

Hotel

Generated by Doxygen 1.9.3

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 Date Class Reference	5
3.1.1 Detailed Description	6
3.1.2 Constructor & Destructor Documentation	6
3.1.2.1 Date()	6
3.1.3 Member Function Documentation	6
3.1.3.1 getToday()	6
3.1.3.2 operator>()	7
3.1.3.3 operator++()	7
3.1.3.4 operator-()	7
3.1.3.5 operator<()	7
3.1.3.6 operator==()	8
3.1.3.7 operator>()	8
3.1.3.8 readDataFromBinary()	9
3.1.3.9 writeToBinaryFile()	9
3.1.4 Friends And Related Function Documentation	9
3.1.4.1 operator<<	9
3.1.4.2 operator>>	10
3.2 DatePeriod Struct Reference	10
3.2.1 Detailed Description	10
3.2.2 Member Function Documentation	11
3.2.2.1 length()	11
3.2.2.2 operator++()	11
3.3 Hotel Class Reference	11
3.3.1 Detailed Description	12
3.3.2 Constructor & Destructor Documentation	12
3.3.2.1 Hotel()	12
3.3.3 Member Function Documentation	13
3.3.3.1 freeRoom()	13
3.3.3.2 getName()	13
3.3.3.3 getReport()	13
3.3.3.4 reserveRoom()	14
3.3.3.5 searchRoom()	14
3.3.3.6 seeRoomForNights()	15
3.3.3.7 serviceRoom()	15
3.3.3.8 showAvailableRooms()	15
3.3.3.9 showToday()	16

3.3.3.10 today()	16
3.3.3.11 workDay()	16
3.4 HotelBuilding Class Reference	17
3.4.1 Detailed Description	17
3.4.2 Constructor & Destructor Documentation	18
3.4.2.1 HotelBuilding()	18
3.4.3 Member Function Documentation	18
3.4.3.1 createReport()	18
3.4.3.2 getRoomCount()	18
3.4.3.3 newDate()	19
3.4.3.4 operator[]()	19
3.4.3.5 readDataFromBinary()	19
3.4.3.6 showAvailableRooms()	19
3.4.3.7 showRoomForNights()	20
3.4.3.8 showRoomsStatesToday()	20
3.4.3.9 suggestRoom()	20
3.4.3.10 writeToBinaryFile()	21
3.5 HotelInterface Class Reference	21
3.5.1 Member Function Documentation	21
3.5.1.1 createHeader()	21
3.6 Reservation Class Reference	22
3.6.1 Detailed Description	23
3.6.2 Constructor & Destructor Documentation	23
3.6.2.1 Reservation()	23
3.6.3 Member Function Documentation	23
3.6.3.1 getFrom()	23
3.6.3.2 getNights()	23
3.6.3.3 getNote()	24
3.6.3.4 getTo()	24
3.6.3.5 isActive()	24
3.6.3.6 isPast()	24
3.6.3.7 isServiced()	25
3.6.3.8 LeavingInAdvance()	25
3.6.3.9 onDate()	25
3.6.3.10 readDataFromBinary()	25
3.6.3.11 stateOnDate()	26
3.6.3.12 writeToBinaryFile()	26
3.7 Room Class Reference	26
3.7.1 Detailed Description	27
3.7.2 Constructor & Destructor Documentation	27
3.7.2.1 Room()	27
3.7.3 Member Function Documentation	28

3.7.3.1 addReservation()	28
3.7.3.2 closeForService()	28
3.7.3.3 freeRoom()	29
3.7.3.4 isFreeInPeriod()	29
3.7.3.5 isFreeOnDate()	29
3.7.3.6 readDataFromBinary()	30
3.7.3.7 showReservationsInPeriod()	30
3.7.3.8 writeToBinaryFile()	30
3.8 RoomAnalyzer Class Reference	31
3.8.1 Detailed Description	31
3.8.2 Member Function Documentation	31
3.8.2.1 soonestFreePeriod()	31
3.8.2.2 suggest()	31
3.9 String Class Reference	33
4 File Documentation	35
4.1 Constants.hpp	35
4.2 Date.hpp	35
4.3 Hotel.hpp	36
4.4 HotelBuilding.hpp	37
4.5 HotelInterface.hpp	37
4.6 Reservation.hpp	38
4.7 Room.hpp	38
4.8 RoomAnalyzer.hpp	39
4.9 String.hpp	40
4.10 Types.hpp	40
Index	41

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Date	Class representing date with day, month and year	5
DatePeriod	Class containing two dates forming a period of time from Date to Date	10
Hotel	Class representing hotel with name, current Date and a building (list of rooms)	11
HotelBuilding	Class representing list of rooms	17
HotelInterface	21
Reservation	Class representing information about a reservation	22
Room	Class representing a room in hotel	26
RoomAnalyzer	Supporting class to perform algorithms on the rooms in a building	31
String	33

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Constants.hpp	??
Date.hpp	??
Hotel.hpp	??
HotelBuilding.hpp	??
HotelInterface.hpp	??
Reservation.hpp	??
Room.hpp	??
RoomAnalyzer.hpp	??
String.hpp	??
Types.hpp	??

Chapter 3

Class Documentation

3.1 Date Class Reference

Class representing date with day, month and year.

```
#include <Date.hpp>
```

Public Member Functions

- [Date](#) (unsigned short d=1, unsigned short m=1, unsigned short y=1900)
Construct a new [Date](#) object from day, month and year. Default [Date](#) is 1/1/1900.
- bool [operator<](#) (const [Date](#) other) const
checks if this [Date](#) is chronologically before other [Date](#)
- bool [operator<=](#) (const [Date](#) other) const
see operator<
- bool [operator>](#) (const [Date](#) other) const
checks if this [Date](#) is chronologically after other [Date](#)
- bool [operator>=](#) (const [Date](#) other) const
see operator>
- bool [operator==](#) (const [Date](#) other) const
checks if two dates are identical
- const char * [operator\(\)](#) (char *buf) const
records this [Date](#) in buffer in format YYYY-MM-DD
- int [operator-](#) ([Date](#) other) const
- [Date](#) & [operator++](#) ()
overloaded prefix incrementation operator for [Date](#)
- void [writeToBinaryFile](#) (std::ofstream &ofs)
write the [Room](#) data into binary file opened by ofstream
- void [readDataFromBinary](#) (std::ifstream &ifs)
read the [Room](#) data from binary file opened by ifstream

Static Public Member Functions

- static [Date](#) [getToday](#) ()
Get the today date.

Friends

- `std::istream & operator>> (std::istream &is, Date &d)`
overloaded operator for inputing [Date](#)
- `std::ostream & operator<< (std::ostream &os, const Date &d)`
overloaded operator for outputing [Date](#)

3.1.1 Detailed Description

Class representing date with day, month and year.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 [Date\(\)](#)

```
Date::Date (
    unsigned short d = 1,
    unsigned short m = 1,
    unsigned short y = 1900 ) [inline]
```

Construct a new [Date](#) object from day, month and year. Default [Date](#) is 1/1/1900.

Parameters

<i>d</i>	day
<i>m</i>	month
<i>y</i>	year

3.1.3 Member Function Documentation

3.1.3.1 [getToday\(\)](#)

```
Date Date::getToday ( ) [static]
```

Get the today date.

Returns

[Date](#)

3.1.3.2 operator()

```
const char * Date::operator() (
    char * buf ) const
```

records this [Date](#) in buffer in format YYYY-MM-DD

Parameters

<i>buf</i>	buffer where Date is recorded
------------	---

Returns

const char* pointer to beginning of buf

3.1.3.3 operator++()

```
Date & Date::operator++ ( )
```

overloaded prefix incrementation operator for [Date](#)

Returns

[Date](#)& reference to this [Date](#)

3.1.3.4 operator-()

```
int Date::operator- (
    Date other ) const
```

Parameters

<i>other</i>	Date
--------------	----------------------

Returns

int difference between of this [Date](#) and other

3.1.3.5 operator<()

```
bool Date::operator< (
    const Date other ) const
```

checks if this [Date](#) is chronologically before other [Date](#)

Parameters

<i>other</i>	compared Date
--------------	-------------------------------

Returns

true this [Date](#) is chronologically before other
false this [Date](#) is not chronologically before other

3.1.3.6 operator==()

```
bool Date::operator== (
    const Date other ) const
```

checks if two dates are identical

Parameters

<i>other</i>	compared Date
--------------	-------------------------------

Returns

true the dates are identical
false the dates are not identical

3.1.3.7 operator>()

```
bool Date::operator> (
    const Date other ) const
```

checks if this [Date](#) is chronologically after other [Date](#)

Parameters

<i>other</i>	compared Date
--------------	-------------------------------

Returns

true this [Date](#) is chronologically after other
false this [Date](#) is not chronologically after other

3.1.3.8 readDataFromBinary()

```
void Date::readDataFromBinary (
    std::ifstream & ifs )
```

read the [Room](#) data from binary file opened by ifstream

Parameters

<i>ofs</i>	input stream connected to binary file
------------	---------------------------------------

3.1.3.9 writeToBinaryFile()

```
void Date::writeToBinaryFile (
    std::ofstream & ofs )
```

write the [Room](#) data into binary file opened by ofstream

Parameters

<i>ofs</i>	output stream connected to binary file
------------	--

3.1.4 Friends And Related Function Documentation

3.1.4.1 operator<<

```
std::ostream & operator<< (
    std::ostream & os,
    const Date & d ) [friend]
```

overloaded operator for outputing [Date](#)

Parameters

<i>os</i>	output stream
<i>d</i>	Date to be output

Returns

std::ostream& reference to the output stream

3.1.4.2 operator>>

```
std::istream & operator>> (
    std::istream & is,
    Date & d ) [friend]
```

overloaded operator for inputting [Date](#)

Parameters

<i>is</i>	input stream
<i>d</i>	Date to be input

Returns

std::istream& reference to the input stream

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

- [Date.hpp](#)
- [Date.cpp](#)

3.2 DatePeriod Struct Reference

Class containing two dates forming a period of time from [Date](#) to [Date](#).

```
#include <Date.hpp>
```

Public Member Functions

- unsigned [length](#) () const
distance in days of the period
- [DatePeriod](#) & [operator++](#) ()
moving period one day froward
- void [readProper](#) ()
method to read from stdin a proper period of time (from is before to)

Public Attributes

- [Date](#) **from**
beginning [Date](#) of the period
- [Date](#) **to**
end [Date](#) of the period

3.2.1 Detailed Description

Class containing two dates forming a period of time from [Date](#) to [Date](#).

3.2.2 Member Function Documentation

3.2.2.1 length()

```
unsigned DatePeriod::length ( ) const [inline]
```

distance in days of the period

Returns

unsigned days between beginning and end

3.2.2.2 operator++()

```
DatePeriod & DatePeriod::operator++ ( )
```

moving period one day forward

Returns

[DatePeriod](#)& this [DatePeriod](#)

The documentation for this struct was generated from the following files:

- [Date.hpp](#)
- [Date.cpp](#)

3.3 Hotel Class Reference

Class representing hotel with name, current [Date](#) and a building (list of rooms)

```
#include <Hotel.hpp>
```

Public Member Functions

- [Hotel](#) ([String](#) hotelDataFile)
Construct a new [Hotel](#) object.
- [Hotel](#) (const [Hotel](#) &)=delete
- [Hotel](#) & **operator=** (const [Hotel](#) &)=delete
- **~Hotel** ()
Destroy the [Hotel](#) object.
- [String](#) **getName** () const
get the name of this [Hotel](#)
- void **nextDay** ()
advance to the nextDay
- bool **reserveRoom** (unsigned number, const [DatePeriod](#) &period, [String](#) name="-", [String](#) note="None\n")
makes a new [Reservation](#) for particular [Room](#) and period with options for name of guest and notes to the [Reservation](#)
- [Hotel](#) & **showAvailableRooms** (std::ostream &, [Date](#))
output to stream all available rooms for a particular [Date](#)
- bool **freeRoom** (unsigned number)
tries to free [Room](#) with particular ID
- [Hotel](#) & **getReport** ([DatePeriod](#) &period)
Creates report for the usage of this [Hotel](#)'s rooms in the period from-to. Report written in file named "report-YYYY-MM-DD.txt" where YYYY-MM-DD is the beginning of the period.
- void **searchRoom** (unsigned minBeds, const [DatePeriod](#) &period) const
given minimum number of beds and a desired period to print most suitable rooms for accomodation
- bool **serviceRoom** (unsigned number, const [DatePeriod](#) &period, [String](#) note)
plans maintenance for particular [Room](#) and period leaving note for the service
- [Hotel](#) & **showToday** ()
print status of the building rooms
- [Hotel](#) & **seeRoomForNights** (unsigned number, unsigned nights)
print soonest period of particular number of days when particular room is free
- bool **workDay** ()
work with this [Hotel](#) for a whole day

Static Public Member Functions

- static [Date](#) **today** ()
get today's [Date](#) according to all [Hotels](#)

3.3.1 Detailed Description

Class representing hotel with name, current [Date](#) and a building (list of rooms)

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Hotel()

```
Hotel::Hotel (
    String hotelDataFile )
```

Construct a new [Hotel](#) object.

Parameters

<i>hotelDataFile</i>	path to file where rooms are recorded
----------------------	---------------------------------------

3.3.3 Member Function Documentation

3.3.3.1 freeRoom()

```
bool Hotel::freeRoom (
    unsigned number )
```

tries to free [Room](#) with particular ID

Parameters

<i>number</i>	Room's ID
---------------	---------------------------

Returns

true room is now free
false room not found

3.3.3.2 getName()

```
String Hotel::getName ( ) const [inline]
```

get the name of this [Hotel](#)

Returns

[String](#)

3.3.3.3 getReport()

```
Hotel & Hotel::getReport (
    DatePeriod & period )
```

Creates report for the usage of this [Hotel's](#) rooms in the period from-to. Report written in file named "report-YYYY-MM-DD.txt" where YYYY-MM-DD is the beginning of the period.

Parameters

<i>period</i>	desired period of time
---------------	------------------------

Returns

[Hotel](#)& this [Hotel](#)

3.3.3.4 reserveRoom()

```
bool Hotel::reserveRoom (
    unsigned number,
    const DatePeriod & period,
    String name = "-",
    String note = "None\n" )
```

makes a new [Reservation](#) for particular [Room](#) and period with options for name of guest and notes to the [Reservation](#)

Parameters

<i>number</i>	of the desired Room
<i>from</i>	accomodation Date
<i>to</i>	leaving Date
<i>name</i>	guest's name
<i>note</i>	note to the reservation

Returns

true successfull reservation
false failed reservation (not made)

3.3.3.5 searchRoom()

```
void Hotel::searchRoom (
    unsigned minBeds,
    const DatePeriod & period ) const
```

given minimum number of beds and a desired period to print most suitable rooms for accomodation

Parameters

<i>minBeds</i>	minimum number of beds
<i>period</i>	desired time period

3.3.3.6 seeRoomForNights()

```
Hotel & Hotel::seeRoomForNights (
    unsigned number,
    unsigned nights )
```

print soonest period of particular number of days when particular room is free

Parameters

<i>number</i>	ID of a room
<i>nights</i>	number of nights to stay in the Hotel for

Returns

[Hotel](#)& this [Hotel](#)

3.3.3.7 serviceRoom()

```
bool Hotel::serviceRoom (
    unsigned number,
    const DatePeriod & period,
    String note )
```

plans maintenance for particular [Room](#) and period leaving note for the service

Parameters

<i>number</i>	Room 's ID
<i>period</i>	desired period of time
<i>note</i>	any notes to the service

Returns

true service planned successfully

false service planning failed (room not found or is reserved for the period)

3.3.3.8 showAvailableRooms()

```
Hotel & Hotel::showAvailableRooms (
    std::ostream & os,
    Date d )
```

output to stream all available rooms for a particular [Date](#)

Returns

[Hotel](#)& this [Hotel](#)

3.3.3.9 showToday()

```
Hotel & Hotel::showToday ( )
```

print status of the building rooms

Returns

[Hotel](#)& this [Hotel](#)

3.3.3.10 today()

```
static Date Hotel::today ( ) [inline], [static]
```

get today's [Date](#) according to all [Hotels](#)

Returns

[Date](#)

3.3.3.11 workDay()

```
bool Hotel::workDay ( )
```

work with this [Hotel](#) for a whole day

Returns

true day ended with the [Hotel](#) still working

false the [Hotel](#) was closed and its data written to "<name>.dat"

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

- [Hotel.hpp](#)
- [Hotel.cpp](#)

3.4 HotelBuilding Class Reference

Class representing list of rooms.

```
#include <HotelBuilding.hpp>
```

Public Member Functions

- [HotelBuilding](#) (std::ifstream &ifs)
Construct a new [HotelBuilding](#) object from a text file containing rooms info Format of the file:
- [HotelBuilding](#) (const [HotelBuilding](#) &other)=delete
- [HotelBuilding](#) & [operator=](#) ([HotelBuilding](#) &other)=delete
- [~HotelBuilding](#) ()
Destroy the [HotelBuilding](#) object.
- size_t [getRoomCount](#) () const
Get the room count.
- [Room](#) * [operator\[\]](#) (unsigned roomNumber) const
seek for a room with particular number
- void [newDate](#) ([Date](#) d)
update this [HotelBuilding](#) rooms data on a new [Date](#)
- void [showAvailableRooms](#) (std::ostream &os, [Date](#) d) const
show available rooms on a particular [Date](#)
- void [createReport](#) ([DatePeriod](#) &period) const
*Create a report for the usage of rooms for a particular period of time (ending before the today [Date](#)) in folder reports
Format of the report:*
- void [suggestRoom](#) (unsigned beds, const [DatePeriod](#) &period)
*show top DISPLAY (or all rooms if less than DISPLAY + 1) sorted by suitability for the guest, given minimal number of
beds and particular period of time for an eventual reservation*
- void [showRoomsStatesToday](#) ([Date](#) today) const
prints to stdout all rooms together with up to today [Date](#) info about their Availability (now, in future or in the past)
- void [showRoomForNights](#) (unsigned number, unsigned nights, [Date](#) today) const
show soonest period of particular nights when particular room is free
- void [writeToBinaryFile](#) (std::ofstream &ofs)
write the [HotelBuilding](#) data into binary file opened by ofstream
- void [readDataFromBinary](#) (std::ifstream &ifs)
read the [HotelBuilding](#) data from binary file opened by ifstream

Friends

- class [RoomAnalyzer](#)
using [RoomAnalyzer](#) to perform algorithms for the room list (database)

3.4.1 Detailed Description

Class representing list of rooms.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 HotelBuilding()

```
HotelBuilding::HotelBuilding (
    std::ifstream & ifs )
```

Construct a new [HotelBuilding](#) object from a text file containing rooms info Format of the file:

1. <size> ... n(>1). <room #(n-2) number> <room #(n-2) count of beds>

Parameters

<i>ifs</i>	input file stream to text file, containing rooms data
------------	---

3.4.3 Member Function Documentation

3.4.3.1 createReport()

```
void HotelBuilding::createReport (
    DatePeriod & period ) const
```

Create a report for the usage of rooms for a particular period of time (ending before the today [Date](#)) in folder reports
Format of the report:

Report for the usage of the rooms between <beginning of period> and <end of period>: ... [Room #](#) <Room number> between <beginning of period> and <end of period>: <count of nights> nights.

Parameters

<i>period</i>	period of time
---------------	----------------

3.4.3.2 getRoomCount()

```
size_t HotelBuilding::getRoomCount ( ) const [inline]
```

Get the room count.

Returns

size_t count of rooms

3.4.3.3 newDate()

```
void HotelBuilding::newDate (
    Date d )
```

update this [HotelBuilding](#) rooms data on a new [Date](#)

Parameters

<i>d</i>	new Date
----------	--------------------------

3.4.3.4 operator[]()

```
Room * HotelBuilding::operator[] (
    unsigned roomNumber ) const
```

seek for a room with particular number

Parameters

<i>roomNumber</i>	number if the sought room
-------------------	---------------------------

Returns

Room* if found -> pointer to this [Room](#) else -> nullptr

3.4.3.5 readDataFromBinary()

```
void HotelBuilding::readDataFromBinary (
    std::ifstream & ifs )
```

read the [HotelBuilding](#) data from binary file opened by ifstream

Parameters

<i>ofs</i>	input stream connected to binary file
------------	---------------------------------------

3.4.3.6 showAvailableRooms()

```
void HotelBuilding::showAvailableRooms (
    std::ostream & os,
    Date d ) const
```

show available rooms on a particular [Date](#)

Parameters

<i>os</i>	output stream where the available rooms will be shown in format: Available rooms for <today>: Number: <room number> Bed count: <count of beds> ...
<i>d</i>	Date

3.4.3.7 showRoomForNights()

```
void HotelBuilding::showRoomForNights (
    unsigned number,
    unsigned nights,
    Date today ) const
```

show soonest period of particular nights when particular room is free

Parameters

<i>number</i>	ID of the room
<i>nights</i>	length of the period
<i>today</i>	today's Date

3.4.3.8 showRoomsStatesToday()

```
void HotelBuilding::showRoomsStatesToday (
    Date today ) const
```

prints to stdout all rooms together with up to today [Date](#) info about their Availability (now, in future or in the past)

Parameters

<i>today</i>	Date
--------------	----------------------

3.4.3.9 suggestRoom()

```
void HotelBuilding::suggestRoom (
    unsigned beds,
    const DatePeriod & period )
```

show top DISPLAY (or all rooms if less than DISPLAY + 1) sorted by suitability for the guest, given minimal number of beds and particular period of time for an eventual reservation

Parameters

<i>beds</i>	minimal number of beds insisted in the room
<i>period</i>	period of time

3.4.3.10 writeToBinaryFile()

```
void HotelBuilding::writeToBinaryFile (
    std::ofstream & ofs )
```

write the [HotelBuilding](#) data into binary file opened by ofstream

Parameters

<i>ofs</i>	output stream connected to binary file
------------	--

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

- [HotelBuilding.hpp](#)
- [HotelBuilding.cpp](#)

3.5 HotelInterface Class Reference

Static Public Member Functions

- static void [createHeader](#) ([Hotel](#) &H)
print to stdout centered name of the [Hotel](#)
- static void [beginDay](#) ()
print the today's date and available commands to use for a [Hotel](#)

3.5.1 Member Function Documentation

3.5.1.1 createHeader()

```
void HotelInterface::createHeader (
    Hotel & H ) [static]
```

print to stdout centered name of the [Hotel](#)

Parameters

<i>H</i>	Hotel whose name is to be printed
----------	---

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

- [HotelInterface.hpp](#)
- [HotelInterface.cpp](#)

3.6 Reservation Class Reference

Class representing information about a reservation.

```
#include <Reservation.hpp>
```

Public Member Functions

- [Reservation](#) ([String](#) name, const [DatePeriod](#) &p, [String](#) n="None.\n", bool s=false)
Construct a new [Reservation](#) object.
- **Reservation** (const [Reservation](#) &)=delete
- [Reservation](#) & **operator=** (const [Reservation](#) &)=delete
- bool [isActive](#) () const
see if this [Reservation](#) is active (today is part of the period)
- bool [isPast](#) () const
see if this [Reservation](#) is past (today is after end of period)
- bool [isServiced](#) () const
see if this [Reservation](#) is a maintenance
- [Date](#) [getFrom](#) () const
get beginning [Date](#) of this [Reservation](#)
- [Date](#) [getTo](#) () const
get end [Date](#) of this [Reservation](#)
- unsigned [getNights](#) () const
get count of nights of this [Reservation](#)
- [String](#) [getNote](#) () const
get the note to this [Reservation](#)
- void [onDate](#) ([Date](#) d)
update the state of the reservation based on new today's [Date](#) (d)
- [ReservationState](#) [stateOnDate](#) ([Date](#)) const
see what would the state of this [Reservation](#) be on particular [Date](#)
- bool [LeavingInAdvance](#) ([Date](#))
try to change end of period for earlier end of this [Reservation](#)
- void [writeToBinaryFile](#) (std::ofstream &ofs)
write the [Reservation](#) data into binary file opened by ofstream
- void [readDataFromBinary](#) (std::ifstream &ifstream)
read the [Reservation](#) data from binary file opened by ifstream

3.6.1 Detailed Description

Class representing information about a reservation.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 Reservation()

```
Reservation::Reservation (
    String name,
    const DatePeriod & p,
    String n = "None.\n",
    bool s = false )
```

Construct a new [Reservation](#) object.

Parameters

<i>name</i>	of the reserver
<i>p</i>	
<i>n</i>	note left for the reservation
<i>s</i>	whether it is reservation or maintenance

3.6.3 Member Function Documentation

3.6.3.1 getFrom()

```
Date Reservation::getFrom ( ) const [inline]
```

get beginning [Date](#) of this [Reservation](#)

Returns

[Date](#) beginning of the [Reservation](#)

3.6.3.2 getNights()

```
unsigned Reservation::getNights ( ) const [inline]
```

get count of nights of this [Reservation](#)

Returns

unsigned count of nights of this [Reservation](#)

3.6.3.3 `getNote()`

```
String Reservation::getNote ( ) const [inline]
```

get the note to this [Reservation](#)

Returns

[String](#)

3.6.3.4 `getTo()`

```
Date Reservation::getTo ( ) const [inline]
```

get end [Date](#) of this [Reservation](#)

Returns

[Date](#) end of the [Reservation](#)

3.6.3.5 `isActive()`

```
bool Reservation::isActive ( ) const [inline]
```

see if this [Reservation](#) is active (today is part of the period)

Returns

true [Reservation](#) is active

false [Reservation](#) is not active (past or future)

3.6.3.6 `isPast()`

```
bool Reservation::isPast ( ) const [inline]
```

see if this [Reservation](#) is past (today is after end of period)

Returns

true [Reservation](#) is past

false [Reservation](#) is not past (active or future)

3.6.3.7 isServiced()

```
bool Reservation::isServiced ( ) const [inline]
```

see if this [Reservation](#) is a maintenance

Returns

true this [Reservation](#) is a maintenace

false this [Reservation](#) is for a guest

3.6.3.8 LeavingInAdvance()

```
bool Reservation::LeavingInAdvance (
    Date newTo )
```

try to change end of period for earlier end of this [Reservation](#)

Returns

true end date modified for leaving in advance

false new leaving [Date](#) not appropriate for earlier leaving

3.6.3.9 onDate()

```
void Reservation::onDate (
    Date d )
```

update the state of the reservation based on new today's [Date](#) (d)

Parameters

<i>d</i>	new today's Date
----------	----------------------------------

3.6.3.10 readDataFromBinary()

```
void Reservation::readDataFromBinary (
    std::ifstream & ifs )
```

read the [Reservation](#) data from binary file opened by ifstream

Parameters

<i>ofs</i>	input stream connected to binary file
------------	---------------------------------------

3.6.3.11 stateOnDate()

```
ReservationState Reservation::stateOnDate (
    Date d ) const
```

see what would the state of this [Reservation](#) be on particular [Date](#)

Returns

ReservationState state on desired [Date](#)

3.6.3.12 writeToBinaryFile()

```
void Reservation::writeToBinaryFile (
    std::ofstream & ofs )
```

write the [Reservation](#) data into binary file opened by ofstream

Parameters

<i>ofs</i>	output stream connected to binary file
------------	--

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

- Reservation.hpp
- Reservation.cpp

3.7 Room Class Reference

Class representing a room in hotel.

```
#include <Room.hpp>
```

Public Member Functions

- [Room](#) (unsigned n, unsigned bC)
Construct a new [Room](#) object.

- **Room** (const [Room](#) &)=delete
forbidden copying of rooms
- **Room & operator=** (const [Room](#) &)=delete
forbidden copying of rooms
- **~Room** ()
Destroy the [Room](#) object.
- unsigned **getNumber** () const
- unsigned **getBedCount** () const
- bool **isFreeNow** () const
- bool **freeRoom** ([Reservation](#) *¤tRes)
try to free this room
- void **newDate** ([Date](#))
apply new [Date](#) to state of all reservations and respectively of the room availability
- bool **isFreeOnDate** ([Date](#)) const
see if this room is free in certain date
- bool **isFreeInPeriod** (const [DatePeriod](#) &period) const
see if this [Room](#) is free in particular period of time (it is free in all days of the period)
- bool **showReservationsInPeriod** (std::ostream &os, const [DatePeriod](#) &period) const
print to output stream info about the number of nights (if positive) in a period this [Room](#) has been taken
- bool **addReservation** ([String](#) name, [String](#) note, const [DatePeriod](#) &period)
try to add [Reservation](#) to this [Room](#)
- bool **closeForService** ([String](#) note, const [DatePeriod](#) &period)
try to add [Reservation](#) (about a maintenance) to this [Room](#)
- void **showActivity** () const
print to stdout information about this [Room](#) latest busyness
- void **writeToBinaryFile** (std::ofstream &ofs)
write the [Room](#) data into binary file opened by ofstream
- void **readDataFromBinary** (std::ifstream &ifis)
read the [Room](#) data from binary file opened by ifstream

3.7.1 Detailed Description

Class representing a room in hotel.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 Room()

```
Room::Room (
    unsigned n,
    unsigned bC )
```

Construct a new [Room](#) object.

Parameters

<i>n</i>	number of constructed Room
<i>bC</i>	number of beds in constructed Room

3.7.3 Member Function Documentation

3.7.3.1 addReservation()

```
bool Room::addReservation (
    String name,
    String note,
    const DatePeriod & period )
```

try to add [Reservation](#) to this [Room](#)

Parameters

<i>name</i>	name of the guest
<i>note</i>	note to this Reservation
<i>period</i>	period of time

Returns

true successfully added [Reservation](#)

false adding a [Reservation](#) failed (the room is not free in this [DatePeriod](#))

3.7.3.2 closeForService()

```
bool Room::closeForService (
    String note,
    const DatePeriod & period )
```

try to add [Reservation](#) (about a maintenance) to this [Room](#)

Parameters

<i>note</i>	note to this maintenance
<i>period</i>	period of time

Returns

true successfully added maintenance
false adding a maintenance failed (the room is not free in this [DatePeriod](#))

3.7.3.3 freeRoom()

```
bool Room::freeRoom (
    Reservation *& currentRes )
```

try to free this room

Returns

true sucesfully freed room
false room is already free

3.7.3.4 isFreeInPeriod()

```
bool Room::isFreeInPeriod (
    const DatePeriod & period ) const
```

see if this [Room](#) is free in particular period of time (it is free in all days of the period)

Parameters

<i>period</i>	period of time
---------------	----------------

Returns

true the room is free (in all days of the period)
false the room is not free (there is a day in period when the room is taken)

3.7.3.5 isFreeOnDate()

```
bool Room::isFreeOnDate (
    Date d ) const
```

see if this room is free in certain date

Returns

true the room is free
false the room is taken

3.7.3.6 readDataFromBinary()

```
void Room::readDataFromBinary (
    std::ifstream & ifs )
```

read the [Room](#) data from binary file opened by ifstream

Parameters

<i>ofs</i>	input stream connected to binary file
------------	---------------------------------------

3.7.3.7 showReservationsInPeriod()

```
bool Room::showReservationsInPeriod (
    std::ostream & os,
    const DatePeriod & period ) const
```

print to output stream info about the number of nights (if positive) in a period this [Room](#) has been taken

Parameters

<i>os</i>	output stream
<i>period</i>	period of time

Returns

true there has been taken for at least one night and info has been printed
false the room has been free during this period and no info has been printed

3.7.3.8 writeToBinaryFile()

```
void Room::writeToBinaryFile (
    std::ofstream & ofs )
```

write the [Room](#) data into binary file opened by ofstream

Parameters

<i>ofs</i>	output stream connected to binary file
------------	--

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

- Room.hpp
- Room.cpp

3.8 RoomAnalyzer Class Reference

supporting class to perform algorithms on the rooms in a building

```
#include <RoomAnalyzer.hpp>
```

Static Public Member Functions

- static void [suggest](#) ([HotelBuilding](#) &hB, unsigned beds, [DatePeriod](#) period)
print top DISPLAY rooms info based on suitability of a [Room](#) (desired number of beds and period of time)
- static void [soonestFreePeriod](#) (const [HotelBuilding](#) &hB, unsigned number, unsigned nights, [Date](#) today)
print soonest period when a particular room is free for particular number of nights

3.8.1 Detailed Description

supporting class to perform algorithms on the rooms in a building

3.8.2 Member Function Documentation

3.8.2.1 [soonestFreePeriod\(\)](#)

```
void RoomAnalyzer::soonestFreePeriod (
    const HotelBuilding & hB,
    unsigned number,
    unsigned nights,
    Date today ) [static]
```

print soonest period when a particular room is free for particular number of nights

Parameters

<i>hB</i>	HotelBuilding
<i>number</i>	ID of the Room
<i>nights</i>	length of a period
<i>today</i>	today's Date

3.8.2.2 [suggest\(\)](#)

```
void RoomAnalyzer::suggest (
    HotelBuilding & hB,
```

```
    unsigned beds,  
    DatePeriod period )    [static]
```

print top DISPLAY rooms info based on suitability of a [Room](#) (desired number of beds and period of time)

Parameters

<i>hB</i>	HotelBuilding
<i>beds</i>	desired number of beds
<i>period</i>	period of time

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

- RoomAnalyzer.hpp
- RoomAnalyzer.cpp

3.9 String Class Reference

Public Member Functions

- **String** (const char * _str="")
- **String** (const [String](#) &other)
- [String](#) & **operator=** (const [String](#) &other)
- size_t **size** () const
- size_t **capacity** () const
- char * **c_str** () const
- [String](#) **operator+** (const [String](#) &other) const
- [String](#) & **operator+=** (const [String](#) &other)
- **operator const char *** () const
- [String](#) & **shrink_to_fit** ()

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

- String.hpp
- String.cpp

Chapter 4

File Documentation

4.1 Constants.hpp

```
1 #ifndef __CONSTANTS_HPP
2 #define __CONSTANTS_HPP
3 #include <cstdint>
4
9 const unsigned daysFromBeginning[] = {0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334};
10
15 const size_t DISPLAY_WIDTH = 130;
16
21 const size_t COMMANDS = 8;
22
27 const size_t STRING_MAX_LENGTH = 128;
28
33 const size_t INIT_CAPACITY = 2;
34
39 const size_t DISPLAY = 5;
40
45 const char cmdArr[COMMANDS][2][STRING_MAX_LENGTH] = {
46     {"To make a reservation, enter "},
47     {"<reserve: [Room number] [Accommodation date] [Departure date] {Guest name[;]} {Note}>"},
48     {"To see list of free rooms for a particular date, enter "},
49     {"<available: [date]>"},
50     {"To free a room now, enter "},
51     {"<free: [Room number]>"},
52     {"To get report about the reservations of a room over a period of time, enter "},
53     {"<report: [From date] [To date]>"},
54     {"To request a room for guests, enter"},
55     {"<request: [minimal number of beds] [Accommodation date] [Departure date]>"},
56     {"To close a room for maintenance, enter"},
57     {"<maintenance: [room number] [From date] [To date] [Note]>"},
58     {"To see activity of all rooms, enter "}, {"<rooms:>"},
59     {"To see soonest date a room is free for some nights, enter "},
60     {"<plan: [Room number] [Number of nights]>"};
61
62 #endif
```

4.2 Date.hpp

```
1 #ifndef __DATE_HPP
2 #define __DATE_HPP
3 #include "Constants.hpp"
4 #include <iostream>
5 #include <fstream>
6 #include <ctime>
7
12 class Date
13 {
18     unsigned short day, month, year;
25     bool isValid() const;
33     bool isLeap(unsigned y) const;
34
35 public:
43     Date(unsigned short d = 1, unsigned short m = 1, unsigned short y = 1900) : day(d), month(m), year(y)
    {}
51     bool operator<(const Date other) const;
```

```

55     bool operator<=(const Date other) const;
63     bool operator>(const Date other) const;
67     bool operator>=(const Date other) const;
75     bool operator==(const Date other) const;
82     const char *operator() (char *buf) const;
89     int operator-(Date other) const;
95     Date &operator++();
96
102     static Date getToday();
103
111     friend std::istream &operator>(std::istream &is, Date &d);
119     friend std::ostream &operator<<(std::ostream &os, const Date &d);
120
126     void writeToBinaryFile(std::ofstream &ofs);
127
133     void readDataFromBinary(std::ifstream &if);
134 };
135
140 struct DatePeriod
141 {
146     Date from;
147
152     Date to;
153
159     unsigned length() const { return to - from; }
160
166     DatePeriod &operator++();
167
172     void readProper();
173 };
174
182 std::istream &operator>(std::istream &is, DatePeriod &dP);
183
184 #endif

```

4.3 Hotel.hpp

```

1  #ifndef __HOTEL_HPP
2  #define __HOTEL_HPP
3  #include "Types.hpp"
4  #include "Constants.hpp"
5  #include "String.hpp"
6  #include "Date.hpp"
7  #include "Room.hpp"
8  #include "Reservation.hpp"
9  #include "HotelBuilding.hpp"
10 #include <string>
11
19 String readFromIfstream(std::ifstream &if, size_t len);
20
25 class Hotel
26 {
31     String name;
32
37     static Date now;
38
43     HotelBuilding *building;
44
49     void writeToBinaryFile();
50
51 public:
52     Hotel() = delete;
53
59     Hotel(String hotelDataFile);
60     Hotel(const Hotel &) = delete;
61     Hotel &operator=(const Hotel &) = delete;
62
67     ~Hotel();
68
74     static Date today() { return now; }
75
81     String getName() const { return name; }
82
87     void nextDay();
88
100     bool reserveRoom(unsigned number, const DatePeriod &period, String name = "-", String note =
        "None\n");
101
107     Hotel &showAvailableRooms(std::ostream &, Date);
108
116     bool freeRoom(unsigned number);
117
124     Hotel &getReport(DatePeriod &period);

```

```

125
132     void searchRoom(unsigned minBeds, const DatePeriod &period) const;
133
143     bool serviceRoom(unsigned number, const DatePeriod &period, String note);
144
150     Hotel &showToday();
151
159     Hotel &seeRoomForNights(unsigned number, unsigned nights);
160
167     bool workDay();
168 };
169
170 #endif

```

4.4 HotelBuilding.hpp

```

1 #ifndef __HOTELBUILDING_HPP
2 #define __HOTELBUILDING_HPP
3 #include "Types.hpp"
4 #include "Constants.hpp"
5 #include "Room.hpp"
6 #include "RoomAnalyzer.hpp"
7 #include <fstream>
8
13 class HotelBuilding
14 {
19     Room **rooms;
24     size_t size;
25
26 public:
36     HotelBuilding(std::ifstream &if);
37     HotelBuilding(const HotelBuilding &other) = delete;
38     HotelBuilding &operator=(HotelBuilding &other) = delete;
43     ~HotelBuilding();
44
50     size_t getRoomCount() const { return size; }
51
59     Room *operator[](unsigned roomNumber) const;
60
66     void newDate(Date d);
67
78     void showAvailableRooms(std::ostream &os, Date d) const;
79
90     void createReport(DatePeriod &period) const;
91
98     void suggestRoom(unsigned beds, const DatePeriod &period);
99
105     void showRoomsStatesToday(Date today) const;
106
114     void showRoomForNights(unsigned number, unsigned nights, Date today) const;
115
121     void writeToBinaryFile(std::ofstream &ofs);
122
128     void readDataFromBinary(std::ifstream &if);
129
134     friend class RoomAnalyzer;
135 };
136
137 #endif

```

4.5 HotelInterface.hpp

```

1 #ifndef __HOTELINTERFACE_HPP
2 #define __HOTELINTERFACE_HPP
3 #include <iomanip>
4 #include <iostream>
5 #include "Constants.hpp"
6 #include "Hotel.hpp"
7
8 class HotelInterface
9 {
10 public:
16     static void createHeader(Hotel &H);
17
22     static void beginDay();
23 };
24
25 #endif

```

4.6 Reservation.hpp

```

1 #ifndef __RESERVATION_HPP
2 #define __RESERVATION_HPP
3 #include "Types.hpp"
4 #include "Room.hpp"
5 #include "Date.hpp"
6 #include "Hotel.hpp"
7 #include <cstring>
8 #include <cassert>
9 #include <fstream>
10 #include <string>
11 #include "Constants.hpp"
12
13 enum ReservationState
14 {
15     UNKNOWN = 0,
16     PAST,
17     ACTIVE,
18     FUTURE
19 };
20
21 class Reservation
22 {
23     String guestName;
24     String note;
25     DatePeriod period;
26     ReservationState state;
27     bool service;
28
29 public:
30     Reservation(String name, const DatePeriod &p, String n = "None.\n", bool s = false);
31     Reservation(const Reservation &) = delete;
32     Reservation &operator=(const Reservation &) = delete;
33
34     bool isActive() const { return state == ACTIVE; }
35
36     bool isPast() const { return state == PAST; }
37
38     bool isServiced() const { return service; }
39
40     Date getFrom() const { return period.from; }
41
42     Date getTo() const { return period.to; }
43
44     unsigned getNights() const { return period.length(); }
45
46     String getNote() const { return note; }
47
48     void onDate(Date d);
49
50     ReservationState stateOnDate(Date) const;
51
52     bool LeavingInAdvance(Date);
53
54     void writeToBinaryFile(std::ofstream &ofs);
55
56     void readDataFromBinary(std::ifstream &ifs);
57 };
58
59 std::ostream &operator<<(std::ostream &, const Reservation &);
60
61 #endif

```

4.7 Room.hpp

```

1 #ifndef __ROOM_HPP
2 #define __ROOM_HPP
3 #include <iostream>
4 #include <string>
5 #include "Types.hpp"
6 #include "Reservation.hpp"
7 #include "Hotel.hpp"
8 #include "Constants.hpp"
9
10 class Room
11 {
12     unsigned number;
13     unsigned bedCount;
14     Reservation **reservations;
15     size_t resCount, resCapacity;
16
17     Reservation **pastReservations;

```

```

46     size_t pastCount, pastCapacity;
47
52     void expand(Reservation **&arr, size_t &size, size_t &capacity);
57     void shrink(Reservation **&arr, size_t &size, size_t &capacity);
58
65     unsigned daysTakenInPeriod(const DatePeriod &period) const;
66
77     bool newReservation(String name, String note, const DatePeriod &period, bool service);
78
83     void moveToPast();
84
85 public:
92     Room(unsigned n, unsigned bC);
97     Room(const Room &) = delete;
102    Room &operator=(const Room &) = delete;
107    ~Room();
108
109    unsigned getNumber() const { return number; }
110    unsigned getBedCount() const { return bedCount; }
111    bool isFreeNow() const;
112
119    bool freeRoom(Reservation *&currentRes);
120
125    void newDate(Date);
126
133    bool isFreeOnDate(Date) const;
134
142    bool isFreeInPeriod(const DatePeriod &period) const;
143
152    bool showReservationsInPeriod(std::ostream &os, const DatePeriod &period) const;
153
163    bool addReservation(String name, String note, const DatePeriod &period);
164
173    bool closeForService(String note, const DatePeriod &period);
174
179    void showActivity() const;
180
186    void writeToBinaryFile(std::ofstream &ofs);
187
193    void readDataFromBinary(std::ifstream &if);
194 };
195
203 std::ostream &operator<<(std::ostream &os, const Room &R);
204
205 #endif

```

4.8 RoomAnalyzer.hpp

```

1  #ifndef __ROOMANALYZER_HPP
2  #define __ROOMANALYZER_HPP
3  #include "Types.hpp"
4  #include "Constants.hpp"
5  #include "HotelBuilding.hpp"
6  #include "Date.hpp"
7
12 class RoomAnalyzer
13 {
22     static void sortRoomsByScore(HotelBuilding &hB, unsigned *score, size_t from, size_t to);
23
31     static void sortRoomsByNumber(HotelBuilding &hB, size_t from, size_t size);
32
40     template <typename T>
41     static void swap(T &a, T &b);
42
43 public:
51     static void suggest(HotelBuilding &hB, unsigned beds, DatePeriod period);
52
61     static void soonestFreePeriod(const HotelBuilding &hB, unsigned number, unsigned nights, Date today);
62 };
63
64 template <typename T>
65 void RoomAnalyzer::swap(T &a, T &b)
66 {
67     T c = a;
68     a = b;
69     b = c;
70 }
71
72 #endif

```

4.9 String.hpp

```
1 #ifndef __STRING_HPP
2 #define __STRING_HPP
3 #include <iostream>
4
5 class String
6 {
7     char *str;
8     size_t m_size, m_capacity;
9     void copy(const String &other);
10    void expand(size_t newCap);
11
12 public:
13     String(const char *_str = "");
14     String(const String &other);
15     String &operator=(const String &other);
16     ~String();
17
18     size_t size() const { return m_size; }
19     size_t capacity() const { return m_capacity; }
20     char *c_str() const { return str; }
21     String operator+(const String &other) const;
22     String &operator+=(const String &other);
23     operator const char *() const { return c_str(); }
24     String &shrink_to_fit();
25 };
26
27 std::ostream &operator<<(std::ostream &os, const String &S);
28
29 #endif
```

4.10 Types.hpp

```
1 #ifndef __TYPES_HPP
2 #define __TYPES_HPP
3
4 class String;
5 class Date;
6 class Room;
7 class HotelBuilding;
8 class Reservation;
9 class RoomAnalyzer;
10 class Hotel;
11
12 #endif
```

Index

- addReservation
 - Room, [28](#)
- closeForService
 - Room, [28](#)
- createHeader
 - HotelInterface, [21](#)
- createReport
 - HotelBuilding, [18](#)
- Date, [5](#)
 - Date, [6](#)
 - getToday, [6](#)
 - operator<, [7](#)
 - operator<<, [9](#)
 - operator>, [8](#)
 - operator>>, [9](#)
 - operator(), [6](#)
 - operator++, [7](#)
 - operator-, [7](#)
 - operator==, [8](#)
 - readDataFromBinary, [8](#)
 - writeToBinaryFile, [9](#)
- DatePeriod, [10](#)
 - length, [11](#)
 - operator++, [11](#)
- freeRoom
 - Hotel, [13](#)
 - Room, [29](#)
- getFrom
 - Reservation, [23](#)
- getName
 - Hotel, [13](#)
- getNights
 - Reservation, [23](#)
- getNote
 - Reservation, [23](#)
- getReport
 - Hotel, [13](#)
- getRoomCount
 - HotelBuilding, [18](#)
- getTo
 - Reservation, [24](#)
- getToday
 - Date, [6](#)
- Hotel, [11](#)
 - freeRoom, [13](#)
 - getName, [13](#)
 - getReport, [13](#)
 - Hotel, [12](#)
 - reserveRoom, [14](#)
 - searchRoom, [14](#)
 - seeRoomForNights, [15](#)
 - serviceRoom, [15](#)
 - showAvailableRooms, [15](#)
 - showToday, [16](#)
 - today, [16](#)
 - workDay, [16](#)
- HotelBuilding, [17](#)
 - createReport, [18](#)
 - getRoomCount, [18](#)
 - HotelBuilding, [18](#)
 - newDate, [18](#)
 - operator[], [19](#)
 - readDataFromBinary, [19](#)
 - showAvailableRooms, [19](#)
 - showRoomForNights, [20](#)
 - showRoomsStatesToday, [20](#)
 - suggestRoom, [20](#)
 - writeToBinaryFile, [21](#)
- HotelInterface, [21](#)
 - createHeader, [21](#)
- isActive
 - Reservation, [24](#)
- isFreeInPeriod
 - Room, [29](#)
- isFreeOnDate
 - Room, [29](#)
- isPast
 - Reservation, [24](#)
- isServiced
 - Reservation, [24](#)
- LeavingInAdvance
 - Reservation, [25](#)
- length
 - DatePeriod, [11](#)
- newDate
 - HotelBuilding, [18](#)
- onDate
 - Reservation, [25](#)
- operator<
 - Date, [7](#)
- operator<<
 - Date, [9](#)

- operator>
 - Date, [8](#)
- operator>>
 - Date, [9](#)
- operator()
 - Date, [6](#)
- operator++
 - Date, [7](#)
 - DatePeriod, [11](#)
- operator-
 - Date, [7](#)
- operator==
 - Date, [8](#)
- operator[]
 - HotelBuilding, [19](#)
- readDataFromBinary
 - Date, [8](#)
 - HotelBuilding, [19](#)
 - Reservation, [25](#)
 - Room, [29](#)
- Reservation, [22](#)
 - getFrom, [23](#)
 - getNights, [23](#)
 - getNote, [23](#)
 - getTo, [24](#)
 - isActive, [24](#)
 - isPast, [24](#)
 - isServiced, [24](#)
 - LeavingInAdvance, [25](#)
 - onDate, [25](#)
 - readDataFromBinary, [25](#)
 - Reservation, [23](#)
 - stateOnDate, [26](#)
 - writeToBinaryFile, [26](#)
- reserveRoom
 - Hotel, [14](#)
- Room, [26](#)
 - addReservation, [28](#)
 - closeForService, [28](#)
 - freeRoom, [29](#)
 - isFreeInPeriod, [29](#)
 - isFreeOnDate, [29](#)
 - readDataFromBinary, [29](#)
 - Room, [27](#)
 - showReservationsInPeriod, [30](#)
 - writeToBinaryFile, [30](#)
- RoomAnalyzer, [31](#)
 - soonestFreePeriod, [31](#)
 - suggest, [31](#)
- searchRoom
 - Hotel, [14](#)
- seeRoomForNights
 - Hotel, [15](#)
- serviceRoom
 - Hotel, [15](#)
- showAvailableRooms
 - Hotel, [15](#)
- HotelBuilding, [19](#)
- showReservationsInPeriod
 - Room, [30](#)
- showRoomForNights
 - HotelBuilding, [20](#)
- showRoomsStatesToday
 - HotelBuilding, [20](#)
- showToday
 - Hotel, [16](#)
- soonestFreePeriod
 - RoomAnalyzer, [31](#)
- stateOnDate
 - Reservation, [26](#)
- String, [33](#)
- suggest
 - RoomAnalyzer, [31](#)
- suggestRoom
 - HotelBuilding, [20](#)
- today
 - Hotel, [16](#)
- workDay
 - Hotel, [16](#)
- writeToBinaryFile
 - Date, [9](#)
 - HotelBuilding, [21](#)
 - Reservation, [26](#)
 - Room, [30](#)