Pierre Ruiz  
matriculation 08009004

**Coursework 2 submission for  
Software Development 2  
(SET08108 2016-7)**

1. Source code: PersonComponent.cs
2. Source code: Person.cs
3. Source code: PersonDecorator.cs
4. Source code: Customer.cs
5. Source code: Guest.cs
6. Source code: PersonFactory.cs
7. Source code: BookingComponent.cs

using System;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Program

{

/\*

\* Abstract component class used to make sure that

\* components and decorators share the same specification

\*

\* author: Pierre Ruiz (matriculation number 08009004)

\* last modified: 2016-12-06

\*/

public abstract class BookingComponent

{

/\*

\* Must add a guest to a given BookingComponent.

\*/

public abstract void AddGuest(PersonComponent guest);

/\*

\* Must return the Booking number.

\*/

public abstract int GetBookingNb();

/\*

\* Must return the Booking's cutomer.

\*/

public abstract PersonComponent GetCustomer();

/\*

\* Must return the Booking's list of guests.

\*/

public abstract List<PersonComponent> GetGuests();

/\*

\* Must return the number of guests included in this

\* BookingComponent.

\*/

public abstract int GetNbGuests();

/\*

\* Must return the Booking start and end dates.

\*/

public abstract void GetDates(out DateTime arrival,

out DateTime departure);

/\*

\* Must return the number of nights booked.

\*/

public abstract int GetNbNights();

/\*

\* Must return the the cost for each individual night booked.

\*/

public abstract float GetCostPerNight();

/\*

\* Returns the BookingComponent itself; and references is null.

\*/

public virtual BookingComponent Unwrap(

out List<BookingDecorator> references)

{

references = null;

return this;

}

/\*

\* Returns true if the BookingDecorator wraps another

\* BookingDecorator, otherwise false.

\*/

public virtual bool isDecorator()

{

return false;

}

/\*

\* Returns the BookingComponent itself.

\*/

public virtual BookingComponent Undecorate(BookingDecorator reference)

{

return this;

}

/\*

\* Must return a textual representation of the

\* BookingComponent in order to persist it to a CSV file.

\*/

public abstract String ToCSV();

}

}

1. Source code: Booking.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Program

{

/\*

\* Represents a booking.

\*

\* author: Pierre Ruiz (matriculation number 08009004)

\* last modified: 2016-12-06

\*/

public class Booking : BookingComponent

{

// PROPERTIES:

// the booking reference number.

private int bookingNb;

public override int GetBookingNb()

{

return this.bookingNb;

}

// the booking's customer.

private PersonComponent cust;

public override PersonComponent GetCustomer()

{

return this.cust;

}

// the booking start and end dates.

private DateTime arrival;

private DateTime departure;

public override void GetDates(out DateTime arrival,

out DateTime departure)

{

arrival = this.arrival;

departure = this.departure;

}

//the booking's list of guests.

private List<PersonComponent> guests = new List<PersonComponent>();

public override List<PersonComponent> GetGuests()

{

return this.guests;

}

// METHODS:

/\*

\* Constructor.

\*

\* throws ArgumentException if bookingNb is not strictly greater

\* than 0, if departure day is not strictly later than arrival

\* day, or if customer is not decorated as a customer.

\*/

public Booking(int bookingNb,

PersonComponent customer,

DateTime arrival,

DateTime departure)

{

if (customer.GetCustNb() <= 0)

{

throw new ArgumentException("Booking.cust must be"

+ " decorated as a Customer");

}

if (bookingNb <= 0)

{

throw new ArgumentException("Booking.bookingNb must be"

+ " strictly greater than 0");

}

if (departure.DayOfYear <= arrival.DayOfYear

&& departure.Year <= arrival.Year)

{

throw new ArgumentException("departure day must be strictly"

+ " later than arrival day");

}

this.bookingNb = bookingNb;

this.cust = customer;

this.arrival = arrival;

this.departure = departure;

}

/\*

\* True if the contents of both bookings are identical, otherwise

\* false

\*/

public override bool Equals(object obj)

{

// self check

if (this == obj) return true;

// null check

if (obj == null) return false;

// type check and cast

if (this.GetType() != obj.GetType()) return false;

Booking b = (Booking)obj;

// attribute content comparison

return this.bookingNb == b.bookingNb

&& this.cust == b.cust

// only checking reference equality because i am lacking

// the time to implement customer 'deep' equallity check

&& this.arrival == b.arrival

&& this.departure == b.departure;

}

/\*

\* Adds a guest to the booking.

\*

\* Throws ArgumentException if there is already 4 guests added to

\* the booking.

\*/

public override void AddGuest(PersonComponent guest)

{

if (guests.Count >= 4)

{

throw new ArgumentException("this booking already has"

+ " 4 guests");

}

this.guests.Add(guest);

}

/\*

\* Returns the number of guests included in this Booking.

\*/

public override int GetNbGuests()

{

return guests.Count;

}

/\*

\* Returns the number of nights booked.

\*/

public override int GetNbNights()

{

return (departure - arrival).Days;

}

/\*

\* Returns the cost for each individual night booked.

\*/

public override float GetCostPerNight()

{

float costPerNight = 0;

foreach (PersonComponent g in guests)

{

if (g.GetAge() < 18)

{

costPerNight += 30;

}

else

{

costPerNight += 50;

}

}

return costPerNight;

}

/\*

\* Returns a textual representation of the Booking in order

\* to persist it to a CSV file.

\*/

public override String ToCSV()

{

int custIndex = indexOfCustomer();

StringBuilder csvBooking = new StringBuilder("#BOOKING\r\n");

csvBooking.Append(bookingNb + ",");

csvBooking.Append(arrival.ToString().Substring(0, 10) + ",");

csvBooking.Append(departure.ToString().Substring(0, 10) + "\r\n");

if (custIndex >= 0)

{

csvBooking.Append(guests.ElementAt(custIndex).ToCSV());

}

else

{

csvBooking.Append(cust.ToCSV());

}

if (GetNbGuests() > 0)

{

foreach (Guest g in guests)

{

if (guests.IndexOf(g) != custIndex)

{

csvBooking.Append(g.ToCSV());

}

}

}

return csvBooking.ToString();

}

/\*

\* Returns the index of the element that is also a customer in

\* list of guests, or -1 if none is.

\*/

private int indexOfCustomer()

{

int i = -1;

foreach (Guest g in guests)

{

if (g.IsCustomer())

{

i = guests.IndexOf(g);

}

}

return i;

}

}

}

1. SourceCode: BookingDecorator.cs
2. SourceCode: Breakfast.cs
3. SourceCode: EveningMeal.cs
4. SourceCode: CarHire.cs
5. SourceCode: BookingFactory.cs
6. SourceCode: DataPersistenceFacade.cs
7. SourceCode: CSVWriter.cs
8. SourceCode: CSVReader.cs
9. SourceCode: PersonField.cs
10. SourceCode: CustomerField.cs
11. SourceCode: GuestField.cs
12. SourceCode: PersonFactoryField.cs
13. SourceCode: BookingField.cs
14. SourceCode: BreakfastField.cs
15. SourceCode: EveningMealField.cs
16. SourceCode: CarHireField.cs
17. SourceCode: BookingFactory.cs
18. SourceCode: ModelFacade.cs
19. SourceCode: MainWindow.xaml.cs
20. SourceCode: WindowLoadBooking.xaml.cs
21. SourceCode: WindowCreateEdit.xaml.cs
22. SourceCode: WindowCustomerDetails.xaml.cs
23. SourceCode: WindowGuestsDetails.xaml.cs
24. SourceCode: WindowBreakfastDetails.xaml.cs
25. SourceCode: WindowEveningMealDetails.xaml.cs
26. SourceCode: WindowCarHireDetails.xaml.cs
27. SourceCode: WindowInvoice.xaml.cs