

## Gestão de Alojamentos Turísticos (Testes)

Generated by Doxygen 1.10.0



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List	1
<b>2 Data Structure Index</b>	<b>3</b>
2.1 Data Structures	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Namespace Documentation</b>	<b>7</b>
4.1 SmartStay Namespace Reference	7
4.2 SmartStay.Tests Namespace Reference	7
<b>5 Data Structure Documentation</b>	<b>9</b>
5.1 SmartStay.Tests.ClientTests Class Reference	9
5.1.1 Detailed Description	10
5.1.2 Member Function Documentation	10
5.1.2.1 Client_FullData_CreatesClient()	10
5.1.2.2 Client_GenerateUniqueClientId_CreatesUniqueIds()	10
5.1.2.3 Client_InvalidAddress_ThrowsValidationException()	10
5.1.2.4 Client_InvalidEmail_ThrowsValidationException()	10
5.1.2.5 Client_InvalidPaymentMethod_ThrowsValidationException()	10
5.1.2.6 Client_InvalidPhoneNumber_ThrowsValidationException()	11
5.1.2.7 Client_SetValidAddress_UpdatesAddress()	11
5.1.2.8 Client_SetValidEmail_UpdatesEmail()	11
5.1.2.9 Client_SetValidFirstName_UpdatesFirstName()	11
5.1.2.10 Client_SetValidLastName_UpdatesLastName()	11
5.1.2.11 Client_SetValidPaymentMethod_UpdatesPaymentMethod()	11
5.1.2.12 Client_SetValidPhoneNumber_UpdatesPhoneNumber()	12
5.1.2.13 Client_ToString_ReturnsValidJson()	12
5.1.2.14 Client_ValidData_CreatesClient()	12
5.2 SmartStay.Tests.ValidatorTests Class Reference	12
5.2.1 Detailed Description	14
5.2.2 Member Function Documentation	14
5.2.2.1 ValidateAccommodationName_InvalidName_ThrowsException()	14
5.2.2.2 ValidateAccommodationName_ValidName_ReturnsName()	14
5.2.2.3 ValidateAccommodationType_InvalidType_ThrowsException()	15
5.2.2.4 ValidateAccommodationType_ValidType_ReturnsType()	15
5.2.2.5 ValidateAddress_InvalidAddress_ThrowsException()	15
5.2.2.6 ValidateAddress_ValidAddress_ReturnsAddress()	15
5.2.2.7 ValidateCheckInDate_PastDate_ThrowsException()	16
5.2.2.8 ValidateCheckInDate_ValidFutureDate_ReturnsDate()	16
5.2.2.9 ValidateCheckOutDate_InvalidDateRange_ThrowsException()	16
5.2.2.10 ValidateCheckOutDate_ValidDateRange_ReturnsCheckOutDate()	16

5.2.2.11 ValidateEmail_InvalidEmail_ThrowsException()	16
5.2.2.12 ValidateEmail_ValidEmail_ReturnsEmail()	17
5.2.2.13 ValidateName_InvalidName_ThrowsException()	17
5.2.2.14 ValidateName_ValidName_ReturnsName()	17
5.2.2.15 ValidatePayment_InvalidPayment_ThrowsException()	17
5.2.2.16 ValidatePayment_ValidPayment_ReturnsPayment()	18
5.2.2.17 ValidatePaymentAmount_InvalidAmount_ThrowsException()	18
5.2.2.18 ValidatePaymentAmount_ValidAmount_ReturnsAmount()	18
5.2.2.19 ValidatePaymentStatus_InvalidStatus_ThrowsException()	18
5.2.2.20 ValidatePaymentStatus_ValidStatus_ReturnsStatus()	18
5.2.2.21 ValidatePhoneNumber_InvalidPhoneNumber_ThrowsException()	18
5.2.2.22 ValidatePhoneNumber_ValidPhoneNumber_ReturnsPhoneNumber()	19
5.2.2.23 ValidatePrice_InvalidPrice_ThrowsException()	19
5.2.2.24 ValidatePrice_ValidPrice_ReturnsPrice()	19
5.2.2.25 ValidateReservationStatus_InvalidStatus_ThrowsException()	19
5.2.2.26 ValidateReservationStatus_ValidStatus_ReturnsStatus()	20
5.2.2.27 ValidateTotalCost_InvalidCost_ThrowsException()	20
5.2.2.28 ValidateTotalCost_ValidCost_ReturnsTotalCost()	20
<b>6 File Documentation</b>	<b>21</b>
6.1 AccommodationTests.cs File Reference	21
6.2 AccommodationTests.cs	21
6.3 ClientTests.cs File Reference	21
6.4 ClientTests.cs	21
6.5 ReservationTests.cs File Reference	24
6.6 ReservationTests.cs	24
6.7 .NETCoreApp,Version=v8.0.AssemblyAttributes.cs File Reference	24
6.8 .NETCoreApp,Version=v8.0.AssemblyAttributes.cs	24
6.9 SmartStay.Tests.AssemblyInfo.cs File Reference	24
6.10 SmartStay.Tests.AssemblyInfo.cs	24
6.11 SmartStay.Tests.GlobalUsings.g.cs File Reference	25
6.12 SmartStay.Tests.GlobalUsings.g.cs	25
6.13 ValidationTests.cs File Reference	25
6.14 ValidationTests.cs	25
<b>Index</b>	<b>31</b>

# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">SmartStay</a> . . . . .	<a href="#">7</a>
<a href="#">SmartStay.Tests</a> . . . . .	<a href="#">7</a>



## Chapter 2

# Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">SmartStay.Tests.ClientTests</a> . . . . .	9
<a href="#">SmartStay.Tests.ValidatorTests</a>	
Defines the ValidatorTests class which tests the validation logic of various input parameters such as names, email addresses, accommodation types, prices, payment details, and more . . . . .	12



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">AccommodationTests.cs</a>	21
<a href="#">ClientTests.cs</a>	21
<a href="#">ReservationTests.cs</a>	24
<a href="#">.NETCoreApp,Version=v8.0.AssemblyAttributes.cs</a>	24
<a href="#">SmartStay.Tests.AssemblyInfo.cs</a>	24
<a href="#">SmartStay.Tests.GlobalUsings.g.cs</a>	25
<a href="#">ValidationTests.cs</a>	25



## Chapter 4

# Namespace Documentation

### 4.1 SmartStay Namespace Reference

#### Namespaces

- namespace [Tests](#)

### 4.2 SmartStay.Tests Namespace Reference

#### Data Structures

- class [ClientTests](#)
- class [ValidatorTests](#)

*Defines the ValidatorTests class which tests the validation logic of various input parameters such as names, email addresses, accommodation types, prices, payment details, and more.*



## Chapter 5

# Data Structure Documentation

### 5.1 SmartStay.Tests.ClientTests Class Reference

#### Public Member Functions

- void [Client\\_ValidData\\_CreatesClient](#) ()  
*Tests the constructor of the Client class when valid data is provided. Ensures the client is created successfully with the given information.*
- void [Client\\_FullData\\_CreatesClient](#) ()  
*Tests the constructor of the Client class when all optional parameters are provided. Ensures the client is created successfully with full details.*
- void [Client\\_InvalidEmail\\_ThrowsValidationException](#) ()  
*Tests the constructor of the Client class when invalid email is provided. Ensures a ValidationException is thrown.*
- void [Client\\_InvalidPhoneNumber\\_ThrowsValidationException](#) ()  
*Tests the constructor of the Client class when invalid phone number is provided. Ensures a ValidationException is thrown.*
- void [Client\\_InvalidAddress\\_ThrowsValidationException](#) ()  
*Tests the constructor of the Client class when invalid address is provided. Ensures a ValidationException is thrown.*
- void [Client\\_InvalidPaymentMethod\\_ThrowsValidationException](#) ()  
*Tests the constructor of the Client class when an invalid payment method is provided. Ensures a ValidationException is thrown.*
- void [Client\\_SetValidFirstName\\_UpdatesFirstName](#) ()  
*Tests the property setter and getter for FirstName. Ensures that a valid name can be set and retrieved correctly.*
- void [Client\\_SetValidLastName\\_UpdatesLastName](#) ()  
*Tests the property setter and getter for LastName. Ensures that a valid last name can be set and retrieved correctly.*
- void [Client\\_SetValidEmail\\_UpdatesEmail](#) ()  
*Tests the property setter and getter for Email. Ensures that a valid email can be set and retrieved correctly.*
- void [Client\\_SetValidPhoneNumber\\_UpdatesPhoneNumber](#) ()  
*Tests the property setter and getter for PhoneNumber. Ensures that a valid phone number can be set and retrieved correctly.*
- void [Client\\_SetValidAddress\\_UpdatesAddress](#) ()  
*Tests the property setter and getter for Address. Ensures that a valid address can be set and retrieved correctly.*
- void [Client\\_SetValidPaymentMethod\\_UpdatesPaymentMethod](#) ()  
*Tests the property setter and getter for PreferredPaymentMethod. Ensures that a valid payment method can be set and retrieved correctly.*
- void [Client\\_GenerateUniqueClientId\\_CreatesUniquelds](#) ()  
*Tests the client ID generation to ensure it increments properly for multiple clients. Ensures that each client has a unique ID.*
- void [Client\\_ToString\\_ReturnsValidJson](#) ()  
*Tests the ToString method of the Client class. Ensures the client object is serialized to a JSON string with proper formatting.*

### 5.1.1 Detailed Description

Definition at line 11 of file [ClientTests.cs](#).

### 5.1.2 Member Function Documentation

#### 5.1.2.1 Client\_FullData\_CreatesClient()

```
void SmartStay.Tests.ClientTests.Client_FullData_CreatesClient ( ) [inline]
```

Tests the constructor of the Client class when all optional parameters are provided. Ensures the client is created successfully with full details.

Definition at line 36 of file [ClientTests.cs](#).

#### 5.1.2.2 Client\_GenerateUniqueClientId\_CreatesUniqueIds()

```
void SmartStay.Tests.ClientTests.Client_GenerateUniqueClientId_CreatesUniqueIds ( ) [inline]
```

Tests the client ID generation to ensure it increments properly for multiple clients. Ensures that each client has a unique ID.

Definition at line 246 of file [ClientTests.cs](#).

#### 5.1.2.3 Client\_InvalidAddress\_ThrowsValidationException()

```
void SmartStay.Tests.ClientTests.Client_InvalidAddress_ThrowsValidationException ( ) [inline]
```

Tests the constructor of the Client class when invalid address is provided. Ensures a ValidationException is thrown.

Definition at line 94 of file [ClientTests.cs](#).

#### 5.1.2.4 Client\_InvalidEmail\_ThrowsValidationException()

```
void SmartStay.Tests.ClientTests.Client_InvalidEmail_ThrowsValidationException ( ) [inline]
```

Tests the constructor of the Client class when invalid email is provided. Ensures a ValidationException is thrown.

Definition at line 61 of file [ClientTests.cs](#).

#### 5.1.2.5 Client\_InvalidPaymentMethod\_ThrowsValidationException()

```
void SmartStay.Tests.ClientTests.Client_InvalidPaymentMethod_ThrowsValidationException ( )  
[inline]
```

Tests the constructor of the Client class when an invalid payment method is provided. Ensures a ValidationException is thrown.

Definition at line 112 of file [ClientTests.cs](#).

#### 5.1.2.6 Client\_InvalidPhoneNumber\_ThrowsValidationException()

```
void SmartStay.Tests.ClientTests.Client_InvalidPhoneNumber_ThrowsValidationException ( ) [inline]
```

Tests the constructor of the Client class when invalid phone number is provided. Ensures a ValidationException is thrown.

Definition at line 77 of file [ClientTests.cs](#).

#### 5.1.2.7 Client\_SetValidAddress\_UpdatesAddress()

```
void SmartStay.Tests.ClientTests.Client_SetValidAddress_UpdatesAddress ( ) [inline]
```

Tests the property setter and getter for Address. Ensures that a valid address can be set and retrieved correctly.

Definition at line 204 of file [ClientTests.cs](#).

#### 5.1.2.8 Client\_SetValidEmail\_UpdatesEmail()

```
void SmartStay.Tests.ClientTests.Client_SetValidEmail_UpdatesEmail ( ) [inline]
```

Tests the property setter and getter for Email. Ensures that a valid email can be set and retrieved correctly.

Definition at line 167 of file [ClientTests.cs](#).

#### 5.1.2.9 Client\_SetValidFirstName\_UpdatesFirstName()

```
void SmartStay.Tests.ClientTests.Client_SetValidFirstName_UpdatesFirstName ( ) [inline]
```

Tests the property setter and getter for FirstName. Ensures that a valid name can be set and retrieved correctly.

Definition at line 131 of file [ClientTests.cs](#).

#### 5.1.2.10 Client\_SetValidLastName\_UpdatesLastName()

```
void SmartStay.Tests.ClientTests.Client_SetValidLastName_UpdatesLastName ( ) [inline]
```

Tests the property setter and getter for LastName. Ensures that a valid last name can be set and retrieved correctly.

Definition at line 149 of file [ClientTests.cs](#).

#### 5.1.2.11 Client\_SetValidPaymentMethod\_UpdatesPaymentMethod()

```
void SmartStay.Tests.ClientTests.Client_SetValidPaymentMethod_UpdatesPaymentMethod ( ) [inline]
```

Tests the property setter and getter for PreferredPaymentMethod. Ensures that a valid payment method can be set and retrieved correctly.

Definition at line 224 of file [ClientTests.cs](#).

#### 5.1.2.12 Client\_SetValidPhoneNumber\_UpdatesPhoneNumber()

```
void SmartStay.Tests.ClientTests.Client_SetValidPhoneNumber_UpdatesPhoneNumber ( ) [inline]
```

Tests the property setter and getter for PhoneNumber. Ensures that a valid phone number can be set and retrieved correctly.

Definition at line 185 of file [ClientTests.cs](#).

#### 5.1.2.13 Client\_ToString\_ReturnsValidJson()

```
void SmartStay.Tests.ClientTests.Client_ToString_ReturnsValidJson ( ) [inline]
```

Tests the ToString method of the Client class. Ensures the client object is serialized to a JSON string with proper formatting.

Definition at line 259 of file [ClientTests.cs](#).

#### 5.1.2.14 Client\_ValidData\_CreatesClient()

```
void SmartStay.Tests.ClientTests.Client_ValidData_CreatesClient ( ) [inline]
```

Tests the constructor of the Client class when valid data is provided. Ensures the client is created successfully with the given information.

Definition at line 18 of file [ClientTests.cs](#).

The documentation for this class was generated from the following file:

- [ClientTests.cs](#)

## 5.2 SmartStay.Tests.ValidatorTests Class Reference

Defines the ValidatorTests class which tests the validation logic of various input parameters such as names, email addresses, accommodation types, prices, payment details, and more.

## Public Member Functions

- void [ValidateName\\_ValidName\\_ReturnsName](#) (string validName)  
*The ValidateName\_ValidName\_ReturnsName Tests the ValidateName method with a valid name input and ensures it returns the same name.*
- void [ValidateName\\_InvalidName\\_ThrowsException](#) (string invalidName)  
*The ValidateName\_InvalidName\_ThrowsException Tests the ValidateName method with invalid name inputs (empty or null) and ensures a ValidationException is thrown.*
- void [ValidateAccommodationName\\_ValidName\\_ReturnsName](#) (string validAccommodationName)  
*The ValidateAccommodationName\_ValidName\_ReturnsName Tests the ValidateAccommodationName method with a valid accommodation name and ensures it returns the same name.*
- void [ValidateAccommodationName\\_InvalidName\\_ThrowsException](#) (string invalidAccommodationName)  
*The ValidateAccommodationName\_InvalidName\_ThrowsException Tests the ValidateAccommodationName method with invalid accommodation names (empty or null) and ensures a ValidationException is thrown.*
- void [ValidateEmail\\_ValidEmail\\_ReturnsEmail](#) (string validEmail)  
*The ValidateEmail\_ValidEmail\_ReturnsEmail Tests the ValidateEmail method with a valid email input and ensures it returns the same email.*
- void [ValidateEmail\\_InvalidEmail\\_ThrowsException](#) (string invalidEmail)  
*The ValidateEmail\_InvalidEmail\_ThrowsException Tests the ValidateEmail method with invalid email inputs (e.g., an email missing '@') and ensures a ValidationException is thrown.*
- void [ValidatePhoneNumber\\_ValidPhoneNumber\\_ReturnsPhoneNumber](#) (string validPhoneNumber)  
*The ValidatePhoneNumber\_ValidPhoneNumber\_ReturnsPhoneNumber Tests the ValidatePhoneNumber method with a valid phone number input and ensures it returns the same phone number.*
- void [ValidatePhoneNumber\\_InvalidPhoneNumber\\_ThrowsException](#) (string invalidPhoneNumber)  
*The ValidatePhoneNumber\_InvalidPhoneNumber\_ThrowsException Tests the ValidatePhoneNumber method with an invalid phone number input and ensures a ValidationException is thrown.*
- void [ValidateAddress\\_ValidAddress\\_ReturnsAddress](#) (string validAddress)  
*The ValidateAddress\_ValidAddress\_ReturnsAddress Tests the ValidateAddress method with a valid address and ensures it returns the same address.*
- void [ValidateAddress\\_InvalidAddress\\_ThrowsException](#) (string invalidAddress)  
*The ValidateAddress\_InvalidAddress\_ThrowsException Tests the ValidateAddress method with an invalid address (e.g., empty) and ensures a ValidationException is thrown.*
- void [ValidatePrice\\_ValidPrice\\_ReturnsPrice](#) ()  
*The ValidatePrice\_ValidPrice\_ReturnsPrice Tests the ValidatePrice method with a valid price input and ensures it returns the same price.*
- void [ValidatePrice\\_InvalidPrice\\_ThrowsException](#) ()  
*The ValidatePrice\_InvalidPrice\_ThrowsException Tests the ValidatePrice method with an invalid price input (e.g., zero or negative) and ensures a ValidationException is thrown.*
- void [ValidatePaymentAmount\\_ValidAmount\\_ReturnsAmount](#) ()  
*The ValidatePaymentAmount\_ValidAmount\_ReturnsAmount Tests the ValidatePaymentAmount method with a valid payment amount input and ensures it returns the same amount.*
- void [ValidatePaymentAmount\\_InvalidAmount\\_ThrowsException](#) ()  
*The ValidatePaymentAmount\_InvalidAmount\_ThrowsException Tests the ValidatePaymentAmount method with an invalid payment amount input (e.g., negative amount) and ensures a ValidationException is thrown.*
- void [ValidatePaymentStatus\\_ValidStatus\\_ReturnsStatus](#) ()  
*Validates that the payment status is correctly processed when a valid status is provided.*
- void [ValidatePaymentStatus\\_InvalidStatus\\_ThrowsException](#) ()  
*Validates that an exception is thrown when an invalid payment status (out of the predefined enum) is provided.*
- void [ValidateAccommodationType\\_ValidType\\_ReturnsType](#) ()  
*Validates that the accommodation type is correctly processed when a valid type is provided.*
- void [ValidateAccommodationType\\_InvalidType\\_ThrowsException](#) ()  
*Validates that an exception is thrown when an invalid accommodation type (out of the predefined enum) is provided.*
- void [ValidateCheckInDate\\_ValidFutureDate\\_ReturnsDate](#) ()  
*Validates that the check-in date is processed correctly when a future date is provided.*

- void [ValidateCheckInDate\\_PastDate\\_ThrowsException](#) ()  
*Validates that an exception is thrown when a past date is provided as the check-in date.*
- void [ValidateCheckOutDate\\_ValidDateRange\\_ReturnsCheckOutDate](#) ()  
*Validates that the check-out date is processed correctly when it is after the check-in date.*
- void [ValidateCheckOutDate\\_InvalidDateRange\\_ThrowsException](#) ()  
*Validates that an exception is thrown when the check-out date is before the check-in date.*
- void [ValidateTotalCost\\_ValidCost\\_ReturnsTotalCost](#) ()  
*Validates that the total cost is correctly processed when a valid cost is provided.*
- void [ValidateTotalCost\\_InvalidCost\\_ThrowsException](#) ()  
*Validates that an exception is thrown when a negative cost is provided.*
- void [ValidatePayment\\_ValidPayment\\_ReturnsPayment](#) ()  
*Validates that the payment amount is correctly processed when a valid payment is provided.*
- void [ValidatePayment\\_InvalidPayment\\_ThrowsException](#) ()  
*Validates that an exception is thrown when a negative payment value is provided.*
- void [ValidateReservationStatus\\_ValidStatus\\_ReturnsStatus](#) ()  
*Validates that the reservation status is correctly processed when a valid status is provided.*
- void [ValidateReservationStatus\\_InvalidStatus\\_ThrowsException](#) ()  
*Validates that an exception is thrown when an invalid reservation status (out of the predefined enum) is provided.*

## 5.2.1 Detailed Description

Defines the ValidatorTests class which tests the validation logic of various input parameters such as names, email addresses, accommodation types, prices, payment details, and more.

Definition at line 32 of file [ValidationTests.cs](#).

## 5.2.2 Member Function Documentation

### 5.2.2.1 ValidateAccommodationName\_InvalidName\_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateAccommodationName_InvalidName_ThrowsException (
    string invalidAccommodationName ) [inline]
```

The `ValidateAccommodationName_InvalidName_ThrowsException` Tests the `ValidateAccommodationName` method with invalid accommodation names (empty or null) and ensures a `ValidationException` is thrown.

#### Parameters

<code>invalidAccommodationName</code>	The invalidAccommodationNamestring
---------------------------------------	------------------------------------

Definition at line 90 of file [ValidationTests.cs](#).

### 5.2.2.2 ValidateAccommodationName\_ValidName\_ReturnsName()

```
void SmartStay.Tests.ValidatorTests.ValidateAccommodationName_ValidName_ReturnsName (
    string validAccommodationName ) [inline]
```

The `ValidateAccommodationName_ValidName_ReturnsName` Tests the `ValidateAccommodationName` method with a valid accommodation name and ensures it returns the same name.

## Parameters

<i>validAccommodationName</i>	The validAccommodationNamestring
-------------------------------	----------------------------------

Definition at line 75 of file [ValidationTests.cs](#).

### 5.2.2.3 ValidateAccommodationType\_InvalidType\_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateAccommodationType_InvalidType_ThrowsException ( )  
[inline]
```

Validates that an exception is thrown when an invalid accommodation type (out of the predefined enum) is provided.

Definition at line 298 of file [ValidationTests.cs](#).

### 5.2.2.4 ValidateAccommodationType\_ValidType\_ReturnsType()

```
void SmartStay.Tests.ValidatorTests.ValidateAccommodationType_ValidType_ReturnsType ( ) [inline]
```

Validates that the accommodation type is correctly processed when a valid type is provided.

Definition at line 286 of file [ValidationTests.cs](#).

### 5.2.2.5 ValidateAddress\_InvalidAddress\_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateAddress_InvalidAddress_ThrowsException (   
    string invalidAddress ) [inline]
```

The ValidateAddress\_InvalidAddress\_ThrowsException Tests the ValidateAddress method with an invalid address (e.g., empty) and ensures a ValidationException is thrown.

## Parameters

<i>invalidAddress</i>	The invalidAddressstring
-----------------------	--------------------------

Definition at line 186 of file [ValidationTests.cs](#).

### 5.2.2.6 ValidateAddress\_ValidAddress\_ReturnsAddress()

```
void SmartStay.Tests.ValidatorTests.ValidateAddress_ValidAddress_ReturnsAddress (   
    string validAddress ) [inline]
```

The ValidateAddress\_ValidAddress\_ReturnsAddress Tests the ValidateAddress method with a valid address and ensures it returns the same address.

## Parameters

<i>validAddress</i>	The validAddressstring
---------------------	------------------------

Definition at line 172 of file [ValidationTests.cs](#).

#### 5.2.2.7 ValidateCheckInDate\_PastDate\_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateCheckInDate_PastDate_ThrowsException ( ) [inline]
```

Validates that an exception is thrown when a past date is provided as the check-in date.

Definition at line 325 of file [ValidationTests.cs](#).

#### 5.2.2.8 ValidateCheckInDate\_ValidFutureDate\_ReturnsDate()

```
void SmartStay.Tests.ValidatorTests.ValidateCheckInDate_ValidFutureDate_ReturnsDate ( ) [inline]
```

Validates that the check-in date is processed correctly when a future date is provided.

Definition at line 314 of file [ValidationTests.cs](#).

#### 5.2.2.9 ValidateCheckOutDate\_InvalidDateRange\_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateCheckOutDate_InvalidDateRange_ThrowsException ( )  
[inline]
```

Validates that an exception is thrown when the check-out date is before the check-in date.

Definition at line 352 of file [ValidationTests.cs](#).

#### 5.2.2.10 ValidateCheckOutDate\_ValidDateRange\_ReturnsCheckOutDate()

```
void SmartStay.Tests.ValidatorTests.ValidateCheckOutDate_ValidDateRange_ReturnsCheckOutDate ( )  
[inline]
```

Validates that the check-out date is processed correctly when it is after the check-in date.

Definition at line 340 of file [ValidationTests.cs](#).

#### 5.2.2.11 ValidateEmail\_InvalidEmail\_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateEmail_InvalidEmail_ThrowsException (   
    string invalidEmail ) [inline]
```

The ValidateEmail\_InvalidEmail\_ThrowsException Tests the ValidateEmail method with invalid email inputs (e.g., an email missing '@') and ensures a ValidationException is thrown.

##### Parameters

<i>invalidEmail</i>	The invalidEmailstring
---------------------	------------------------

Definition at line 122 of file [ValidationTests.cs](#).

#### 5.2.2.12 ValidateEmail\_ValidEmail\_ReturnsEmail()

```
void SmartStay.Tests.ValidatorTests.ValidateEmail_ValidEmail_ReturnsEmail (
    string validEmail ) [inline]
```

The ValidateEmail\_ValidEmail\_ReturnsEmail Tests the ValidateEmail method with a valid email input and ensures it returns the same email.

##### Parameters

<i>validEmail</i>	The validEmailstring
-------------------	----------------------

Definition at line 108 of file [ValidationTests.cs](#).

#### 5.2.2.13 ValidateName\_InvalidName\_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateName_InvalidName_ThrowsException (
    string invalidName ) [inline]
```

The ValidateName\_InvalidName\_ThrowsException Tests the ValidateName method with invalid name inputs (empty or null) and ensures a ValidationException is thrown.

##### Parameters

<i>invalidName</i>	The invalidNamestring
--------------------	-----------------------

Definition at line 58 of file [ValidationTests.cs](#).

#### 5.2.2.14 ValidateName\_ValidName\_ReturnsName()

```
void SmartStay.Tests.ValidatorTests.ValidateName_ValidName_ReturnsName (
    string validName ) [inline]
```

The ValidateName\_ValidName\_ReturnsName Tests the ValidateName method with a valid name input and ensures it returns the same name.

##### Parameters

<i>validName</i>	The validNamestring
------------------	---------------------

Definition at line 43 of file [ValidationTests.cs](#).

#### 5.2.2.15 ValidatePayment\_InvalidPayment\_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidatePayment_InvalidPayment_ThrowsException ( ) [inline]
```

Validates that an exception is thrown when a negative payment value is provided.

Definition at line 406 of file [ValidationTests.cs](#).

#### 5.2.2.16 ValidatePayment\_ValidPayment\_ReturnsPayment()

```
void SmartStay.Tests.ValidatorTests.ValidatePayment_ValidPayment_ReturnsPayment ( ) [inline]
```

Validates that the payment amount is correctly processed when a valid payment is provided.

Definition at line 395 of file [ValidationTests.cs](#).

#### 5.2.2.17 ValidatePaymentAmount\_InvalidAmount\_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidatePaymentAmount_InvalidAmount_ThrowsException ( )  
[inline]
```

The ValidatePaymentAmount\_InvalidAmount\_ThrowsException Tests the ValidatePaymentAmount method with an invalid payment amount input (e.g., negative amount) and ensures a ValidationException is thrown.

Definition at line 243 of file [ValidationTests.cs](#).

#### 5.2.2.18 ValidatePaymentAmount\_ValidAmount\_ReturnsAmount()

```
void SmartStay.Tests.ValidatorTests.ValidatePaymentAmount_ValidAmount_ReturnsAmount ( ) [inline]
```

The ValidatePaymentAmount\_ValidAmount\_ReturnsAmount Tests the ValidatePaymentAmount method with a valid payment amount input and ensures it returns the same amount.

Definition at line 230 of file [ValidationTests.cs](#).

#### 5.2.2.19 ValidatePaymentStatus\_InvalidStatus\_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidatePaymentStatus_InvalidStatus_ThrowsException ( )  
[inline]
```

Validates that an exception is thrown when an invalid payment status (out of the predefined enum) is provided.

Definition at line 270 of file [ValidationTests.cs](#).

#### 5.2.2.20 ValidatePaymentStatus\_ValidStatus\_ReturnsStatus()

```
void SmartStay.Tests.ValidatorTests.ValidatePaymentStatus_ValidStatus_ReturnsStatus ( ) [inline]
```

Validates that the payment status is correctly processed when a valid status is provided.

Definition at line 259 of file [ValidationTests.cs](#).

#### 5.2.2.21 ValidatePhoneNumber\_InvalidPhoneNumber\_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidatePhoneNumber_InvalidPhoneNumber_ThrowsException (   
    string invalidPhoneNumber ) [inline]
```

The ValidatePhoneNumber\_InvalidPhoneNumber\_ThrowsException Tests the ValidatePhoneNumber method with an invalid phone number input and ensures a ValidationException is thrown.

## Parameters

<i>invalidPhoneNumber</i>	The invalidPhoneNumberstring
---------------------------	------------------------------

Definition at line 154 of file [ValidationTests.cs](#).

**5.2.2.22 ValidatePhoneNumber\_ValidPhoneNumber\_ReturnsPhoneNumber()**

```
void SmartStay.Tests.ValidatorTests.ValidatePhoneNumber_ValidPhoneNumber_ReturnsPhoneNumber (
    string validPhoneNumber ) [inline]
```

The ValidatePhoneNumber\_ValidPhoneNumber\_ReturnsPhoneNumber Tests the ValidatePhoneNumber method with a valid phone number input and ensures it returns the same phone number.

## Parameters

<i>validPhoneNumber</i>	The validPhoneNumberstring
-------------------------	----------------------------

Definition at line 140 of file [ValidationTests.cs](#).

**5.2.2.23 ValidatePrice\_InvalidPrice\_ThrowsException()**

```
void SmartStay.Tests.ValidatorTests.ValidatePrice_InvalidPrice_ThrowsException ( ) [inline]
```

The ValidatePrice\_InvalidPrice\_ThrowsException Tests the ValidatePrice method with an invalid price input (e.g., zero or negative) and ensures a ValidationException is thrown.

Definition at line 214 of file [ValidationTests.cs](#).

**5.2.2.24 ValidatePrice\_ValidPrice\_ReturnsPrice()**

```
void SmartStay.Tests.ValidatorTests.ValidatePrice_ValidPrice_ReturnsPrice ( ) [inline]
```

The ValidatePrice\_ValidPrice\_ReturnsPrice Tests the ValidatePrice method with a valid price input and ensures it returns the same price.

Definition at line 201 of file [ValidationTests.cs](#).

**5.2.2.25 ValidateReservationStatus\_InvalidStatus\_ThrowsException()**

```
void SmartStay.Tests.ValidatorTests.ValidateReservationStatus_InvalidStatus_ThrowsException (
) [inline]
```

Validates that an exception is thrown when an invalid reservation status (out of the predefined enum) is provided.

Definition at line 433 of file [ValidationTests.cs](#).

#### 5.2.2.26 ValidateReservationStatus\_ValidStatus\_ReturnsStatus()

```
void SmartStay.Tests.ValidatorTests.ValidateReservationStatus_ValidStatus_ReturnsStatus ( )  
[inline]
```

Validates that the reservation status is correctly processed when a valid status is provided.

Definition at line 421 of file [ValidationTests.cs](#).

#### 5.2.2.27 ValidateTotalCost\_InvalidCost\_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateTotalCost_InvalidCost_ThrowsException ( ) [inline]
```

Validates that an exception is thrown when a negative cost is provided.

Definition at line 380 of file [ValidationTests.cs](#).

#### 5.2.2.28 ValidateTotalCost\_ValidCost\_ReturnsTotalCost()

```
void SmartStay.Tests.ValidatorTests.ValidateTotalCost_ValidCost_ReturnsTotalCost ( ) [inline]
```

Validates that the total cost is correctly processed when a valid cost is provided.

Definition at line 369 of file [ValidationTests.cs](#).

The documentation for this class was generated from the following file:

- [ValidationTests.cs](#)

## Chapter 6

# File Documentation

### 6.1 AccommodationTests.cs File Reference

### 6.2 AccommodationTests.cs

[Go to the documentation of this file.](#)

00001

### 6.3 ClientTests.cs File Reference

#### Data Structures

- class [SmartStay.Tests.ClientTests](#)

#### Namespaces

- namespace [SmartStay](#)
- namespace [SmartStay.Tests](#)

### 6.4 ClientTests.cs

[Go to the documentation of this file.](#)

```
00001 using Microsoft.VisualStudio.TestTools.UnitTesting;
00002 using SmartStay.Models;
00003 using SmartStay.Models.Enums;
00004 using SmartStay.Services;
00005 using SmartStay.Validation;
00006 using System;
00007
00008 namespace SmartStay.Tests
00009 {
00010     [TestClass]
00011     public class ClientTests
00012     {
00017         [TestMethod]
00018         public void Client_ValidData_CreatesClient ()
00019         {
00020             var firstName = "John";
```

```

00021         var lastName = "Doe";
00022         var email = "john.doe@example.com";
00023         var client = new Client(firstName, lastName, email);
00024
00025         Assert.IsNotNull(client);
00026         Assert.AreEqual(firstName, client.FirstName);
00027         Assert.AreEqual(lastName, client.LastName);
00028         Assert.AreEqual(email, client.Email);
00029     }
00030
00031     [TestMethod]
00032     public void Client_FullData_CreatesClient()
00033     {
00034         var firstName = "Jane";
00035         var lastName = "Smith";
00036         var email = "jane.smith@example.com";
00037         var phoneNumber = "+351999999999";
00038         var address = "123 Elm St, Springfield";
00039         var preferredPaymentMethod = PaymentMethod.BankTransfer;
00040
00041         var client = new Client(firstName, lastName, email, phoneNumber, address,
00042             preferredPaymentMethod);
00043
00044         Assert.IsNotNull(client);
00045         Assert.AreEqual(firstName, client.FirstName);
00046         Assert.AreEqual(lastName, client.LastName);
00047         Assert.AreEqual(email, client.Email);
00048         Assert.AreEqual(phoneNumber, client.PhoneNumber);
00049         Assert.AreEqual(address, client.Address);
00050         Assert.AreEqual(preferredPaymentMethod, client.PreferredPaymentMethod);
00051     }
00052
00053     [TestMethod]
00054     public void Client_InvalidEmail_ThrowsValidationException()
00055     {
00056         var firstName = "John";
00057         var lastName = "Doe";
00058         var invalidEmail = "invalid-email";
00059
00060         var exception =
00061             Assert.ThrowsException<ValidationException>(() => new Client(firstName, lastName,
00062                 invalidEmail));
00063         Assert.AreEqual(ValidationErrorCode.InvalidEmail, exception.ErrorCode);
00064     }
00065
00066     [TestMethod]
00067     public void Client_InvalidPhoneNumber_ThrowsValidationException()
00068     {
00069         var firstName = "John";
00070         var lastName = "Doe";
00071         var email = "john.doe@example.com";
00072         var invalidPhoneNumber = "invalid-phone";
00073
00074         var exception = Assert.ThrowsException<ValidationException>(
00075             () => new Client(firstName, lastName, email, invalidPhoneNumber, "123 Elm St"));
00076         Assert.AreEqual(ValidationErrorCode.InvalidPhoneNumber, exception.ErrorCode);
00077     }
00078
00079     [TestMethod]
00080     public void Client_InvalidAddress_ThrowsValidationException()
00081     {
00082         var firstName = "John";
00083         var lastName = "Doe";
00084         var email = "john.doe@example.com";
00085         var phoneNumber = "+351999999999";
00086         var invalidAddress = "";
00087
00088         var exception = Assert.ThrowsException<ValidationException>(
00089             () => new Client(firstName, lastName, email, phoneNumber, invalidAddress));
00090         Assert.AreEqual(ValidationErrorCode.InvalidAddress, exception.ErrorCode);
00091     }
00092
00093     [TestMethod]
00094     public void Client_InvalidPaymentMethod_ThrowsValidationException()
00095     {
00096         var firstName = "Jane";
00097         var lastName = "Smith";
00098         var email = "jane.smith@example.com";
00099         var phoneNumber = "+351999999999";
00100         var address = "123 Elm St, Springfield";
00101         var invalidPaymentMethod = (PaymentMethod)999; // Invalid payment method
00102
00103         var exception = Assert.ThrowsException<ValidationException>(
00104             () => new Client(firstName, lastName, email, phoneNumber, address, invalidPaymentMethod));
00105         Assert.AreEqual(ValidationErrorCode.InvalidPaymentMethod, exception.ErrorCode);
00106     }
00107
00108     [TestMethod]
00109     public void Client_InvalidPaymentMethod_ThrowsValidationException()
00110     {
00111         var firstName = "Jane";
00112         var lastName = "Smith";
00113         var email = "jane.smith@example.com";
00114         var phoneNumber = "+351999999999";
00115         var address = "123 Elm St, Springfield";
00116         var invalidPaymentMethod = (PaymentMethod)999; // Invalid payment method
00117
00118         var exception = Assert.ThrowsException<ValidationException>(
00119             () => new Client(firstName, lastName, email, phoneNumber, address, invalidPaymentMethod));
00120         Assert.AreEqual(ValidationErrorCode.InvalidPaymentMethod, exception.ErrorCode);
00121     }
00122
00123     [TestMethod]
00124     public void Client_InvalidPaymentMethod_ThrowsValidationException()
00125     {
00126         var firstName = "Jane";
00127         var lastName = "Smith";
00128         var email = "jane.smith@example.com";
00129         var phoneNumber = "+351999999999";
00130         var address = "123 Elm St, Springfield";
00131         var invalidPaymentMethod = (PaymentMethod)999; // Invalid payment method
00132
00133         var exception = Assert.ThrowsException<ValidationException>(
00134             () => new Client(firstName, lastName, email, phoneNumber, address, invalidPaymentMethod));
00135         Assert.AreEqual(ValidationErrorCode.InvalidPaymentMethod, exception.ErrorCode);
00136     }

```

```
00130     [TestMethod]
00131     public void Client_SetValidFirstName_UpdatesFirstName()
00132     {
00133         var firstName = "John";
00134         var lastName = "Doe";
00135         var email = "john.doe@example.com";
00136         var client = new Client(firstName, lastName, email);
00137
00138         var newFirstName = "Johnny";
00139         client.FirstName = newFirstName;
00140
00141         Assert.AreEqual(newFirstName, client.FirstName);
00142     }
00143
00144     [TestMethod]
00145     public void Client_SetValidLastName_UpdatesLastName()
00146     {
00147         var firstName = "John";
00148         var lastName = "Doe";
00149         var email = "john.doe@example.com";
00150         var client = new Client(firstName, lastName, email);
00151
00152         var newLastName = "Smith";
00153         client.LastName = newLastName;
00154
00155         Assert.AreEqual(newLastName, client.LastName);
00156     }
00157
00158     [TestMethod]
00159     public void Client_SetValidEmail_UpdatesEmail()
00160     {
00161         var firstName = "John";
00162         var lastName = "Doe";
00163         var email = "john.doe@example.com";
00164         var client = new Client(firstName, lastName, email);
00165
00166         var newEmail = "johnny.doe@example.com";
00167         client.Email = newEmail;
00168
00169         Assert.AreEqual(newEmail, client.Email);
00170     }
00171
00172     [TestMethod]
00173     public void Client_SetValidPhoneNumber_UpdatesPhoneNumber()
00174     {
00175         var firstName = "John";
00176         var lastName = "Doe";
00177         var email = "john.doe@example.com";
00178         var phoneNumber = "+351999999999";
00179         var client = new Client(firstName, lastName, email, phoneNumber, "123 Elm St");
00180
00181         var newPhoneNumber = "+351888888888";
00182         client.PhoneNumber = newPhoneNumber;
00183
00184         Assert.AreEqual(newPhoneNumber, client.PhoneNumber);
00185     }
00186
00187     [TestMethod]
00188     public void Client_SetValidAddress_UpdatesAddress()
00189     {
00190         var firstName = "John";
00191         var lastName = "Doe";
00192         var email = "john.doe@example.com";
00193         var phoneNumber = "+351999999999";
00194         var address = "123 Elm St, Springfield";
00195         var client = new Client(firstName, lastName, email, phoneNumber, address);
00196
00197         var newAddress = "456 Oak St, Springfield";
00198         client.Address = newAddress;
00199
00200         Assert.AreEqual(newAddress, client.Address);
00201     }
00202
00203     [TestMethod]
00204     public void Client_SetValidPaymentMethod_UpdatesPaymentMethod()
00205     {
00206         var firstName = "John";
00207         var lastName = "Doe";
00208         var email = "john.doe@example.com";
00209         var phoneNumber = "+351999999999";
00210         var address = "123 Elm St, Springfield";
00211         var preferredPaymentMethod = PaymentMethod.PayPal;
00212         var client = new Client(firstName, lastName, email, phoneNumber, address,
00213                                 preferredPaymentMethod);
00214
00215         var newPaymentMethod = PaymentMethod.BankTransfer;
```

```

00236         client.PreferredPaymentMethod = newPaymentMethod;
00237
00238         Assert.AreEqual(newPaymentMethod, client.PreferredPaymentMethod);
00239     }
00240
00241     [TestMethod]
00242     public void Client_GenerateUniqueClientId_CreatesUniqueIds()
00243     {
00244         var firstClient = new Client("John", "Doe", "john.doe@example.com");
00245         var secondClient = new Client("Jane", "Smith", "jane.smith@example.com");
00246
00247         Assert.AreNotEqual(firstClient.Id, secondClient.Id);
00248     }
00249
00250     [TestMethod]
00251     public void Client_ToString_ReturnsValidJson()
00252     {
00253         var firstName = "John";
00254         var lastName = "Doe";
00255         var email = "john.doe@example.com";
00256         var client = new Client(firstName, lastName, email);
00257
00258         var json = client.ToString();
00259
00260         Assert.IsTrue(json.Contains("\"FirstName\": \"John\""));
00261         Assert.IsTrue(json.Contains("\"LastName\": \"Doe\""));
00262         Assert.IsTrue(json.Contains("\"Email\": \"john.doe@example.com\""));
00263     }
00264 }
00265
00266 }
00267
00268 }
00269
00270 }
00271
00272 }
00273 }

```

## 6.5 ReservationTests.cs File Reference

## 6.6 ReservationTests.cs

[Go to the documentation of this file.](#)

```
00001
```

## 6.7 .NETCoreApp,Version=v8.0.AssemblyAttributes.cs File Reference

## 6.8 .NETCoreApp,Version=v8.0.AssemblyAttributes.cs

[Go to the documentation of this file.](#)

```

00001 // <autogenerated />
00002 using System;
00003 using System.Reflection;
00004 [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0",
    FrameworkDisplayName = ".NET 8.0")]

```

## 6.9 SmartStay.Tests.AssemblyInfo.cs File Reference

## 6.10 SmartStay.Tests.AssemblyInfo.cs

[Go to the documentation of this file.](#)

```

00001 //-----
00002 // <auto-generated>
00003 //     This code was generated by a tool.
00004 //     Runtime Version:4.0.30319.42000
00005 //
00006 //     Changes to this file may cause incorrect behavior and will be lost if

```

```

00007 //      the code is regenerated.
00008 // </auto-generated>
00009 //-----
00010
00011 using System;
00012 using System.Reflection;
00013
00014 [assembly: System.Reflection.AssemblyCompanyAttribute("SmartStay.Tests")]
00015 [assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]
00016 [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]
00017 [assembly:
    System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+547a43e26c0e6800e3b3c5899fba97a9e932c12b")]
00018 [assembly: System.Reflection.AssemblyProductAttribute("SmartStay.Tests")]
00019 [assembly: System.Reflection.AssemblyTitleAttribute("SmartStay.Tests")]
00020 [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]
00021
00022 // Generated by the MSBuild WriteCodeFragment class.
00023

```

## 6.11 SmartStay.Tests.GlobalUsings.g.cs File Reference

## 6.12 SmartStay.Tests.GlobalUsings.g.cs

[Go to the documentation of this file.](#)

```

00001 // <auto-generated/>
00002 global using global::Microsoft.VisualStudio.TestTools.UnitTesting;
00003 global using global::System;
00004 global using global::System.Collections.Generic;
00005 global using global::System.IO;
00006 global using global::System.Linq;
00007 global using global::System.Net.Http;
00008 global using global::System.Threading;
00009 global using global::System.Threading.Tasks;

```

## 6.13 ValidationTests.cs File Reference

### Data Structures

- class [SmartStay.Tests.ValidatorTests](#)

*Defines the ValidatorTests class which tests the validation logic of various input parameters such as names, email addresses, accommodation types, prices, payment details, and more.*

### Namespaces

- namespace [SmartStay](#)
- namespace [SmartStay.Tests](#)

## 6.14 ValidationTests.cs

[Go to the documentation of this file.](#)

```

00001
00022 using SmartStay.Models.Enums;
00023 using SmartStay.Validation;
00024
00025 namespace SmartStay.Tests
00026 {
00031 [TestClass]
00032 public class ValidatorTests
00033 {
00034     #region Name Validation Tests

```

```

00035
00041     [TestMethod]
00042     [DataRow("John Doe")]
00043     public void ValidateName_ValidName_ReturnsName(string validName)
00044     {
00045         var result = Validator.ValidateName(validName);
00046         Assert.AreEqual(validName, result);
00047     }
00048
00055     [TestMethod]
00056     [DataRow("")]
00057     [DataRow(null)]
00058     public void ValidateName_InvalidName_ThrowsException(string invalidName)
00059     {
00060         var exception = Assert.ThrowsException<ValidationException>(() =>
Validator.ValidateName(invalidName));
00061         Assert.AreEqual(ValidationErrorCode.InvalidName, exception.ErrorCode);
00062     }
00063
00064     #endregion
00065
00066     #region Accommodation Name Validation Tests
00067
00073     [TestMethod]
00074     [DataRow("Ocean View Resort")]
00075     public void ValidateAccommodationName_ValidName_ReturnsName(string validAccommodationName)
00076     {
00077         var result = Validator.ValidateAccommodationName(validAccommodationName);
00078         Assert.AreEqual(validAccommodationName, result);
00079     }
00080
00087     [TestMethod]
00088     [DataRow("")]
00089     [DataRow(null)]
00090     public void ValidateAccommodationName_InvalidName_ThrowsException(string invalidAccommodationName)
00091     {
00092         var exception = Assert.ThrowsException<ValidationException>(
() => Validator.ValidateAccommodationName(invalidAccommodationName));
00093         Assert.AreEqual(ValidationErrorCode.InvalidAccommodationName, exception.ErrorCode);
00094     }
00095
00096     #endregion
00097
00098     #region Email Validation Tests
00099
01006     [TestMethod]
01007     [DataRow("user@example.com")]
01008     public void ValidateEmail_ValidEmail_ReturnsEmail(string validEmail)
01009     {
01010         var result = Validator.ValidateEmail(validEmail);
01011         Assert.AreEqual(validEmail, result);
01012     }
01013
01020     [TestMethod]
01021     [DataRow("invalid-email")]
01022     public void ValidateEmail_InvalidEmail_ThrowsException(string invalidEmail)
01023     {
01024         var exception = Assert.ThrowsException<ValidationException>(() =>
Validator.ValidateEmail(invalidEmail));
01025         Assert.AreEqual(ValidationErrorCode.InvalidEmail, exception.ErrorCode);
01026     }
01027
01028     #endregion
01029
01030     #region Phone Number Validation Tests
01031
01038     [TestMethod]
01039     [DataRow("+351234567890")]
01040     public void ValidatePhoneNumber_ValidPhoneNumber_ReturnsPhoneNumber(string validPhoneNumber)
01041     {
01042         var result = Validator.ValidatePhoneNumber(validPhoneNumber);
01043         Assert.AreEqual(validPhoneNumber, result);
01044     }
01045
01052     [TestMethod]
01053     [DataRow("12345")]
01054     public void ValidatePhoneNumber_InvalidPhoneNumber_ThrowsException(string invalidPhoneNumber)
01055     {
01056         var exception =
Assert.ThrowsException<ValidationException>(() =>
Validator.ValidatePhoneNumber(invalidPhoneNumber));
01057         Assert.AreEqual(ValidationErrorCode.InvalidPhoneNumber, exception.ErrorCode);
01058     }
01059
01060     #endregion
01061     #region Address Validation Tests

```

```

00164
00170     [TestMethod]
00171     [DataRow("123 Main St")]
00172     public void ValidateAddress_ValidAddress_ReturnsAddress(string validAddress)
00173     {
00174         var result = Validator.ValidateAddress(validAddress);
00175         Assert.AreEqual(validAddress, result);
00176     }
00177
00184     [TestMethod]
00185     [DataRow("")]
00186     public void ValidateAddress_InvalidAddress_ThrowsException(string invalidAddress)
00187     {
00188         var exception = Assert.ThrowsException<ValidationException>(() =>
00189             Validator.ValidateAddress(invalidAddress));
00189         Assert.AreEqual(ValidationErrorCode.InvalidAddress, exception.ErrorCode);
00190     }
00191
00192 #endregion
00193
00194 #region Price Validation Tests
00195
00200     [TestMethod]
00201     public void ValidatePrice_ValidPrice_ReturnsPrice()
00202     {
00203         decimal validPrice = 100.0m;
00204         var result = Validator.ValidatePrice(validPrice);
00205         Assert.AreEqual(validPrice, result);
00206     }
00207
00213     [TestMethod]
00214     public void ValidatePrice_InvalidPrice_ThrowsException()
00215     {
00216         decimal invalidPrice = 0.0m;
00217         var exception = Assert.ThrowsException<ValidationException>(() =>
00218             Validator.ValidatePrice(invalidPrice));
00218         Assert.AreEqual(ValidationErrorCode.InvalidPrice, exception.ErrorCode);
00219     }
00220
00221 #endregion
00222
00223 #region Payment Amounts Validation Tests
00224
00229     [TestMethod]
00230     public void ValidatePaymentAmount_ValidAmount_ReturnsAmount()
00231     {
00232         decimal validAmount = 150.0m;
00233         var result = Validator.ValidatePaymentAmount(validAmount);
00234         Assert.AreEqual(validAmount, result);
00235     }
00236
00242     [TestMethod]
00243     public void ValidatePaymentAmount_InvalidAmount_ThrowsException()
00244     {
00245         decimal invalidAmount = -10.0m;
00246         var exception =
00247             Assert.ThrowsException<ValidationException>(() =>
00248                 Validator.ValidatePaymentAmount(invalidAmount));
00248         Assert.AreEqual(ValidationErrorCode.InvalidPaymentValue, exception.ErrorCode);
00249     }
00250
00251 #endregion
00252
00253 #region Payment Status Validation Tests
00254
00258     [TestMethod]
00259     public void ValidatePaymentStatus_ValidStatus_ReturnsStatus()
00260     {
00261         var validStatus = PaymentStatus.Completed;
00262         var result = Validator.ValidatePaymentStatus(validStatus);
00263         Assert.AreEqual(validStatus, result);
00264     }
00265
00269     [TestMethod]
00270     public void ValidatePaymentStatus_InvalidStatus_ThrowsException()
00271     {
00272         var invalidStatus = (PaymentStatus)999;
00273         var exception =
00274             Assert.ThrowsException<ValidationException>(() =>
00275                 Validator.ValidatePaymentStatus(invalidStatus));
00275         Assert.AreEqual(ValidationErrorCode.InvalidPaymentStatus, exception.ErrorCode);
00276     }
00277
00278 #endregion
00279
00280 #region Accommodation Type Validation Tests
00281

```

```

00285     [TestMethod]
00286     public void ValidateAccommodationType_ValidType_ReturnsType()
00287     {
00288         var validType = AccommodationType.Hotel;
00289         var result = Validator.ValidateAccommodationType(validType);
00290         Assert.AreEqual(validType, result);
00291     }
00292
00297     [TestMethod]
00298     public void ValidateAccommodationType_InvalidType_ThrowsException()
00299     {
00300         var invalidType = (AccommodationType)999;
00301         var exception =
00302             Assert.ThrowsException<ValidationException>(() =>
Validator.ValidateAccommodationType(invalidType));
00303         Assert.AreEqual(ValidationErrorCode.InvalidAccommodationType, exception.ErrorCode);
00304     }
00305
00306 #endregion
00307
00308 #region Check In Date Validation Tests
00309
00313     [TestMethod]
00314     public void ValidateCheckInDate_ValidFutureDate_ReturnsDate()
00315     {
00316         var futureDate = DateTime.Now.AddDays(1);
00317         var result = Validator.ValidateCheckInDate(futureDate);
00318         Assert.AreEqual(futureDate, result);
00319     }
00320
00324     [TestMethod]
00325     public void ValidateCheckInDate_PastDate_ThrowsException()
00326     {
00327         var pastDate = DateTime.Now.AddDays(-1);
00328         var exception = Assert.ThrowsException<ValidationException>(() =>
Validator.ValidateCheckInDate(pastDate));
00329         Assert.AreEqual(ValidationErrorCode.InvalidDate, exception.ErrorCode);
00330     }
00331
00332 #endregion
00333
00334 #region Check Out Date Validation Tests
00335
00339     [TestMethod]
00340     public void ValidateCheckOutDate_ValidDateRange_ReturnsCheckOutDate()
00341     {
00342         var checkInDate = DateTime.Now.AddDays(1);
00343         var checkOutDate = checkInDate.AddDays(1);
00344         var result = Validator.ValidateCheckOutDate(checkOutDate, checkInDate);
00345         Assert.AreEqual(checkOutDate, result);
00346     }
00347
00351     [TestMethod]
00352     public void ValidateCheckOutDate_InvalidDateRange_ThrowsException()
00353     {
00354         var checkInDate = DateTime.Now.AddDays(1);
00355         var checkOutDate = checkInDate.AddDays(-1);
00356         var exception = Assert.ThrowsException<ValidationException>(
() => Validator.ValidateCheckOutDate(checkOutDate, checkInDate));
00357         Assert.AreEqual(ValidationErrorCode.InvalidDateRange, exception.ErrorCode);
00358     }
00359
00360 #endregion
00361
00362 #region Total Cost Validation Tests
00363
00368     [TestMethod]
00369     public void ValidateTotalCost_ValidCost_ReturnsTotalCost()
00370     {
00371         decimal validCost = 300.0m;
00372         var result = Validator.ValidateTotalCost(validCost);
00373         Assert.AreEqual(validCost, result);
00374     }
00375
00379     [TestMethod]
00380     public void ValidateTotalCost_InvalidCost_ThrowsException()
00381     {
00382         decimal invalidCost = -5.0m;
00383         var exception = Assert.ThrowsException<ValidationException>(() =>
Validator.ValidateTotalCost(invalidCost));
00384         Assert.AreEqual(ValidationErrorCode.InvalidTotalCost, exception.ErrorCode);
00385     }
00386
00387 #endregion
00388
00389 #region Payment Validation Tests
00390

```

```
00394     [TestMethod]
00395     public void ValidatePayment_ValidPayment_ReturnsPayment()
00396     {
00397         decimal validPayment = 10.0m;
00398         var result = Validator.ValidatePayment(validPayment);
00399         Assert.AreEqual(validPayment, result);
00400     }
00401
00402     [TestMethod]
00403     public void ValidatePayment_InvalidPayment_ThrowsException()
00404     {
00405         decimal invalidPayment = -100.0m;
00406         var exception = Assert.ThrowsException<ValidationException>(() =>
00407             Validator.ValidatePayment(invalidPayment));
00408         Assert.AreEqual(ValidationErrorCode.InvalidPaymentValue, exception.ErrorCode);
00409     }
00410 }
00411 #endregion
00412
00413 #region Reservation Status Validation Tests
00414
00415     [TestMethod]
00416     public void ValidateReservationStatus_ValidStatus_ReturnsStatus()
00417     {
00418         var validStatus = ReservationStatus.Confirmed;
00419         var result = Validator.ValidateReservationStatus(validStatus);
00420         Assert.AreEqual(validStatus, result);
00421     }
00422
00423     [TestMethod]
00424     public void ValidateReservationStatus_InvalidStatus_ThrowsException()
00425     {
00426         var invalidStatus = (ReservationStatus)999;
00427         var exception =
00428             Assert.ThrowsException<ValidationException>(() =>
00429                 Validator.ValidateReservationStatus(invalidStatus));
00430         Assert.AreEqual(ValidationErrorCode.InvalidReservationStatus, exception.ErrorCode);
00431     }
00432 }
00433 #endregion
00434 }
```



# Index

.NETCoreApp,Version=v8.0.AssemblyAttributes.cs, [24](#)

AccommodationTests.cs, [21](#)

Client\_FullData\_CreatesClient

SmartStay.Tests.ClientTests, [10](#)

Client\_GenerateUniqueClientId\_CreatesUniquelds

SmartStay.Tests.ClientTests, [10](#)

Client\_InvalidAddress\_ThrowsValidationException

SmartStay.Tests.ClientTests, [10](#)

Client\_InvalidEmail\_ThrowsValidationException

SmartStay.Tests.ClientTests, [10](#)

Client\_InvalidPaymentMethod\_ThrowsValidationException

SmartStay.Tests.ClientTests, [10](#)

Client\_InvalidPhoneNumber\_ThrowsValidationException

SmartStay.Tests.ClientTests, [10](#)

Client\_SetValidAddress\_UpdatesAddress

SmartStay.Tests.ClientTests, [11](#)

Client\_SetValidEmail\_UpdatesEmail

SmartStay.Tests.ClientTests, [11](#)

Client\_SetValidFirstName\_UpdatesFirstName

SmartStay.Tests.ClientTests, [11](#)

Client\_SetValidLastName\_UpdatesLastName

SmartStay.Tests.ClientTests, [11](#)

Client\_SetValidPaymentMethod\_UpdatesPaymentMethod

SmartStay.Tests.ClientTests, [11](#)

Client\_SetValidPhoneNumber\_UpdatesPhoneNumber

SmartStay.Tests.ClientTests, [11](#)

Client\_ToString\_ReturnsValidJson

SmartStay.Tests.ClientTests, [12](#)

Client\_ValidData\_CreatesClient

SmartStay.Tests.ClientTests, [12](#)

ClientTests.cs, [21](#)

ReservationTests.cs, [24](#)

SmartStay, [7](#)

SmartStay.Tests, [7](#)

SmartStay.Tests.AssemblyInfo.cs, [24](#)

SmartStay.Tests.ClientTests, [9](#)

Client\_FullData\_CreatesClient, [10](#)

Client\_GenerateUniqueClientId\_CreatesUniquelds,  
[10](#)

Client\_InvalidAddress\_ThrowsValidationException,  
[10](#)

Client\_InvalidEmail\_ThrowsValidationException,  
[10](#)

Client\_InvalidPaymentMethod\_ThrowsValidationException,ValidatePaymentStatus\_ValidStatus\_ReturnsStatus,  
[10](#) [18](#)

Client\_InvalidPhoneNumber\_ThrowsValidationException, ValidatePhoneNumber\_InvalidPhoneNumber\_ThrowsException,  
[10](#)

Client\_SetValidAddress\_UpdatesAddress, [11](#)

Client\_SetValidEmail\_UpdatesEmail, [11](#)

Client\_SetValidFirstName\_UpdatesFirstName, [11](#)

Client\_SetValidLastName\_UpdatesLastName, [11](#)

Client\_SetValidPaymentMethod\_UpdatesPaymentMethod,  
[11](#)

Client\_SetValidPhoneNumber\_UpdatesPhoneNumber,  
[11](#)

Client\_ToString\_ReturnsValidJson, [12](#)

Client\_ValidData\_CreatesClient, [12](#)

SmartStay.Tests.GlobalUsings.g.cs, [25](#)

SmartStay.Tests.ValidatorTests, [12](#)

ValidateAccommodationName\_InvalidName\_ThrowsException,  
[14](#)

ValidateAccommodationName\_ValidName\_ReturnsName,  
[14](#)

ValidateAccommodationType\_InvalidType\_ThrowsException,  
[15](#)

ValidateAccommodationType\_ValidType\_ReturnsType,  
[15](#)

ValidateAddress\_InvalidAddress\_ThrowsException,  
[15](#)

ValidateAddress\_ValidAddress\_ReturnsAddress,  
[15](#)

ValidateCheckInDate\_PastDate\_ThrowsException,  
[16](#)

ValidateCheckInDate\_ValidFutureDate\_ReturnsDate,  
[16](#)

ValidateCheckOutDate\_InvalidDateRange\_ThrowsException,  
[16](#)

ValidateCheckOutDate\_ValidDateRange\_ReturnsCheckOutDate,  
[16](#)

ValidateEmail\_InvalidEmail\_ThrowsException, [16](#)

ValidateEmail\_ValidEmail\_ReturnsEmail, [17](#)

ValidateName\_InvalidName\_ThrowsException, [17](#)

ValidateName\_ValidName\_ReturnsName, [17](#)

ValidatePayment\_InvalidPayment\_ThrowsException,  
[17](#)

ValidatePayment\_ValidPayment\_ReturnsPayment,  
[18](#)

ValidatePaymentAmount\_InvalidAmount\_ThrowsException,  
[18](#)

ValidatePaymentAmount\_ValidAmount\_ReturnsAmount,  
[18](#)

ValidatePaymentStatus\_InvalidStatus\_ThrowsException,  
[18](#)

ValidatePaymentStatus\_ValidStatus\_ReturnsStatus,  
[18](#)

ValidatePhoneNumber\_InvalidPhoneNumber\_ThrowsException,

- 18
- ValidatePhoneNumber\_ValidPhoneNumber\_ReturnsPhoneNumber, 18
- 19
- ValidatePrice\_InvalidPrice\_ThrowsException, 19
- ValidatePrice\_ValidPrice\_ReturnsPrice, 19
- ValidateReservationStatus\_InvalidStatus\_ThrowsException, 19
- ValidateReservationStatus\_ValidStatus\_ReturnsStatus, 19
- ValidateTotalCost\_InvalidCost\_ThrowsException, 20
- ValidateTotalCost\_ValidCost\_ReturnsTotalCost, 20
- ValidatePrice\_ValidPrice\_ReturnsPrice, SmartStay.Tests.ValidatorTests, 19
- ValidateReservationStatus\_InvalidStatus\_ThrowsException, SmartStay.Tests.ValidatorTests, 19
- ValidateReservationStatus\_ValidStatus\_ReturnsStatus, SmartStay.Tests.ValidatorTests, 19
- ValidateTotalCost\_InvalidCost\_ThrowsException, SmartStay.Tests.ValidatorTests, 20
- ValidateTotalCost\_ValidCost\_ReturnsTotalCost, SmartStay.Tests.ValidatorTests, 20
- ValidationTests.cs, 25
- ValidateAccommodationName\_InvalidName\_ThrowsException, SmartStay.Tests.ValidatorTests, 14
- ValidateAccommodationName\_ValidName\_ReturnsName, SmartStay.Tests.ValidatorTests, 14
- ValidateAccommodationType\_InvalidType\_ThrowsException, SmartStay.Tests.ValidatorTests, 15
- ValidateAccommodationType\_ValidType\_ReturnsType, SmartStay.Tests.ValidatorTests, 15
- ValidateAddress\_InvalidAddress\_ThrowsException, SmartStay.Tests.ValidatorTests, 15
- ValidateAddress\_ValidAddress\_ReturnsAddress, SmartStay.Tests.ValidatorTests, 15
- ValidateCheckInDate\_PastDate\_ThrowsException, SmartStay.Tests.ValidatorTests, 16
- ValidateCheckInDate\_ValidFutureDate\_ReturnsDate, SmartStay.Tests.ValidatorTests, 16
- ValidateCheckOutDate\_InvalidDateRange\_ThrowsException, SmartStay.Tests.ValidatorTests, 16
- ValidateCheckOutDate\_ValidDateRange\_ReturnsCheckOutDate, SmartStay.Tests.ValidatorTests, 16
- ValidateEmail\_InvalidEmail\_ThrowsException, SmartStay.Tests.ValidatorTests, 16
- ValidateEmail\_ValidEmail\_ReturnsEmail, SmartStay.Tests.ValidatorTests, 17
- ValidateName\_InvalidName\_ThrowsException, SmartStay.Tests.ValidatorTests, 17
- ValidateName\_ValidName\_ReturnsName, SmartStay.Tests.ValidatorTests, 17
- ValidatePayment\_InvalidPayment\_ThrowsException, SmartStay.Tests.ValidatorTests, 17
- ValidatePayment\_ValidPayment\_ReturnsPayment, SmartStay.Tests.ValidatorTests, 18
- ValidatePaymentAmount\_InvalidAmount\_ThrowsException, SmartStay.Tests.ValidatorTests, 18
- ValidatePaymentAmount\_ValidAmount\_ReturnsAmount, SmartStay.Tests.ValidatorTests, 18
- ValidatePaymentStatus\_InvalidStatus\_ThrowsException, SmartStay.Tests.ValidatorTests, 18
- ValidatePaymentStatus\_ValidStatus\_ReturnsStatus, SmartStay.Tests.ValidatorTests, 18
- ValidatePhoneNumber\_InvalidPhoneNumber\_ThrowsException, SmartStay.Tests.ValidatorTests, 18
- ValidatePhoneNumber\_ValidPhoneNumber\_ReturnsPhoneNumber, SmartStay.Tests.ValidatorTests, 19
- ValidatePrice\_InvalidPrice\_ThrowsException, SmartStay.Tests.ValidatorTests, 19