDate -
aCarHire.StartDate).TotalDays;
                }
            }
            return totalCost; // Returns the totalCost of the Booking
        }
    }
}
```

Guest

```
//Author: Pedro Mendes     MatricNum:40218056
//Description: Class used to represent a Guest. Constains Guest properties (with validation
checking) and some usefull methods
//Date last modified: 2016-12-07

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

namespace HollydayBooking
{
    public class Guest
    {
        private string name; //Holds the name guest name
        private string passportNumber; //holds the guest passport number
        private int age; //holds the guest age


        //Get/set methods for the Name property
        public string Name
        {
            get
            {
                return name;
            }
            set
            {
                //Name can not be set to the empty string (throws excepcion)
                if (value == "")
                {
                    throw new ArgumentException("Error: Please enter a valid Guest name");
                }
                name = value;
            }
        }

        //Get/set methods for the PassportNumber property
        public string PassportNumber
        {
            get
            {
                return passportNumber;
            }
            set
            {
                //Passport number can not be set to the empty string and cannot be longer
than 10 characters (throws excepcion)
                if (value.Length > 10 || value == "") // check string is 0-10 char long
                {
                    throw new ArgumentException("Error: Please enter a valid passport
number (maximum 10 characters)!");
                }
                passportNumber = value;
            }
        }

        //Get/set methods for the Age property
        public int Age
        {
            get
```

```
            {
                return age;
            }
            set
            {
                //Age must be an integer between 0 and 101 (throws excepcion otherwise)
                if (value < 0 || value > 101)
                {
                    throw new ArgumentException("Error: Plase enter a valid age (between 0
and 101)!");
                }
                age = value;
            }
        }
    }
}
```

Extra

```
//Author: Pedro Mendes     MatricNum:40218056
//Description: Class used to represent a Booking extra. Different types of extras inherit
from this class
//Date last modified: 2016-12-07
//

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

namespace HollydayBooking
{
    public abstract class Extra //Abstract class extra. No instance of this class will ever
be created. Represents a purely conceptual object that contains nothing else than a price
    {
        private double price;

        public abstract double Price { get; }
    }
}
```

Breakfast

```
//Author: Pedro Mendes     MatricNum:40218056
//Description: Class used to represent a Breakfast. Inherits from extra.
//Date last modified: 2016-11-18
//

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

namespace HollydayBooking
{
    public class Breakfast : Extra // Class inherits from Extra
    {

        // Returns the price of a Brekfast (double)
        public override double Price
        {
            get
            {
                return 5.00;
            }
        }
    }
}
```

CarHire

```
//Author: Pedro Mendes     MatricNum:40218056
//Description: Class used to represent a CarHire. Inherits from extra.
//Date last modified: 2016-11-18
//

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

namespace HollydayBooking
{
    public class CarHire : Extra //inherit from extra
    {
        private string driverName; //holds the name of the driver
        private DateTime startDate; //holds the hiring startDate
        private DateTime returnDate; //holds the hiring endtDate

        //Get and set methods for the DriverName property. Driver Name can not be left
blank, throw exception if an empty string is assigned to the driver name
        public string DriverName
        {
            get
            {
                return driverName;
            }
            set
            {
                if (value == "")
                {
                    throw new ArgumentException("Error: Please insert a valid Driver
name!");
                }
                driverName = value;
            }
        }

        //Get and set methods for the StartDate property.
        public DateTime StartDate
        {
            get
            {
                return startDate;
            }
            set
            {
                startDate = value;
            }
        }

        //Get and set methods for the ReturnDate property. Except is thrown when ReturnDate
is a Date before the StartDate (it would not make sence).
        public DateTime ReturnDate
        {
            get
            {
                return returnDate;
            }
            set
            {
                if (value < startDate)
```

```
                {
                    throw new ArgumentException("Error: Return date cannot be before pick-
up date!");
                }
                returnDate = value;
            }
        }

        //Returns the rate of a carhire/day
        public override double Price
        {
            get
            {
                return 50.00;
            }
        }
    }
}
```

Evening Meal

```
//Author: Pedro Mendes     MatricNum:40218056
//Description: Class used to represent an Evening Meal. Inherits from extra.
//Date last modified: 2016-11-18
//

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

namespace HollydayBooking
{
    public class EveningMeal : Extra // Class inherits from Extra
    {
        // Returns the price of an Evening Meal (double)
        public override double Price
        {
            get
            {
                return 15.00;
            }
        }
    }
}
```

CostumerRefGenerator (Singleton)

```
//Author: Pedro Mendes     MatricNum:40218056
//Description: Class used to autogenerate distinct (auto-incremented) reference numbers for
new Customers
//Date last modified: 2016-11-26
//Class uses the Singleton Design Pattern

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

namespace HollydayBooking
{
    public class CostumerRefGenerator
    {

        private static CostumerRefGenerator instance; //Declaration of a private static
instance of the class
        private int refNumber; //holds the current CustomerRefNumber

        //Method always return the same instance of the class
        public static CostumerRefGenerator Instance()
        {
            //Create new instance if it hasnt been created previously (if it is null)
            if (instance == null)
            {
                instance = new CostumerRefGenerator(); //Create new instance
                CSVHandler CSVFile = CSVHandler.Instance(); //Get a reference to the
CSVFacade
                instance.refNumber = CSVFile.GetCurrentCostumerRefNumber(); //Get last
reference number given to a customer from the records in the CSV and assign the value to
the reference number of this class
            }
            return instance; //Return the instance of the class (always the same one)
        }


        public int GetReference()
        {
            refNumber++; //Increment reference number by one
            return refNumber; //Return the new reference number
        }
    }
}
```

BookingRefGenerator (Singleton)

```
//Author: Pedro Mendes     MatricNum:40218056
//Description: Class used to autogenerate distinct (auto-incremented) reference numbers for
new bookings
//Date last modified: 2016-11-26
//Class uses the Singleton Design Pattern

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

namespace HollydayBooking
{
    public class BookingRefGenerator
    {
        private static BookingRefGenerator instance; //Declaration of a private static
instance of the class
        private int refNumber; //holds the current bookingRefNumber


        //Method always return the same instance of the class
        public static BookingRefGenerator Instance()
        {
            //Create new instance if it hasnt been created previously (if it is null)
            if (instance == null)
            {
                instance = new BookingRefGenerator(); //Create new instance
                CSVHandler CSVFile = CSVHandler.Instance(); //Get a reference to the
CSVFacade
                instance.refNumber = CSVFile.GetCurrentBookingRefNumber(); //Get last
reference number given to a booking from the records in the CSV and assign the value to the
reference number of this class
            }
            return instance; //Return the instance of the class (always the same one)
        }

       //Method used to get the reference number for the creation of a new booking instance
        public int GetReference()
        {
            refNumber++; //Increment reference number by one
            return refNumber; //Return the new reference number
        }
    }
}
```

CSVHandler (Façade & Singleton)

```
//Author: Pedro Mendes     MatricNum:40218056
//Description: Classed used to load and save records to/from the CSV file
//Date last modified: 2016-12-03
//Class uses the Singleton Design Pattern and the facade pattern (Works as a facade
containing all the methods to access the CSV file, hiding the complexity of this operation)

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


namespace HollydayBooking
{
    public class CSVHandler
    {

        private static CSVHandler instance; //Creates static instance of the class (typical
of the cingleton disign pattern)
        private string csvCostumerPath = @"H:\Software development
2\CourseWork2\HollydayBooking\costumer.csv"; //holds the path to the costumer.csv file
        private string csvBookingPath = @"H:\Software development
2\CourseWork2\HollydayBooking\booking.csv"; //holds the path to the booking.csv file
        private string csvGuestsPath = @"H:\Software development
2\CourseWork2\HollydayBooking\guests.csv"; //holds the path to the guests.csv file
        private string csvExtrasPath = @"H:\Software development
2\CourseWork2\HollydayBooking\extras.csv"; //holds the path to the extras.csv file


        // Method always returns the same instance of the class (no more than one instance
can be created)
        public static CSVHandler Instance()
        {
            //Create new instance if one does not exist, otherwise return the existing
instance
            if (instance == null)
            {
                instance = new CSVHandler();

            }
            return instance;
        }


        protected CSVHandler()
        {

        }

        // Method returns the last (max) reference number for a customer from the
costumer.csv file
        public int GetCurrentCostumerRefNumber()
        {
            int ccrf = 0; //local int variable (used as a paceholder for the costumer
reference number)
```

```
            // Try to read from costumer.csv file (will fail if file does not exist)
            try
            {
                // Create a streamReader object to access the file
                StreamReader reader = new StreamReader(File.OpenRead(csvCostumerPath));
                //Read file line by line until endOfFile
                while (!reader.EndOfStream)
                {
                    var line = reader.ReadLine(); //reads the entire line
                    var values = line.Split(','); // splits the line on the commas in
creates a list with the values
                    if (Int32.Parse(values[0]) >= ccrf) //Check if the reference number
(values[0]) is greater than the previously stored reference number
                    {
                        ccrf = Int32.Parse(values[0]); //Assign new reference number if its
greater then the previously stored value
                    }
                }
                reader.Dispose(); //Dispose the reader
                return ccrf; //Return (the greatest) reference number found in the file
            }
            // If reading from file fails, return 0;
            catch
            {
                return ccrf;
            }
        }


        // Same as the previous method but for booking
        public int GetCurrentBookingRefNumber()
        {
            int cbrf = 0;
            try
            {
                StreamReader reader = new StreamReader(File.OpenRead(csvBookingPath));
                while (!reader.EndOfStream)
                {
                    var line = reader.ReadLine();
                    var values = line.Split(',');
                    if (Int32.Parse(values[1]) >= cbrf)
                    {
                        cbrf = Int32.Parse(values[1]);
                    }
                }
                reader.Dispose();
                return cbrf;
            }
            catch
            {
                return cbrf;
            }
        }

        //Method Loads all the records from the CSV files into appropriate objects
        public void LoadAllRecords()
        {
            ListOfCostumers costumerList = ListOfCostumers.Instance(); //Get an instance of
the ListOfCustomer (Singleton) class
            if (File.Exists(csvCostumerPath)) //check if customer.csv file exists
            {
```

```
                GetCostumers(); //cal the GetCostumers() method
            }
            if (File.Exists(csvBookingPath)) //check if booking.csv file exists
            {
                foreach (Costumer aCostumer in costumerList.CostumerList) //Iterates
through all the costumers in the list
                {
                    GetBookings(aCostumer); //Call the GetBooking method (Populates the
booking list (of each customer) with all its bookings)
                }
                foreach (Costumer aCostumer in costumerList.CostumerList) //Iterates
through all the costumers in the list
                {
                    foreach (Booking aBooking in aCostumer.CostumerBookings) //Iterates
through all the bookings for a specific customer
                    {
                        if (File.Exists(csvGuestsPath)) //check if guests.csv file exists
                        {
                            GetGuests(aBooking); //Call the GetGuests method
                        }
                        if (File.Exists(csvExtrasPath)) //check if extras.csv file exists
                        {
                            GetExtras(aBooking); //Call the GetExtras method
                        }
                    }
                }
            }
        }

        // Loads all the customers from the CSV into a list of customers (ListOfCostumer
Singleton Class)
        public void GetCostumers()
        {
            // Create a streamReader object to access the file
            StreamReader reader = new StreamReader(File.OpenRead(csvCostumerPath));
            ListOfCostumers costumerList = ListOfCostumers.Instance();//Get an instance of
the ListOfCustomer (Singleton) class
            // Create a streamReader object to access the file
            while (!reader.EndOfStream)
            {
                var line = reader.ReadLine();
                var values = line.Split(',');
                Costumer aCostumer = new Costumer(Int32.Parse(values[0]), values[1],
values[2]); //Create a new customer and sets its properties to the values on the csv
                costumerList.AddCostumer(aCostumer); //adds the customer to the custumer
list (in the ListOfCustumers in the singleton class)
            }
            reader.Dispose();
        }

        // Reads from booking.csv file to populate the booking list (of each customer) with
all its bookings
        public void GetBookings(Costumer aCostumer)
        {
            StreamReader reader = new StreamReader(File.OpenRead(csvBookingPath));
            while (!reader.EndOfStream)
            {
                var line = reader.ReadLine();
                var values = line.Split(',');
                if (Int32.Parse(values[0]) == aCostumer.ReferenceNumber) //if the booking
belongs to the custumer (aCostumer)
```

```
                {
                    Booking aBooking = new Booking(Int32.Parse(values[1]),
DateTime.Parse(values[2]), DateTime.Parse(values[3]), values[4]); //Create a new booking
object an set its values from the CSV record
                    aCostumer.AddBooking(aBooking); //Add new booking to the Custumer
listofbookings
                }
            }
            reader.Dispose();
        }

        // Reads from guests.csv file to populate the a booking list of guests  (creates
Guest object and adds it to the list)
        public void GetGuests(Booking aBooking)
        {
            StreamReader reader = new StreamReader(File.OpenRead(csvGuestsPath));
            while (!reader.EndOfStream)
            {
                var line = reader.ReadLine();
                var values = line.Split(',');
                if (Int32.Parse(values[0]) == aBooking.ReferenceNumber)
                {
                    Guest aGuest = new Guest();
                    aGuest.Name = values[1];
                    aGuest.PassportNumber = values[2];
                    aGuest.Age = Int32.Parse(values[3]);
                    aBooking.Guests.Add(aGuest);
                }
            }
            reader.Dispose();
        }

        // Reads from extras.csv file to populate the a booking list of extras  (creates
extra objects (Breakfasts, MealHire, CarHire) and adds it to the list)
        public void GetExtras(Booking aBooking)
        {
            StreamReader reader = new StreamReader(File.OpenRead(csvExtrasPath));
            while (!reader.EndOfStream)
            {
                var line = reader.ReadLine();
                var values = line.Split(',');
                if (Int32.Parse(values[0]) == aBooking.ReferenceNumber)
                {
                    if (values[1] == "HollydayBooking.Breakfast") //Check if extra is a
breakfast
                    {
                        Breakfast aBreakfast = new Breakfast();
                        aBooking.Extras.Add(aBreakfast);
                    }
                    else if (values[1] == "HollydayBooking.EveningMeal") //Check if extra
is an EveningMeal
                    {
                        EveningMeal anEveningMeal = new EveningMeal();
                        aBooking.Extras.Add(anEveningMeal);
                    }
                    else if (values[1] == "HollydayBooking.CarHire") //Check if extra is
CarHire
                    {
                        CarHire aCarHire = new CarHire();
                        aCarHire.DriverName = values[2];
                        aCarHire.StartDate = DateTime.Parse(values[3]);
```

```
                    aCarHire.ReturnDate = DateTime.Parse(values[4]);
                    aBooking.Extras.Add(aCarHire);
                }
            }
        }
        reader.Dispose();
    }

    //Method Saves all the objects (customers, booking, guests and extras) in the
system to the csv files
    public void SaveAllRecords()
    {
        //Delete all the csv files (to write back from scratch)
        File.Delete(csvCostumerPath);
        File.Delete(csvBookingPath);
        File.Delete(csvGuestsPath);
        File.Delete(csvExtrasPath);
        ListOfCostumers costumerList = ListOfCostumers.Instance();
        foreach (Costumer aCostumer in costumerList.CostumerList)
        {
            WriteCostumer(aCostumer);
            foreach (Booking aBooking in aCostumer.CostumerBookings)
            {
                WriteBooking(aCostumer.ReferenceNumber, aBooking);
            }
        }
    }

    //Method writes (appends) a costumer to the costumer.csv file
    public void WriteCostumer (Costumer aCostumer)
    {
        string line = aCostumer.ReferenceNumber.ToString() + ","
               + aCostumer.Name.ToString() + ","
               + aCostumer.Address.ToString();
        string filepath = csvCostumerPath;
        StringBuilder sb = new StringBuilder();
        sb.AppendLine(line);
        System.IO.File.AppendAllText(filepath, sb.ToString());
    }

    public void WriteBooking(int costumerRef, Booking aBooking)
    {
        StringBuilder sb = new StringBuilder();

        // Write booking details to booking.csv
        string line = costumerRef.ToString() + ","
                      + aBooking.ReferenceNumber.ToString() + ","
                      + aBooking.ArrivalDate.ToString(aBooking.DatePattern) + ","
                      + aBooking.DepartureDate.ToString(aBooking.DatePattern) + ","
                      + aBooking.DietaryRequirements.ToString();
        string filepath = csvBookingPath;
        sb.AppendLine(line);
        System.IO.File.AppendAllText(filepath, sb.ToString());
        sb.Clear(); //Clears the buffer's current content

        //Write Guest details to guests.csv
        filepath = csvGuestsPath; //write guest details to guests.csv
        foreach (Guest guest in aBooking.Guests)
        {
             line = aBooking.ReferenceNumber.ToString() + ","
                    + guest.Name.ToString() + ","
```

```
                    + guest.PassportNumber.ToString() + ","
                    + guest.Age.ToString();
                sb.AppendLine(line);
            }
            System.IO.File.AppendAllText(filepath, sb.ToString());
            sb.Clear();

            //Write Extra details to extras.csv
            filepath = csvExtrasPath; //write extra details to extra.csv
            foreach (Extra extra in aBooking.Extras)
            {
                if (extra.GetType().ToString() == "HollydayBooking.Breakfast" ||
extra.GetType().ToString() == "HollydayBooking.EveningMeal")
                {
                    line = aBooking.ReferenceNumber.ToString() + "," +
extra.GetType().ToString();
                    sb.AppendLine(line);
                }
                else
                {
                    CarHire aCarHire = new CarHire();
                    aCarHire = (CarHire)extra;
                    line = aBooking.ReferenceNumber.ToString() + ","
                      + extra.GetType().ToString() + ","
                      + aCarHire.DriverName.ToString() + ","
                      + aCarHire.StartDate.ToString(aBooking.DatePattern) + ","
                      + aCarHire.ReturnDate.ToString(aBooking.DatePattern);
                    sb.AppendLine(line);
                }
            }
            System.IO.File.AppendAllText(filepath, sb.ToString());
        }
    }
}
```

GUIFacade (Façade & Singleton)

```
//Author: Pedro Mendes     MatricNum:40218056
//Description: Class used as a facade between the GUI classes and the business classes. It
contains methods that provide all the functionality of the business classes
//Date last modified: 2016-12-03
//Class uses the Singleton Design Pattern and the facade pattern (Works as an abstraction
layer between the GUI classes and the business classes)

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

namespace HollydayBooking
{
    public class GUIFacade
    {
        private static GUIFacade instance; //static instance of GUIFacade
        private static ListOfCostumers costumerList; //instance of the ListOfCustomer class
        private static CSVHandler CSVFile; //instance of the CSVHandler class
        private Costumer currentCostumer; //holds the current custumer (the one that has
been selected by the user)
        private Booking currentBooking; //holds the current booking (the one that has been
selected by the user)

        // Property used to get the current custumer
        public Costumer CurrentCostumer
        {
            get
            {
                return currentCostumer;
            }
        }

        // Property used to get the current booking
        public Booking CurrentBooking
        {
            get
            {
                return currentBooking;
            }
        }

        // Always return the same instance of the class. Create a new instance if one has
not already been created
        public static GUIFacade Instance
        {
            get
            {
                if (instance == null)
                {
                    instance = new GUIFacade();
                    costumerList = ListOfCostumers.Instance();
                    CSVFile = CSVHandler.Instance();
                }
                return instance;
            }
        }

        // Set the current custumer given its reference number (when selected by the user)
        public void SetCurrentCostumer(int refNumber)
        {
```

```
            currentCostumer = costumerList.FindCostumer(refNumber);
        }

        // Set the current booking given its reference number (when selected by the user)
        public void SetCurrentBooking(int refNumber)
        {
            currentBooking = currentCostumer.GetBooking(refNumber);
        }

        //Load everything from the csv files into objects
        public void LoadFromCSV()
        {
            CSVFile.LoadAllRecords();
        }

        //Saves all objects in memory to the csv files
        public void SaveToCSV()
        {
            CSVFile.SaveAllRecords();
        }

        //Return a costumer given its reference number
        public Costumer GetCostumer(int refNumber)
        {
            return costumerList.FindCostumer(refNumber);
        }

        //Adds a new custumer to the custumer list given a costumer object
        public void AddCostumer(Costumer aCostumer)
        {
            costumerList.AddCostumer(aCostumer);
        }


        //Removes a costumer from the costumer list given its reference number
        public void RemoveCostumer(int refNumber)
        {
            costumerList.RemoveCostumer(refNumber);
        }

        // Returns a booking belonging to the selected custumer by passing in its reference
number
        public Booking GetBooking(int refNumber)
        {
            return currentCostumer.GetBooking(refNumber);
        }

        // Adds a booking to the selected custumer booking list given the booking object
        public void AddBooking(Booking aBooking)
        {
            currentCostumer.CostumerBookings.Add(aBooking);
        }

        // Deletes a booking from the selected customer list of bookings given the booking
reference number
        public void RemoveBooking(int refNumber)
        {
            currentCostumer.RemoveBooking(refNumber);
        }

        // Returns a guest from the selected booking list of guests given its index
        public Guest GetGuest(int index)
        {
            return currentBooking.GetGuest(index);
        }
```

```
        // Adds a new guest to the selected booking list of guests given the guest object
        public void AddGuest(Guest aGuest)
        {
            currentBooking.AddGuest(aGuest);
        }

        // Deletes a guest from he selected booking list of guests given its index
        public void RemoveGuest(int index)
        {
            currentBooking.RemoveGuest(index);
        }

        // Returns an extra from the selected booking list of extras given the extra index
        public Extra GetExtra(int index)
        {
            return currentBooking.GetExtra(index);
        }

        //Adds an extra to the selected booking list of extras given the extra object
        public void AddExtra(Extra anExtra)
        {
            currentBooking.AddExtra(anExtra);
        }

        //Removes an extra from the selected booking list of extras given its index
        public void RemoveExtra(int index)
        {
            currentBooking.RemoveExtra(index);
        }
    }
}
```

GUI Classes

AddNewCostumer

```
//Author: Pedro Mendes     MatricNum:40218056
//Description: GUI class used to get user input for custumer details in order to create new
customers.
//Date last modified: 2016-12-07
//

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace HollydayBooking
{
    /// <summary>
    /// Interaction logic for AddNewCostumer.xaml
    /// </summary>
    public partial class AddNewCostumer : Window
    {
        GUIFacade facade = GUIFacade.Instance; //Get an instance of the GUIFacade class
        public AddNewCostumer()
        {
            InitializeComponent();
        }

        //Method validates user input for custumer name and address fields in order to
create new customers
        private void btn_addNewCostumerDone_Click(object sender, RoutedEventArgs e)
        {
            //Check if name field is not empty
            if (txt_addNewCostumerName.Text == "")
            {
                MessageBox.Show("Please enter a valid customer name!");
            }
            //Check if address field is not empty
            else if (txt_addNewCostumerAddress.Text == "")
            {
                MessageBox.Show("Please enter a valid customer address!");
            }
            else
            {
                //Create new customer object
                Costumer aCostumer = new Costumer();
                // try to assign values to the new customer object
                try
                {
                    aCostumer.Name = txt_addNewCostumerName.Text; // (Try to) assign
costumer name from user input
                    aCostumer.Address = txt_addNewCostumerAddress.Text; // (Try to) assign
costumer address from user input
```

```
                    facade.AddCostumer(aCostumer); //use the facade class to add new
customer to customer list
                    facade.SaveToCSV(); //saves it to CSV file
                    facade.SetCurrentCostumer(aCostumer.ReferenceNumber); //Sets the
current custumer in the facade class to the customer just created
                    MessageBox.Show("New Costumer has been added. Costumer reference
number: " + aCostumer.ReferenceNumber);
                    this.Close();
                }
                // If assignement of name or address fails, catch and show exception
message
                catch (Exception excep)
                {
                    MessageBox.Show(excep.Message);
                }
            }
        }
    }
}
```

AmendCostumer

```
//Author: Pedro Mendes     MatricNum:40218056
//Description: GUI class used to amend custumer details of existing Customers
//Date last modified: 2016-12-07


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace HollydayBooking
{
    /// <summary>
    /// Interaction logic for AmendCostumer.xaml
    /// </summary>
    public partial class AmendCostumer : Window
    {
        GUIFacade facade = GUIFacade.Instance; //Create a reference to the GUIfacade class
        public AmendCostumer()
        {
            InitializeComponent();
        }

        // Updates the details of an existing costumer
        private void btn_amendCostumerUpdate_Click(object sender, RoutedEventArgs e)
        {
            Costumer aCostumer = facade.CurrentCostumer; //Get Current costumer from the
facade
            // Try to re-assing the values of the customer details from the user input
            try
            {
                aCostumer.Name = txt_amendCostumerName.Text; // (Try to) re-assign costumer
name from user input
                aCostumer.Address = txt_amendCostumerAddress.Text; // (Try to) re-assign
costumer address from user input
                facade.SaveToCSV(); // Save Customer Details (amended) to CSV file
                MessageBox.Show("Costumer details were successfully updated!"); //Insforms
the user the customer detaisl were successfully updated
                MainWindow mainWin = Owner as MainWindow; // Get a reference of main window
to access its properties
                mainWin.txt_costumerReferenceNumber.Text = ""; //clear textbox in main
window
                mainWin.lbl_costumerNameOutput.Content = ""; // clear textbox in main
window
                mainWin.lbl_costumerAddressOutput.Content = ""; //clear textbox in main
window
                this.Close(); //close this window
            }
            // If re-assignement of name or address fails, catch and show exception message
            catch (Exception excep)
            {
                MessageBox.Show(excep.Message);
            }
```

```
        }
    }
}
```

AddNewBooking

```
//Author: Pedro Mendes     MatricNum:40218056
//Description: GUI class used to get user input for a Booking details in order to create a
new Booking. Also allows the user
//  to add guests and extras to the same booking on its creation
//Date last modified: 2016-12-05


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace HollydayBooking
{
    /// <summary>
    /// Interaction logic for AddNewBooking.xaml
    /// </summary>
    public partial class AddNewBooking : Window
    {
        GUIFacade facade = GUIFacade.Instance; //Get a reference of the GUIFacade class to
access its methods
        private List<Guest> tempGuests = new List<Guest>(); // Create a global list of
guests to hold guests until the list is added to booking
        private List<Extra> tempExtras = new List<Extra>(); // Create a global list of
extras to hold guests until the list is added to booking
        private string dietaryRequirements = ""; //Holds Booking dietary requirements


        public AddNewBooking()
        {
            InitializeComponent();
            //populate the extraTypes comboBox on Window inicialization
            cmb_selectExtraType.Items.Add("Breakfast");
            cmb_selectExtraType.Items.Add("Evening meal");
            cmb_selectExtraType.Items.Add("Car hire");
        }


        //Method called when add guest button is clicked. Gets user input to create a new
guest and add it to the global list of guest (tempGuests)
        private void btn_addNewBookingGuests_Click(object sender, RoutedEventArgs e)
        {
            Guest aGuest = new Guest();//Create new guest instance
            // Try to assign input details to user ( succeds if the inputted values are in
a correct format)
            try
            {
                aGuest.Name = txt_name.Text; // (Try) Assign guest name from user input
```

```
                aGuest.PassportNumber = txt_passportNumber.Text; // (Try) Assign guest
address from user input
                int age;
                //Check if age is convertible to int (if it is a valid integer)
                if (Int32.TryParse(txt_age.Text, out age))
                {
                    aGuest.Age = age; //(Try) Assign guest age from user input
                    //If there is less than 4 guests in the list of guests than add guest
to the list
                    if (tempGuests.Count < 4)
                    {
                        tempGuests.Add(aGuest); //Add guest to list of guests
                        MessageBox.Show("New guest has been added to booking."); //promt
the user that the creation og guest succeded
                        cmb_guests.IsEnabled = true; //Enable guest comboBox
                        cmb_guests.Items.Add("Guest " + tempGuests.Count); //Add newly
created guest to the comboBox of guests (So user can than check how many guests there is
for the booking)
                        cmb_selectExtraType.IsEnabled = true; //Enable the comboBox extra
type (that will allow the user to create new Extras) - At least a guest must be created
before extras can be added
                        txt_name.Text = "";//clear the textbox
                        txt_passportNumber.Text = "";//clear the textbox
                        txt_age.Text = "";//clear the textbox
                    }
                    // otherwise guest is not added to the list and user is informed that
creation of guest failed due to the fact that the list id full
                    else
                    {
                        MessageBox.Show("Sorry! A booking may have a maximum of 4
guests."); //Inform the user that creation of guest failed due to the list being full
                        txt_name.Text = "";//clear the textbox
                        txt_passportNumber.Text = "";//clear the textbox
                        txt_age.Text = "";//clear the textbox
                    }
                }
                else
                {
                    MessageBox.Show("Please enter a valid age."); //Message shown when age
input is not an integer
                    txt_age.Text = "";//clear the textbox
                }
            }
            // Catch exception messages when assignement of Guest Properties from user
input fail
            catch (Exception excep)
            {
                MessageBox.Show(excep.Message);//Show exception message in a MessageBox
            }
        }


        //Method is called when the Selected item of the extra type comboBox changes
        private void cmb_selectExtraType_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            btn_extras.IsEnabled = true;//Enable add extra button when extra type is
selected
            //Change visibility properties of GUI object according to the type of extra
selected (Breakfast or Evening Meal), givin the user different input option
```

```
            if (cmb_selectExtraType.SelectedItem.ToString() == "Breakfast" ||
cmb_selectExtraType.SelectedItem.ToString() == "Evening meal")
            {
                lbl_dietaryRequirements.Visibility = System.Windows.Visibility.Visible;
                txt_dietaryRequirements.Visibility = System.Windows.Visibility.Visible;
                lbl_driverName.Visibility = System.Windows.Visibility.Hidden;
                lbl_startDate.Visibility = System.Windows.Visibility.Hidden;
                lbl_endDate.Visibility = System.Windows.Visibility.Hidden;
                txt_driverName.Visibility = System.Windows.Visibility.Hidden;
                txt_startDate.Visibility = System.Windows.Visibility.Hidden;
                txt_endDate.Visibility = System.Windows.Visibility.Hidden;
            }
            //Change visibility properties of GUI object according to the type of extra
selected (CarHire), givin the user different input option
            else if (cmb_selectExtraType.SelectedItem.ToString() == "Car hire")
            {
                lbl_dietaryRequirements.Visibility = System.Windows.Visibility.Hidden;
                txt_dietaryRequirements.Visibility = System.Windows.Visibility.Hidden;
                lbl_driverName.Visibility = System.Windows.Visibility.Visible;
                lbl_startDate.Visibility = System.Windows.Visibility.Visible;
                lbl_endDate.Visibility = System.Windows.Visibility.Visible;
                txt_driverName.Visibility = System.Windows.Visibility.Visible;
                txt_startDate.Visibility = System.Windows.Visibility.Visible;
                txt_endDate.Visibility = System.Windows.Visibility.Visible;
            }
            else
            {
                btn_extras.IsEnabled = false;//Disable add_extra button
            }
        }

        //Method called when add extra button is clicked. Gets user input to create a new
extra and add it to the global list of extras (tempExtras)
        private void btn_extras_Click(object sender, RoutedEventArgs e)
        {
            //If Brekfast extra is selected...
            if (cmb_selectExtraType.SelectedItem.ToString() == "Breakfast")
            {
                Breakfast aBreakfast = new Breakfast();//Create new Breakfast object
                tempExtras.Add(aBreakfast);//Add Breakfast Object to the booking list of
extras
                dietaryRequirements = txt_dietaryRequirements.Text;//Set the booking
dietary requirements from user input
                txt_dietaryRequirements.Text = "";//clears the textbox
                cmb_extras.IsEnabled = true; //Enable the extras combobox
                cmb_extras.Items.Add(cmb_selectExtraType.SelectedItem.ToString()); //Add
the new extra (type Breakfast) to the extras comboBox
                MessageBox.Show("Breakfast has been successfully added");//Informs the user
that Breakfast extra was added successfuly
            }
            //If Evening Meal extra is selected...
            else if (cmb_selectExtraType.SelectedItem.ToString() == "Evening meal")
            {
                EveningMeal anEveningMeal = new EveningMeal();//Create new EveningMeal
object
                tempExtras.Add(anEveningMeal);//Add Evening Meal Object to the booking list
of extras
                dietaryRequirements = txt_dietaryRequirements.Text;//Set the booking
dietary requirements from user input
                txt_dietaryRequirements.Text = "";//clears the textbox
                cmb_extras.IsEnabled = true;//Enable the extras combobox
```

```
                cmb_extras.Items.Add(cmb_selectExtraType.SelectedItem.ToString());//Add the
new extra (type EveningMeal) to the extras comboBox
                MessageBox.Show("Evening meal has been successfully added");//Informs the
user that EveningMeal extra was added successfuly
            }
            else if (cmb_selectExtraType.SelectedItem.ToString() == "Car hire")
            {
                DateTime arrivalDate;//local variable to hold a DateTime value (Car Hire
pick-up date)
                DateTime departureDate;//local variable to hold a DateTime value (Car Hire
return date)
                if (DateTime.TryParse(txt_startDate.Text, out arrivalDate)) //Check if
pick-up date input by the user is a valid DateTime format
                {
                    if(DateTime.TryParse(txt_endDate.Text, out departureDate)) //Check if
return date input by the user is a valid DateTime format
                    {
                        //Try to assign CarHire properties from user input and create a
CarHire extra
                        try
                        {
                            CarHire aCarHire = new CarHire(); //Create new CarHire object
                            aCarHire.DriverName = txt_driverName.Text; // (try to) Set
driver name from user input
                            aCarHire.StartDate = arrivalDate; // (try to) Set pick-up date
name from user input
                            aCarHire.ReturnDate = departureDate; // (try to) Set return
date from user input
                            tempExtras.Add(aCarHire); //Add Car Hire Object to the booking
list of extras
                            txt_driverName.Text = ""; //clear the textbox
                            txt_startDate.Text = ""; //clear the textbox
                            txt_endDate.Text = ""; //clear the textbox
                            cmb_extras.IsEnabled = true; //enable comboBox with the extras

cmb_extras.Items.Add(cmb_selectExtraType.SelectedItem.ToString()); //Add the new extra
(type CarHire) to the extras comboBox
                            MessageBox.Show("Car hire has been successfully
added");//Informs the user that CarHire extra was added successfuly
                        }
                        // Catch exception messages if the assignemt of CarHire properties
fail
                        catch (Exception excep)
                        {
                            MessageBox.Show(excep.Message);//Print message in Message Box
                        }

                    }
                    else
                    {
                        MessageBox.Show("Please enter a valid return date date. (YYYY-MM-
DD)");//Promp user to input a valid DateTime showing the correct format
                        txt_endDate.Text = "";//clear the textbox
                    }
                }
                else
                {
                    MessageBox.Show("Please enter a valid pick-up date. (YYYY-MM-
DD)");//Promp user to input a valid DateTime showing the correct format
                    txt_startDate.Text = "";//clear the textbox
                }
```

```
            }
        }


        private void btn_addNewBookingDone_Click(object sender, RoutedEventArgs e)
        {
            DateTime arrivalDate; //local variable to hold a DateTime value (Booking
arrivalDate date)
            DateTime departureDate; //local variable to hold a DateTime value (Booking
departureDate date)
            if (DateTime.TryParse(txt_arrivalDate.Text, out arrivalDate)) //Check if
Booking arrivalDate input by the user is a valid DateTime format
            {
                if (DateTime.TryParse(txt_departureDate.Text, out departureDate)) //Check
if Booking departureDate input by the user is a valid DateTime format
                {
                    if (tempGuests.Count != 0) //If there is at least 1 guest for the
booking
                    {
                        //Try to assign Booking properties from user input and create a
NewBooking with associated guests and extras
                        try
                        {
                            Booking aBooking = new Booking(); //Create new Booking object
                            aBooking.ArrivalDate = arrivalDate; // (try to) Set booking
arrivalDate from user input
                            aBooking.DepartureDate = departureDate; // (try to) Set booking
departureDate from user input
                            aBooking.Guests = tempGuests; //Add list of Guests (previously
created) to booking
                            aBooking.Extras = tempExtras; //Add list of Extras (previously
created) to booking
                            aBooking.DietaryRequirements = dietaryRequirements; // Set the
booking dietary requirements (if any) from user input
                            MainWindow mainWin = Owner as MainWindow; //Get a reference
from Main Window to access its properties

facade.SetCurrentCostumer(Int32.Parse(mainWin.txt_costumerReferenceNumber.Text)); //Set
CurrentCostumer in the facade using the reference number input by the user in Main WIndow
                            facade.AddBooking(aBooking); // Add the newly created booking
to the CurrentCustomer (using the facade method)
                            facade.SaveToCSV(); //Save new Booking to CSV
                            // Update text and combo Boxes on the Booking Menu (on main
window)
                            MessageBox.Show("Booking successfully added! Reference Number:"
+ aBooking.ReferenceNumber.ToString());// Inform the user the booking has been successfuly
created
                            mainWin.cmbBox_Bookings.IsEnabled = true; //Enable the Bookings
comboBox in Main window (so the user can select a booking)
                            mainWin.cmbBox_Bookings.Items.Add(aBooking.ReferenceNumber);
//Add newly created booking to the Bookings comboBox in main window
                            this.Close(); //Close the window
                        }
                        // Catch exception messages if the assignemt of Booking properties
fail
                        catch (Exception excep)
                        {
                            MessageBox.Show(excep.Message); //Print message in Message Box
                        }
                    }
                    else
```

```
                    {
                        MessageBox.Show("New bookings must have at least one guest. Please
add a guest to booking");
                    }
                }
                else
                {
                    MessageBox.Show("Departure date entered is not valid. Please enter a
valid date format (YYYY-MM-DD)");
                }
            }
            else
            {
                MessageBox.Show("Arrival date entered is not valid. Please enter a valid
date format (YYYY-MM-DD)");
            }
        }
    }
}
```

AddExtra

```
//Author: Pedro Mendes     MatricNum:40218056
//Description: GUI class used to get user input for extra details in order to create new
extras.
//Date last modified: 2016-12-07
//

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace HollydayBooking
{
    /// <summary>
    /// Interaction logic for addExtra.xaml
    /// </summary>
    public partial class addExtra : Window
    {
        //Get a GUIFacade reference
        GUIFacade facade = GUIFacade.Instance;
        public addExtra()
        {
            InitializeComponent();
        }

        // Change the GUI properties based on the type of extra to be added (Different
options are given to the user depending on the type of extra)
        private void cmb_extraTypes_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            btn_addExtra.IsEnabled = true; //Enable the addExtra button when user selects
an extra type from the combobox
            if (cmb_extraTypes.SelectedItem.ToString() == "Breakfast" ||
cmb_extraTypes.SelectedItem.ToString() == "Evening meal")
            {
                //Change visibility setting of labels and text boxes
                lbl_dietaryRequirements.Visibility = System.Windows.Visibility.Visible;
                txt_dietaryRequirements.Visibility = System.Windows.Visibility.Visible;
                lbl_driverName.Visibility = System.Windows.Visibility.Hidden;
                lbl_startDate.Visibility = System.Windows.Visibility.Hidden;
                lbl_endDate.Visibility = System.Windows.Visibility.Hidden;
                txt_driverName.Visibility = System.Windows.Visibility.Hidden;
                txt_startDate.Visibility = System.Windows.Visibility.Hidden;
                txt_endDate.Visibility = System.Windows.Visibility.Hidden;
            }
            else if (cmb_extraTypes.SelectedItem.ToString() == "Car hire")
            {
                //Change visibility setting of labels and text boxes
                lbl_dietaryRequirements.Visibility = System.Windows.Visibility.Hidden;
                txt_dietaryRequirements.Visibility = System.Windows.Visibility.Hidden;
                lbl_driverName.Visibility = System.Windows.Visibility.Visible;
                lbl_startDate.Visibility = System.Windows.Visibility.Visible;
                lbl_endDate.Visibility = System.Windows.Visibility.Visible;
```

```
                txt_driverName.Visibility = System.Windows.Visibility.Visible;
                txt_startDate.Visibility = System.Windows.Visibility.Visible;
                txt_endDate.Visibility = System.Windows.Visibility.Visible;
            }
        }

        private void btn_addExtra_Click(object sender, RoutedEventArgs e)
        {
            MainWindow mainWin = Owner as MainWindow; //Create a reference to the main
window (in order to have access to its properties)
            Costumer aCostumer = facade.CurrentCostumer; //Get the currentCostumer object
            Booking aBooking = facade.CurrentBooking; //Get the currentBooking object

            //If Brekfast extra is selected...
            if (cmb_extraTypes.SelectedItem.ToString() == "Breakfast")
            {
                Breakfast aBreakfast = new Breakfast();//Create new Breakfast object
                aBooking.Extras.Add(aBreakfast);//Add Breakfast Object to the booking list
of extras
                aBooking.DietaryRequirements = txt_dietaryRequirements.Text;//Set the
booking dietary requirements from user input
                mainWin.txt_bookingDietaryRequirements.Text =
aBooking.DietaryRequirements;//update the dietary requirement of booking on main window
textbox
                txt_dietaryRequirements.Text = "";//clear the textbox
                mainWin.cmb_bookingExtras.IsEnabled = true;//enable comboBox with the
extras on main window

mainWin.cmb_bookingExtras.Items.Add(cmb_extraTypes.SelectedItem.ToString());//Add extra
Type (Breakfast) to the extra comboBox in main window
                MessageBox.Show("Breakfast has been successfully added"); //Prompt user
that Breakfast extra has been created
                this.Close();
            }
            //If Evening Meal extra is selected...
            else if (cmb_extraTypes.SelectedItem.ToString() == "Evening meal")
            {
                EveningMeal anEveningMeal = new EveningMeal();//Create new Evening Meal
object
                aBooking.Extras.Add(anEveningMeal);//Add Evening Meal Object to the booking
list of extras
                aBooking.DietaryRequirements = txt_dietaryRequirements.Text;//Set the
booking dietary requirements from user input
                mainWin.txt_bookingDietaryRequirements.Text =
aBooking.DietaryRequirements;//update the dietary requirement of booking on main window
textbox
                txt_dietaryRequirements.Text = "";//clear the textbox
                mainWin.cmb_bookingExtras.IsEnabled = true;//enable comboBox with the
extras on main window

mainWin.cmb_bookingExtras.Items.Add(cmb_extraTypes.SelectedItem.ToString());//Add extra
Type (Evening Meal) to the extra comboBox in main window
                MessageBox.Show("Evening Meal has been successfully added"); //Prompt user
that Evening Meal extra has been created
                this.Close();
            }
            //If Car Hire extra is selected...
            else if (cmb_extraTypes.SelectedItem.ToString() == "Car hire")
            {
                DateTime arrivalDate; //local variable to hold a DateTime value (Car Hire
pick-up date)
                DateTime departureDate; //local variable to hold a DateTime value (Car Hire
return Date)
                if (DateTime.TryParse(txt_startDate.Text, out arrivalDate)) //Check if
pick-up date input is a valid DateTime value
                {
```

```
                    if (DateTime.TryParse(txt_endDate.Text, out departureDate)) //Check if
return date input is a valid DateTime value
                    {
                        // Try to assign input values to CarHire object
                        try
                        {
                            CarHire aCarHire = new CarHire(); //Create new Car Hire object
                            aCarHire.DriverName = txt_driverName.Text; // (try to) Set
driver name from user input
                            aCarHire.StartDate = arrivalDate; // (try to) Set pick-up date
from user input
                            aCarHire.ReturnDate = departureDate; // (try to) Set return
date from user input
                            aBooking.Extras.Add(aCarHire); //Add Car Hire Object to the
booking list of extras
                            txt_driverName.Text = "";//clear the textbox
                            txt_startDate.Text = "";//clear the textbox
                            txt_endDate.Text = "";//clear the textbox
                            mainWin.cmb_bookingExtras.IsEnabled = true; //enable comboBox
with the extras on main window

mainWin.cmb_bookingExtras.Items.Add(cmb_extraTypes.SelectedItem.ToString()); //Add extra
Type (Car Hire) to the extra comboBox in main window
                            MessageBox.Show("Car hire has been successfully added");
//Prompt user that Car Hire extra has been created
                            this.Close();
                        }
                        // Catch exception messages if the assignemt of CarHire properties
fail
                        catch(Exception excep)
                        {
                            MessageBox.Show(excep.Message);//Print message in Message Box
                        }
                    }
                    else
                    {
                        MessageBox.Show("Please enter a valid return date date. (YYYY-MM-
DD)");//Promp user to input a valid DateTime showing the correct format
                        txt_endDate.Text = "";//clear the textbox
                    }
                }
                else
                {
                    MessageBox.Show("Please enter a valid pick-up date. (YYYY-MM-DD)");
//Promp user to input a valid DateTime showing the correct format
                    txt_startDate.Text = "";//clear the textbox
                }
            }
        }
    }
}
```

MainWindow

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace HollydayBooking
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {

        GUIFacade facade = GUIFacade.Instance; //Get an instance of the GUI facade

        public MainWindow()
        {
            InitializeComponent();
            facade.LoadFromCSV(); //Load all records from CSV
        }

        // Opens a new window to allow the user to add new costumer
        private void btn_addCostumer_Click(object sender, RoutedEventArgs e)
        {
            AddNewCostumer addCostumerWin = new AddNewCostumer();
            addCostumerWin.ShowDialog();
        }


        // Finds a Costumer by getting a valid reference number input from the user (Does
some validation of user input)
        private void btn_findCostumer_Click(object sender, RoutedEventArgs e)
        {
            int refNumber;
            if (!Int32.TryParse(txt_costumerReferenceNumber.Text, out refNumber)) //if
refnumber input is not an int
            {
                MessageBox.Show("Please input a valid reference number");
                txt_costumerReferenceNumber.Text = "";
            }
            else // if reference number is a valid int
            {
                Costumer aCostumer = facade.GetCostumer(refNumber);
                if (aCostumer != null) //if costumer was found
                {
                    cmbBox_Bookings.Items.Clear();
                    cmbBox_Bookings.IsEnabled = false;
```

```
                    facade.SetCurrentCostumer(refNumber);
                    EnableCostumerMenuOptions();
                    lbl_costumerNameOutput.Content = aCostumer.Name;
                    lbl_costumerAddressOutput.Content = aCostumer.Address;

                    // Add all the booking reference numbers to the booking menu comboBox
                    foreach (Booking booking in aCostumer.CostumerBookings)
                    {
                        cmbBox_Bookings.Items.Add(booking.ReferenceNumber);
                        cmbBox_Bookings.IsEnabled = true;
                    }
                }
                else
                {
                    MessageBox.Show("Costumer with the reference number entered does not
exist.");
                    DisableCostumerMenuOptions();
                }
            }
        }

        //Method deletes a Customer Given the reference number
        private void btn_removeCostumer_Click(object sender, RoutedEventArgs e)
        {
            int refNumber;
            if (!Int32.TryParse(txt_costumerReferenceNumber.Text, out refNumber))
            {
                MessageBox.Show("Please input a valid reference number");
            }
            else
            {
                Costumer aCostumer = facade.GetCostumer(refNumber);
                if (aCostumer != null)
                {
                    facade.RemoveCostumer(refNumber);
                    facade.SaveToCSV();
                    MessageBox.Show("Costumer Successfully removed");
                }
                else
                {
                    MessageBox.Show("Costumer with the reference number entered does not
exist.");
                }
            }
            txt_costumerReferenceNumber.Text = "";
            DisableCostumerMenuOptions();
        }

        //Change some main window properties on text changed (disable buttons and clears
textboxs and comboboxes)
        private void txt_costumerReferenceNumber_TextChanged(object sender,
TextChangedEventArgs e)
        {
            cmbBox_Bookings.Items.Clear();
            cmbBox_Bookings.IsEnabled = false;
            btn_searchBooking.IsEnabled = false;
            DisableCostumerMenuOptions();
        }

        //Open new window to allow user to amend an existing customer
        private void btn_amendCostumer_Click(object sender, RoutedEventArgs e)
```

```
        {
            AmendCostumer amendCostumerWin = new AmendCostumer();
            amendCostumerWin.Owner = this;
            amendCostumerWin.txt_amendCostumerRefNumber.Text =
txt_costumerReferenceNumber.Text;
            amendCostumerWin.txt_amendCostumerName.Text =
lbl_costumerNameOutput.Content.ToString();
            amendCostumerWin.txt_amendCostumerAddress.Text =
lbl_costumerAddressOutput.Content.ToString();
            amendCostumerWin.ShowDialog();
        }


        //Open new window to allow user to add a new booking
        private void btn_addBooking_Click(object sender, RoutedEventArgs e)
        {
            int refNumb;
            if (Int32.TryParse(txt_costumerReferenceNumber.Text, out refNumb))
            {
                AddNewBooking addNewBookingWin = new AddNewBooking();
                addNewBookingWin.Owner = this;
                addNewBookingWin.ShowDialog();
            }
            else
            {
                MessageBox.Show("Please select a costumer before adding a booking.");
            }
        }

        //Change some main window properties on text changed (disable buttons and clears
textboxs and comboboxes)
        private void cmbBox_Bookings_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            if (cmbBox_Bookings.SelectedItem != null)
            {
                if (cmbBox_Bookings.SelectedItem.ToString() == "")
                {
                    btn_removeBooking.IsEnabled = false;
                    btn_searchBooking.IsEnabled = false;
                    btn_invoice.IsEnabled = false;
                }
                else
                {

facade.SetCurrentBooking(Int32.Parse(cmbBox_Bookings.SelectedItem.ToString()));
                    btn_removeBooking.IsEnabled = true;
                    btn_searchBooking.IsEnabled = true;
                    btn_invoice.IsEnabled = true;
                }
            }
            else
            {
                btn_removeBooking.IsEnabled = false;
                btn_searchBooking.IsEnabled = false;
                btn_invoice.IsEnabled = false;
            }
        }


        // Loads the booking selected in the combobox and loads its details to main window
```

```
        private void btn_searchBooking_Click(object sender, RoutedEventArgs e)
        {
            Costumer aCostumer = facade.CurrentCostumer;
            Booking aBooking = null;
            if (cmbBox_Bookings.SelectedItem != null)
            {
                aBooking =
facade.GetBooking(Int32.Parse(cmbBox_Bookings.SelectedItem.ToString()));
            }

            if (aBooking == null)
            {
                MessageBox.Show("Booking does not exist");
            }
            else
            {
                facade.SetCurrentBooking(aBooking.ReferenceNumber);
                ClearBookingMenu();
                txt_bookingReferenceNumber.Text = aBooking.ReferenceNumber.ToString();
                txt_bookingArrivalDate.Text =
aBooking.ArrivalDate.ToString(aBooking.DatePattern);
                txt_bookingDepartureDate.Text =
aBooking.DepartureDate.ToString(aBooking.DatePattern);
                txt_bookingDietaryRequirements.Text =
aBooking.DietaryRequirements.ToString();
                btn_removeBooking.IsEnabled = true;
                btn_addGuest.IsEnabled = true;
                btn_addExtra.IsEnabled = true;
                btn_saveChanges.IsEnabled = true;
                btn_invoice.IsEnabled = true;
                foreach (Guest aGuest in aBooking.Guests)
                {
                    cmb_bookingGuests.Items.Add(aGuest.Name);
                }
                cmb_bookingGuests.IsEnabled = true;

                if (aBooking.Extras.Count > 0)
                {
                    cmb_bookingExtras.IsEnabled = true;
                    foreach (Extra anExtra in aBooking.Extras)
                    {
                        if (anExtra.GetType().ToString() == "HollydayBooking.Breakfast")
                        {
                            cmb_bookingExtras.Items.Add("Breakfast");
                        }
                        else if (anExtra.GetType().ToString() ==
"HollydayBooking.EveningMeal")
                        {
                            cmb_bookingExtras.Items.Add("Evening meal");
                        }
                        else if (anExtra.GetType().ToString() == "HollydayBooking.CarHire")
                        {
                            cmb_bookingExtras.Items.Add("Car hire");
                        }
                    }
                }

            }
        }

        //Add a new guest to an existing booking
```

```
        private void btn_addGuest_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                Costumer aCostumer = facade.CurrentCostumer;
                Booking aBooking = facade.CurrentBooking;
                Guest aGuest = new Guest();
                aGuest.Name = txt_guestName.Text;
                aGuest.PassportNumber = txt_guestPassportNumber.Text;
                int guestAge;
                if (Int32.TryParse(txt_guestAge.Text, out guestAge))
                {
                    aGuest.Age = Int32.Parse(txt_guestAge.Text);
                    aBooking.Guests.Add(aGuest);
                    cmb_bookingGuests.Items.Add(aGuest.Name.ToString());
                    MessageBox.Show("New guest was successfully added!");
                    txt_guestName.Text = "";
                    txt_guestPassportNumber.Text = "";
                    txt_guestAge.Text = "";
                }
                else
                {
                    MessageBox.Show("Please insert a valid Guest age!");
                }
            }
            catch (Exception excep)
            {
                MessageBox.Show(excep.Message);
            }
        }

        // Loads details of the selected guest in the combo box
        private void cmb_bookingGuests_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            Costumer aCostumer = facade.CurrentCostumer;
            Guest aGuest = null;
            if (cmb_bookingGuests.SelectedItem != null)
            {
                Booking aBooking = facade.CurrentBooking;
                if (cmb_bookingGuests.SelectedIndex < aBooking.Guests.Count())
                {
                    aGuest = facade.GetGuest(cmb_bookingGuests.SelectedIndex);
                }
            }

            if (aGuest != null)
            {
                txt_guestName.Text = aGuest.Name;
                txt_guestPassportNumber.Text = aGuest.PassportNumber;
                txt_guestAge.Text = aGuest.Age.ToString();
                btn_deleteGuest.IsEnabled = true;
                btn_amendGuest.IsEnabled = true;
            }
            else
            {
                btn_deleteGuest.IsEnabled = false;
                btn_amendGuest.IsEnabled = false;
            }
        }
```

```
        // Delete the selected booking from the combobox
        private void btn_removeBooking_Click(object sender, RoutedEventArgs e)
        {
            int bookingRefNumber = Int32.Parse(cmbBox_Bookings.SelectedItem.ToString());
            facade.RemoveBooking(bookingRefNumber);
            facade.SaveToCSV();
            MessageBox.Show("Booking with referece number: " + bookingRefNumber + " has
been successfully deleted!");
            cmbBox_Bookings.Items.RemoveAt(cmbBox_Bookings.SelectedIndex);
            cmbBox_Bookings.Items.Clear();
            cmbBox_Bookings.IsEnabled = false;
            ClearBookingMenu();
        }

        // Delete the selected guest from the combobox
        private void btn_deleteGuest_Click(object sender, RoutedEventArgs e)
        {
            facade.RemoveGuest(cmb_bookingGuests.SelectedIndex);
            MessageBox.Show("Guest sucessfully deleted!");
            cmb_bookingGuests.Items.RemoveAt(cmb_bookingGuests.SelectedIndex);
            txt_guestName.Text = "";
            txt_guestPassportNumber.Text = "";
            txt_guestAge.Text = "";

            if (cmb_bookingGuests.Items.Count == 0)
            {
                btn_deleteGuest.IsEnabled = false;
                btn_amendGuest.IsEnabled = false;
            }
        }

        // Amend a guest given its index in the list of guests
        private void btn_amendGuest_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                int guestAge;
                if (Int32.TryParse(txt_guestAge.Text, out guestAge))
                {
                    Guest aGuest = facade.GetGuest(cmb_bookingGuests.SelectedIndex);
                    aGuest.Name = txt_guestName.Text;
                    aGuest.PassportNumber = txt_guestPassportNumber.Text;
                    aGuest.Age = Int32.Parse(txt_guestAge.Text);
                    MessageBox.Show("Guest details have been successfully updated!");
                }
                else
                {
                    MessageBox.Show("Please enter a valid age for guest!");
                }
            }
            catch (Exception excep)
            {
                MessageBox.Show(excep.Message);
            }

        }

        // Open new window that allows user to add a new extra to booking
        private void btn_addExtra_Click(object sender, RoutedEventArgs e)
        {
            addExtra extraWin = new addExtra();
```

```
            extraWin.cmb_extraTypes.Items.Add("Breakfast");
            extraWin.cmb_extraTypes.Items.Add("Evening meal");
            extraWin.cmb_extraTypes.Items.Add("Car hire");
            extraWin.Owner = this;
            extraWin.ShowDialog();
        }


        // Saves all changes made to booking (to the csv file)
        private void btn_saveChanges_Click(object sender, RoutedEventArgs e)
        {
            DateTime arrivalDate;
            DateTime departureDate;
            if (DateTime.TryParse(txt_bookingArrivalDate.Text, out arrivalDate))
            {
                if (DateTime.TryParse(txt_bookingDepartureDate.Text, out departureDate))
                {
                    try
                    {
                        Booking aBooking = facade.CurrentBooking;
                        aBooking.ArrivalDate = arrivalDate;
                        aBooking.DepartureDate = departureDate;
                        facade.SaveToCSV();
                        MessageBox.Show("Booking details were successfully saved!");
                    }
                    catch (Exception excep)
                    {
                        MessageBox.Show(excep.Message);
                    }
                }
                else
                {
                    MessageBox.Show("Please enter a valid Departure Date (YYYY-MM-DD)");
                }
            }
            else
            {
                MessageBox.Show("Please enter a valid Arrival Date (YYYY-MM-DD)");
            }
        }


        // Loads the extra selected extra details to Mainwindow
        private void cmb_bookingExtras_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            Extra selectedExtra = null;
            if (cmb_bookingExtras.SelectedItem != null)
            {
                selectedExtra = facade.GetExtra(cmb_bookingExtras.SelectedIndex);
            }

            if (selectedExtra != null)
            {
                if (cmb_bookingExtras.SelectedItem.ToString() == "Breakfast" ||
cmb_bookingExtras.SelectedItem.ToString() == "Evening meal")
                {
                    lbl_extraDietaryRequirements.Visibility =
System.Windows.Visibility.Visible;
                    txt_extraDietaryRequirements.Visibility =
System.Windows.Visibility.Visible;
```

```
                    lbl_extraDriverName.Visibility = System.Windows.Visibility.Hidden;
                    txt_extraDriverName.Visibility = System.Windows.Visibility.Hidden;
                    lbl_extraStartDate.Visibility = System.Windows.Visibility.Hidden;
                    txt_extraStartDate.Visibility = System.Windows.Visibility.Hidden;
                    lbl_extraEndDate.Visibility = System.Windows.Visibility.Hidden;
                    txt_extraEndDate.Visibility = System.Windows.Visibility.Hidden;
                    txt_extraDietaryRequirements.Text =
txt_bookingDietaryRequirements.Text;
                }
                else if (cmb_bookingExtras.SelectedItem.ToString() == "Car hire")
                {
                    CarHire aCarHire = (CarHire)selectedExtra;
                    lbl_extraDietaryRequirements.Visibility =
System.Windows.Visibility.Hidden;
                    txt_extraDietaryRequirements.Visibility =
System.Windows.Visibility.Hidden;
                    lbl_extraDriverName.Visibility = System.Windows.Visibility.Visible;
                    txt_extraDriverName.Visibility = System.Windows.Visibility.Visible;
                    lbl_extraStartDate.Visibility = System.Windows.Visibility.Visible;
                    txt_extraStartDate.Visibility = System.Windows.Visibility.Visible;
                    lbl_extraEndDate.Visibility = System.Windows.Visibility.Visible;
                    txt_extraEndDate.Visibility = System.Windows.Visibility.Visible;
                    txt_extraDriverName.Text = aCarHire.DriverName;
                    txt_extraStartDate.Text = aCarHire.StartDate.ToString("yyyy-MM-dd");
                    txt_extraEndDate.Text = aCarHire.ReturnDate.ToString("yyyy-MM-dd");
                }
                btn_deleteExtra.IsEnabled = true;
                btn_amendExtra.IsEnabled = true;
            }
            else
            {
                btn_deleteExtra.IsEnabled = false;
                btn_amendExtra.IsEnabled = false;
            }
        }


        //Deletes the selected extra
        private void btn_deleteExtra_Click(object sender, RoutedEventArgs e)
        {
            facade.RemoveExtra(cmb_bookingExtras.SelectedIndex);
            MessageBox.Show("Extra sucessfully deleted!");
            cmb_bookingExtras.Items.RemoveAt(cmb_bookingExtras.SelectedIndex);

            lbl_extraDietaryRequirements.Visibility = System.Windows.Visibility.Hidden;
            txt_extraDietaryRequirements.Visibility = System.Windows.Visibility.Hidden;
            lbl_extraDriverName.Visibility = System.Windows.Visibility.Hidden;
            txt_extraDriverName.Visibility = System.Windows.Visibility.Hidden;
            lbl_extraStartDate.Visibility = System.Windows.Visibility.Hidden;
            txt_extraStartDate.Visibility = System.Windows.Visibility.Hidden;
            lbl_extraEndDate.Visibility = System.Windows.Visibility.Hidden;
            txt_extraEndDate.Visibility = System.Windows.Visibility.Hidden;

            txt_extraDietaryRequirements.Text = "";
            txt_extraDriverName.Text = "";
            txt_extraStartDate.Text = "";
            txt_extraEndDate.Text = "";


            if (cmb_bookingExtras.Items.Count == 0)
            {
```

```
                btn_deleteExtra.IsEnabled = false;
                btn_amendExtra.IsEnabled = false;
            }
        }

        // Amends the selected extra
        private void btn_amendExtra_Click(object sender, RoutedEventArgs e)
        {

            Booking aBooking = facade.CurrentBooking;
            Extra anExtra = facade.GetExtra(cmb_bookingExtras.SelectedIndex);
            if (anExtra.GetType().ToString() == "HollydayBooking.Breakfast") //Check if
extra is of type Brekfast
            {
                Breakfast aBreakfast = (Breakfast) anExtra;
                aBooking.DietaryRequirements = txt_extraDietaryRequirements.Text;
                txt_bookingDietaryRequirements.Text = txt_extraDietaryRequirements.Text;
                MessageBox.Show("Extra details have been successfully updated!");
            }
            else if (anExtra.GetType().ToString() == "HollydayBooking.EveningMeal") //Check
if extra is of type EveningMeal
            {
                EveningMeal anEveningMeal = (EveningMeal) anExtra;
                aBooking.DietaryRequirements = txt_extraDietaryRequirements.Text;
                txt_bookingDietaryRequirements.Text = txt_extraDietaryRequirements.Text;
                MessageBox.Show("Extra details have been successfully updated!");
            }
            else if (anExtra.GetType().ToString() == "HollydayBooking.CarHire") //Check if
extra is of type CarHire
            {
                DateTime arrivalDate;
                DateTime departureDate;
                if (DateTime.TryParse(txt_extraStartDate.Text, out arrivalDate))
                {
                    if (DateTime.TryParse(txt_extraEndDate.Text, out departureDate))
                    {
                        try
                        {
                            CarHire aCarHire = (CarHire)anExtra;
                            aCarHire.DriverName = txt_extraDriverName.Text;
                            aCarHire.StartDate = arrivalDate;
                            aCarHire.ReturnDate = departureDate;
                            MessageBox.Show("Extra details have been successfully
updated!");
                        }
                        catch (Exception excep)
                        {
                            MessageBox.Show(excep.Message);
                        }
                    }
                    else
                    {
                        MessageBox.Show("Please enter a valid return date. (YYYY-MM-DD)");
                        txt_extraEndDate.Text = "";
                    }
                }
                else
                {
                    MessageBox.Show("Please enter a valid pick-up date. (YYYY-MM-DD)");
                    txt_extraStartDate.Text = "";
                }
```

```
    }
}


// Enable the window properties for the Costumer menu section
public void EnableCostumerMenuOptions()
{
    lbl_costumerName.Visibility = System.Windows.Visibility.Visible;
    lbl_costumerAddress.Visibility = System.Windows.Visibility.Visible;
    lbl_costumerNameOutput.Visibility = System.Windows.Visibility.Visible;
    lbl_costumerAddressOutput.Visibility = System.Windows.Visibility.Visible;

    btn_removeCostumer.IsEnabled = true;
    btn_amendCostumer.IsEnabled = true;
    btn_addBooking.IsEnabled = true;
}

// Disable the window properties for the Costumer menu section
public void DisableCostumerMenuOptions()
{
    btn_removeCostumer.IsEnabled = false;
    btn_amendCostumer.IsEnabled = false;
    btn_addBooking.IsEnabled = false;
    lbl_costumerNameOutput.Content = "";
    lbl_costumerAddressOutput.Content = "";
    lbl_costumerName.Visibility = System.Windows.Visibility.Hidden;
    lbl_costumerAddress.Visibility = System.Windows.Visibility.Hidden;
    lbl_costumerNameOutput.Visibility = System.Windows.Visibility.Hidden;
    lbl_costumerAddressOutput.Visibility = System.Windows.Visibility.Hidden;
    ClearBookingMenu();
}

// Clears/Disable window properties for the for the Booking menu section
private void ClearBookingMenu()
{
    btn_removeBooking.IsEnabled = false;
    txt_bookingReferenceNumber.Text = "";
    txt_bookingArrivalDate.Text = "";
    txt_bookingDepartureDate.Text = "";
    txt_bookingDietaryRequirements.Text = "";
    cmb_bookingGuests.Items.Clear();
    cmb_bookingGuests.IsEnabled = false;
    btn_addGuest.IsEnabled = false;
    btn_amendGuest.IsEnabled = false;
    btn_deleteGuest.IsEnabled = false;
    txt_guestName.Text = "";
    txt_guestPassportNumber.Text = "";
    txt_guestAge.Text = "";
    cmb_bookingExtras.Items.Clear();
    cmb_bookingExtras.IsEnabled = false;
    btn_addExtra.IsEnabled = false;
    btn_amendExtra.IsEnabled = false;
    btn_deleteExtra.IsEnabled = false;
    btn_saveChanges.IsEnabled = false;
    lbl_extraDietaryRequirements.Visibility = System.Windows.Visibility.Hidden;
    txt_extraDietaryRequirements.Visibility = System.Windows.Visibility.Hidden;
    lbl_extraDriverName.Visibility = System.Windows.Visibility.Hidden;
    txt_extraDriverName.Visibility = System.Windows.Visibility.Hidden;
    lbl_extraStartDate.Visibility = System.Windows.Visibility.Hidden;
    txt_extraStartDate.Visibility = System.Windows.Visibility.Hidden;
    lbl_extraEndDate.Visibility = System.Windows.Visibility.Hidden;
```

```
            txt_extraEndDate.Visibility = System.Windows.Visibility.Hidden;
        }

        // Opens a new window and prints a detailed invoice for the selected booking
        private void btn_invoice_Click(object sender, RoutedEventArgs e)
        {
            Invoice inv = new Invoice();
            Costumer aCostumer = facade.CurrentCostumer;
            Booking aBooking = facade.CurrentBooking;
            inv.lbl_invoiceDateOutput.Content = DateTime.Now.ToString("yyyy-MM-dd");
            inv.lbl_invoiceBookingIDOutput.Content = aBooking.ReferenceNumber;
            inv.lbl_invoiceNumberOfGuestsOutput.Content = aBooking.Guests.Count();
            inv.lbl_customerIDOutput.Content = aCostumer.ReferenceNumber;
            inv.lbl_customerNameOutput.Content = aCostumer.Name;
            inv.lbl_customerAddressOutput.Content = aCostumer.Address;
            foreach (Guest aGuest in aBooking.Guests)
            {
                if (aGuest.Age <= 18)
                {
                    inv.lbl_bookingDescription.Content += "Guest" +
(aBooking.Guests.IndexOf(aGuest) + 1) + ": £30.00 (child rate) x " +
(aBooking.DepartureDate - aBooking.ArrivalDate).TotalDays + " nights\n";
                    inv.lbl_bookingAmount.Content += 30.00*((aBooking.DepartureDate -
aBooking.ArrivalDate).TotalDays) + "£\n";
                }
                else
                {
                    inv.lbl_bookingDescription.Content += "Guest" +
(aBooking.Guests.IndexOf(aGuest) + 1) + ": £50.00 (adult rate) x " +
(aBooking.DepartureDate - aBooking.ArrivalDate).TotalDays + " nights\n";
                    inv.lbl_bookingAmount.Content += 50.00 * ((aBooking.DepartureDate -
aBooking.ArrivalDate).TotalDays) + "£\n";
                }
            }
            foreach (Extra anExtra in aBooking.Extras)
            {
                if (anExtra.GetType().ToString() == "HollydayBooking.Breakfast")
                {
                    inv.lbl_bookingDescription.Content += "Breakfast: " + "£5.00 x " +
aBooking.Guests.Count()+ " guests x " + (aBooking.DepartureDate -
aBooking.ArrivalDate).TotalDays + " nights\n";
                    inv.lbl_bookingAmount.Content += anExtra.Price *
aBooking.Guests.Count() * ((aBooking.DepartureDate - aBooking.ArrivalDate).TotalDays) +
"£\n";
                }
                else if (anExtra.GetType().ToString() == "HollydayBooking.EveningMeal")
                {
                    inv.lbl_bookingDescription.Content += "Evening Meal: " + "£15.00 x " +
aBooking.Guests.Count() + " guests x " + (aBooking.DepartureDate -
aBooking.ArrivalDate).TotalDays + " nights\n";
                    inv.lbl_bookingAmount.Content += anExtra.Price *
aBooking.Guests.Count() * ((aBooking.DepartureDate - aBooking.ArrivalDate).TotalDays) +
"£\n";
                }
                else if (anExtra.GetType().ToString() == "HollydayBooking.CarHire")
                {
                    CarHire aCarHire = (CarHire)anExtra;
                    inv.lbl_bookingDescription.Content += "Car Hire: " + "£50.00 x " +
(aCarHire.ReturnDate - aCarHire.StartDate).TotalDays + " days\n";
                    inv.lbl_bookingAmount.Content += aCarHire.Price * ((aCarHire.ReturnDate
- aCarHire.StartDate).TotalDays) + "£\n";
```

```
                }
            }
            inv.lbl_invoiceTotalAmountOutput.Content = aBooking.GetCost() + "£";
            inv.ShowDialog();
        }
    }
}
```

Invoice

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace HollydayBooking
{
    /// <summary>
    /// Interaction logic for Invoice.xaml
    /// </summary>
    public partial class Invoice : Window
    {
        public Invoice()
        {
            InitializeComponent();
        }
    }
}
```

Test Class

BookingClassTest

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using HollydayBooking;

namespace BookingClassTest
{
    [TestClass]
    public class BankClassTests
    {
        [TestMethod]
        [ExpectedException(typeof(ArgumentException))]
        //Method will pass if Argument exception is thrown (When )
        public void DepartureDate_Is_Before_Arrival_Date_Test()
        {
            DateTime arrivalDate = DateTime.Parse("2016-08-08");
            DateTime departureDate = DateTime.Parse("2016-08-06");

            Booking aBooking = new Booking();
            aBooking.ArrivalDate = arrivalDate;
            aBooking.DepartureDate = departureDate; //Should throw exception
        }

        [TestMethod]
        //Method will pass if departure date can be successfuly set
        public void DepartureDate_Is_After_Arrival_Date_Test()
        {
            DateTime arrivalDate = DateTime.Parse("2016-08-08");
            DateTime departureDate = DateTime.Parse("2016-08-11");

            Booking aBooking = new Booking();
            aBooking.ArrivalDate = arrivalDate;
            aBooking.DepartureDate = departureDate;
            //Check if the departure date for booking was successfuly set
            Assert.AreEqual(aBooking.DepartureDate, departureDate, "Booking departure date
successfully set!");
        }

        [TestMethod]
        //Method will pass if Guest is added to Booking guest list successfuly
        public void Add_Guests_Working_Correctly()
        {
            Booking aBooking = new Booking();
            Guest aGuest = new Guest();
            aBooking.AddGuest(aGuest);
            //The number of guests in the guest list must be 1 at this point
            Assert.AreEqual(aBooking.Guests.Count, 1, "Guest sucessfuly added to
booking!");
        }

        [TestMethod]
        [ExpectedException(typeof(ArgumentException))]
        //Method throw exception when adding a new guest to alist with 4 guests (maximum
number of guests = 4)
        public void Add_Guest_Fail_When_4_Guests_Already_In_The_List()
        {
            Booking aBooking = new Booking();
            //Create 5 guests
            Guest Guest1 = new Guest();
```

```
            Guest Guest2 = new Guest();
            Guest Guest3 = new Guest();
            Guest Guest4 = new Guest();
            Guest Guest5 = new Guest();

            //Add guests to booking list
            aBooking.AddGuest(Guest1);
            aBooking.AddGuest(Guest2);
            aBooking.AddGuest(Guest3);
            aBooking.AddGuest(Guest4);
            aBooking.AddGuest(Guest5); //This line should throw an Argument exception
        }

        [TestMethod]
        // Returns a null guest if guest cannot be found in the list
        public void GetGuest_Method_Returns_Null_If_Guest_Does_Not_Exist()
        {
            Booking aBooking = new Booking();
            //Create Guests
            Guest Guest1 = new Guest();
            Guest Guest2 = new Guest();
            //Add guests to booking list
            aBooking.AddGuest(Guest1);
            aBooking.AddGuest(Guest2);

            int index = 2;
            Guest Guest3 = aBooking.GetGuest(index); //Should return a null Guest
            Assert.AreEqual(Guest3, null, "Guest can not be found! (null)");
        }

        [TestMethod]
        // Returns the right guest if guest exist
        public void GetGuest_Method_Returns_The_Right_Guest_If_It_Does_Exist_In_The_List()
        {
            Booking aBooking = new Booking();
            //Create Guests
            Guest Guest1 = new Guest();
            Guest1.Name = "zero";
            aBooking.AddGuest(Guest1);

            Guest Guest2 = new Guest();
            Guest2.Name = "one";
            aBooking.AddGuest(Guest2);

            Guest Guest3 = new Guest();
            Guest3.Name = "two";
            aBooking.AddGuest(Guest3);


            int index = 0;
            Guest GuestOne = aBooking.GetGuest(index); //Should return the guest at index 0
(Guest1)
            Assert.AreEqual(Guest1.Name, GuestOne.Name, "Method returns the right guest");
        }

        [TestMethod]
        [ExpectedException(typeof(ArgumentException))]
        // Throws exception if guest with the provided index does not exist
        public void RemoveGuest_Method_Throws_Exception_If_Guest_Does_Not_Exist()
        {
            Booking aBooking = new Booking();
            //Create Guest
            Guest Guest1 = new Guest();
            //Add guest to booking list
            aBooking.AddGuest(Guest1);
```

```
            int index = 2;
            aBooking.RemoveGuest(index); // Should throw exception since guest at index 2
does not exist
        }


        [TestMethod]
        // Throws exception if guest with the provided index does not exist
        public void RemoveGuest_Method_Removes_Guest_If_Guest_Does_Exist()
        {
            Booking aBooking = new Booking();
            //Create Guest
            Guest Guest1 = new Guest();
            Guest Guest2 = new Guest();
            //Add guest to booking list
            aBooking.AddGuest(Guest1);
            aBooking.AddGuest(Guest2);

            int index = 1;
            aBooking.RemoveGuest(index); // Should remove the Guest at index 1
            Assert.AreEqual(aBooking.Guests.Count, 1, "Guest was removed successfuly");
        }

        [TestMethod]
        // Should add an extra successfuly given an extra object
        public void AddExtra_Method_Works_Correctly()
        {
            Booking aBooking = new Booking();
            //Create Guest
            Extra Extra1 = new Breakfast();

            aBooking.AddExtra(Extra1); //Should add the extra to the list of extras
correctly
            Assert.AreEqual(aBooking.Extras.Count, 1, "Guest was removed successfuly");
        }


        [TestMethod]
        // Should return the right extra given its index
        public void GetExtra_Method_Returns_The_Right_Extra_If_It_Does_Exist_In_The_List()
        {
            Booking aBooking = new Booking();
            //Create Extras
            Extra Extra1 = new Breakfast();
            Extra Extra2 = new CarHire();
            Extra Extra3 = new Breakfast();

            aBooking.AddExtra(Extra1);
            aBooking.AddExtra(Extra2);
            aBooking.AddExtra(Extra3);

            CarHire carhire1 = (CarHire)Extra2;
            int index = 1;
            CarHire carhire2 = (CarHire)aBooking.GetExtra(index); //Should return Extra2

            Assert.AreEqual(carhire1.DriverName, carhire2.DriverName, "Method returns the
right extra");
        }

        [TestMethod]
        // Returns a null extra if extra cannot be found in the list
        public void GetExtra_Method_Returns_Null_If_Extra_Does_Not_Exist()
        {
            Booking aBooking = new Booking();
            //Create Extras
            Extra Extra1 = new Breakfast();
```

```
            Extra Extra2 = new CarHire();
            //Add Extras to booking list
            aBooking.AddExtra(Extra1);
            aBooking.AddExtra(Extra2);

            int index = 2;
            Extra extra3 = aBooking.GetExtra(index); //Should return a null Extra
            Assert.AreEqual(extra3, null, "Guest can not be found! (null)");
        }

        [TestMethod]
        public void RemoveExtra_Method_Removes_Guest_If_Guest_Does_Exist()
        {
            Booking aBooking = new Booking();
            //Create Extras
            Extra Extra1 = new Breakfast();
            Extra Extra2 = new CarHire();
            //Add Extras to booking list
            aBooking.AddExtra(Extra1);
            aBooking.AddExtra(Extra2);

            int index = 1;
            aBooking.RemoveExtra(index); // Should remove the Extra at index 1
            Assert.AreEqual(aBooking.Extras.Count, 1, "Extra was removed successfuly");
        }

        [TestMethod]
        [ExpectedException(typeof(ArgumentException))]
        // Throws exception if extra with the provided index does not exist
        public void RemoveExtra_Method_Throws_Exception_If_Extra_Does_Not_Exist()
        {
            Booking aBooking = new Booking();
            //Create extra
            Extra extra1 = new Breakfast();
            //Add extra to booking list
            aBooking.AddExtra(extra1);

            int index = 2;
            aBooking.RemoveExtra(index); // Should throw exception since extra at index 2
does not exist
        }

        [TestMethod]
        public void GetCost_Method_Calculates_Cost_Of_Bookings_Correctly()
        {
            Booking aBooking = new Booking();
            // Booking for 2 nights
            aBooking.ArrivalDate = DateTime.Parse("2000-01-01");
            aBooking.DepartureDate = DateTime.Parse("2000-01-03");

            //Create 2 guests
            Guest guest1 = new Guest();
            guest1.Age = 12; //Create a guest (child charged at £30.00/night)
            aBooking.AddGuest(guest1);
            Guest guest2 = new Guest();
            guest2.Age = 32; //Create a guest (adult charged at £50.00/night)
            aBooking.AddGuest(guest2);

            //Create Extras
            Breakfast breakfast1 = new Breakfast(); //Breakfast charged at £5.00 per guest
per night
            CarHire carhire1 = new CarHire(); //CarHire charged at £50.00/day
            //Car hire booked for 1 day
            carhire1.StartDate = DateTime.Parse("2000-01-01");
            carhire1.ReturnDate = DateTime.Parse("2000-01-02");
            aBooking.AddExtra(breakfast1);
```

```
            aBooking.AddExtra(carhire1);


            double expected = 230.00; //expected cost for booking
            double calculated = aBooking.GetCost(); //Cost returned from the GetCost method

            Assert.AreEqual(expected, calculated, "GetCost method calculated booking costs
correctly!");
        }
    }
}
```