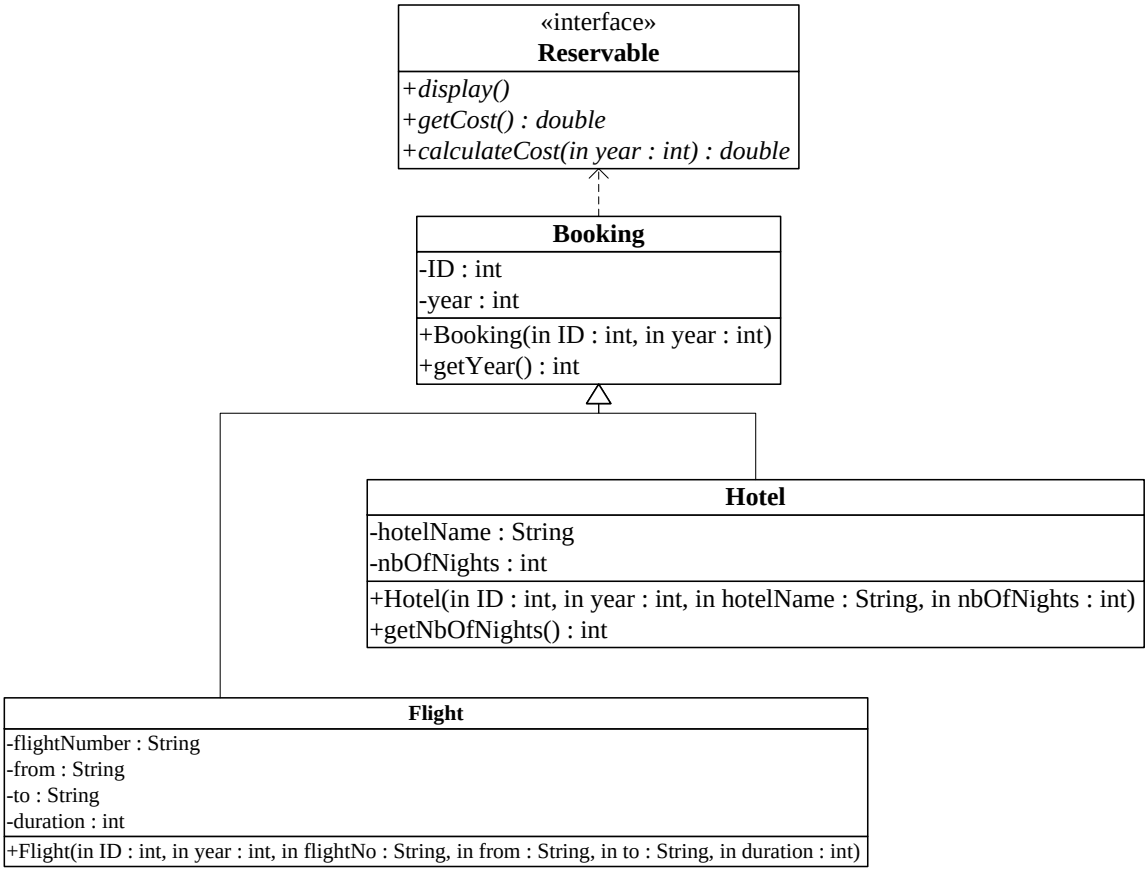
King Saud University
College of Computer and Information Sciences
Department of Computer Science
CSC113 – Computer Programming II – Midterm 2 Exam – Spring 2016

Exercise1:



Reservable Interface***:***

- Methods:
  - ***display()*:** this method displays **<u>all</u>** attributes of the Reservable object.
  - ***getCost()*:** this method returns the cost of the Reservable object calculated by the method calculateCost and using the attribute year of the Reservable object.
  - ***calculateCost(year: int)*:** this method calculates and returns the cost of the Reservable object. It is calculated as follows:
    - For ***Flight Booking***: cost = year / 10 + Flight Duration * 10 .
    - For ***Hotel Booking:*** if the Hotel Booking is done in or before 2010, the cost is 2000 SAR. For any year after 2010 the cost is 10 % greater than the cost of the previous year.

      Cost for current year = 2000 SAR if current year is less or equal to 2010.

      Otherwise: cost for current year = 1.1 * cost for previous year.

Booking class***:***

- Attributes:
  - ***ID***: the ID of the Booking.
  - ***year:*** the year of the Booking.
- Methods:
  - ***Booking (ID:*** *int****, year:*** *int****)***: constructor
  - ***getYear()*****:** this method returns the year of the Booking.

Hotel class

- Attributes:
  - ***hotelName:*** the name of the Hotel.
  - ***nbOfNights:*** the number of nights spent in the Hotel.
- Methods:
  - ***Hotel (ID:*** *int****, year:*** *int****, hotelName:*** *String****, nbOfNights:*** *int****)***: constructor.
  - ***getNbOfNights()***: this method returns the number of nights spent in the Hotel.

Flight class

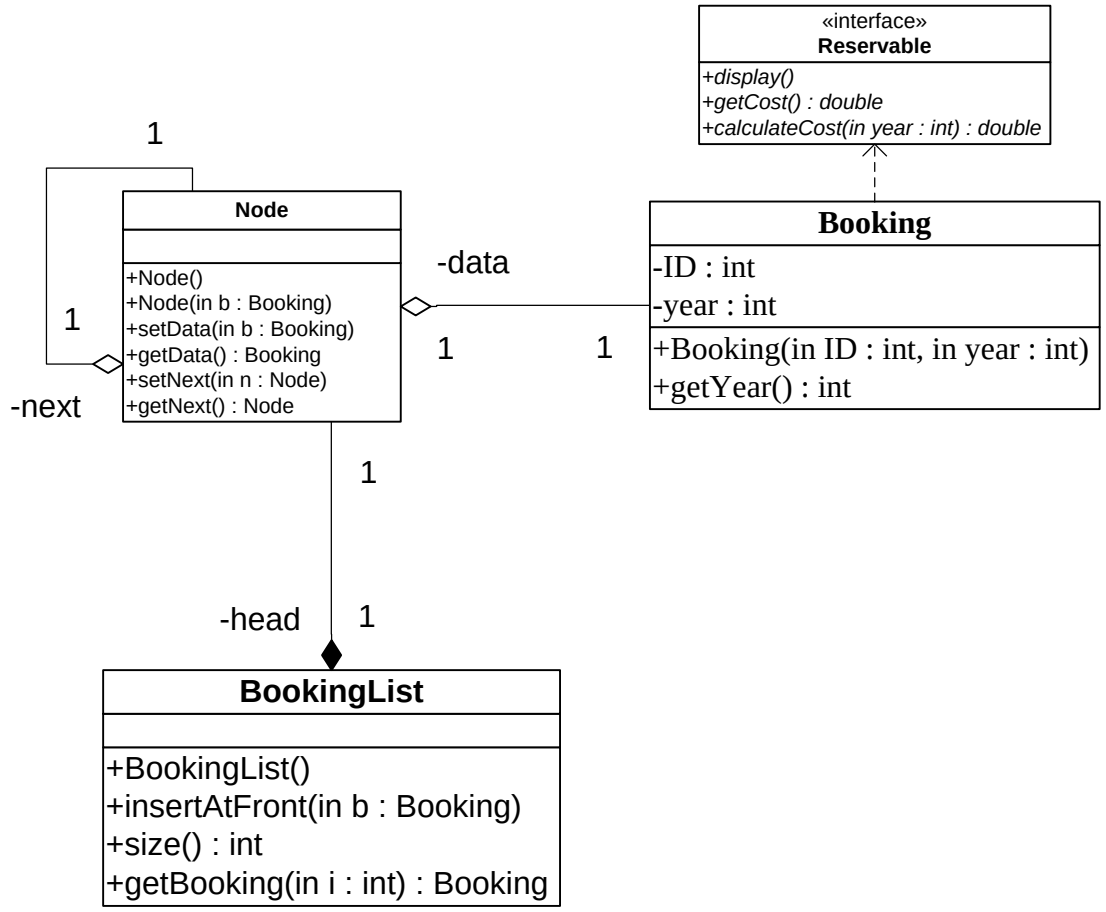
- Attributes:
  - ***flightNumber:*** the Flight number.
  - ***from***: the name of the departure Airport.
  - ***to:*** the name of the arrival Airport.
  - ***duration:*** the Flight's duration (in minutes).
- Methods:
  - ***Flight (ID:*** *int****, year:*** *int****, flightNo:*** *String****, from:*** *String****, to:*** *String****, duration:*** *int****)***: constructor.

QUESTION: Translate into Java code:

- the Interface ***Reservable***,
- the class ***Booking***
- and the class ***Hotel***.
- For the method ***calculateCost*** , propose 2 solutions (an **iterative solution** and a **recursive solution**).

Exercise 2:

Let's consider the same class ***Booking*** described in exercise 1.



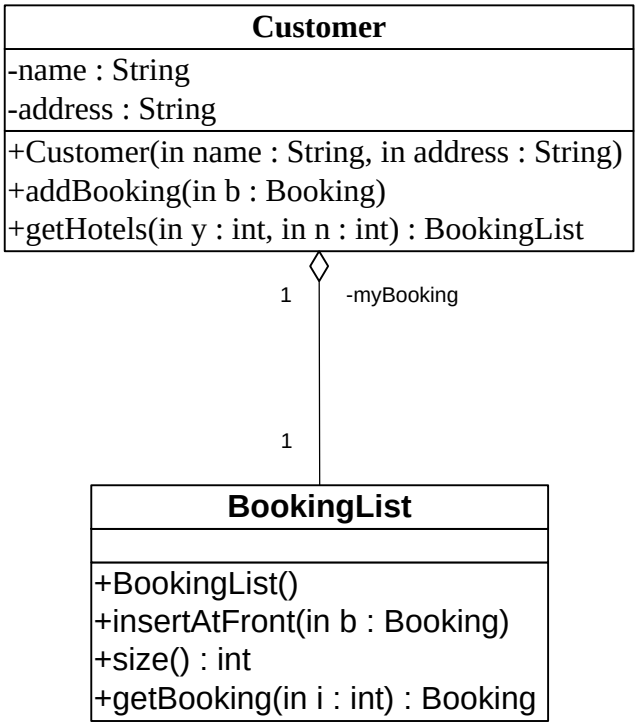
BookingList class***:***

- Methods:
  - ***BookingList ()***: constructor.
  - ***insertAtFront (b: Booking***): this method insert the Booking ***b*** at the beginning of the list.
  - ***size ()*:** this method returns the number of elements of the list.
  - ***getBooking(i: int*):** this method returns the Booking object stored in the node at position ***i***. The position of the first node is 1. If the parameter ***i*** is less than 1 or greater than the number of elements of the list, this method throws an ***Exception*** with the message "Position out of bounds".

QUESTION: Translate into Java code the class ***BookingList.***

Exercise 3:

Let's consider the same class ***BookingList*** described in exercise 2.



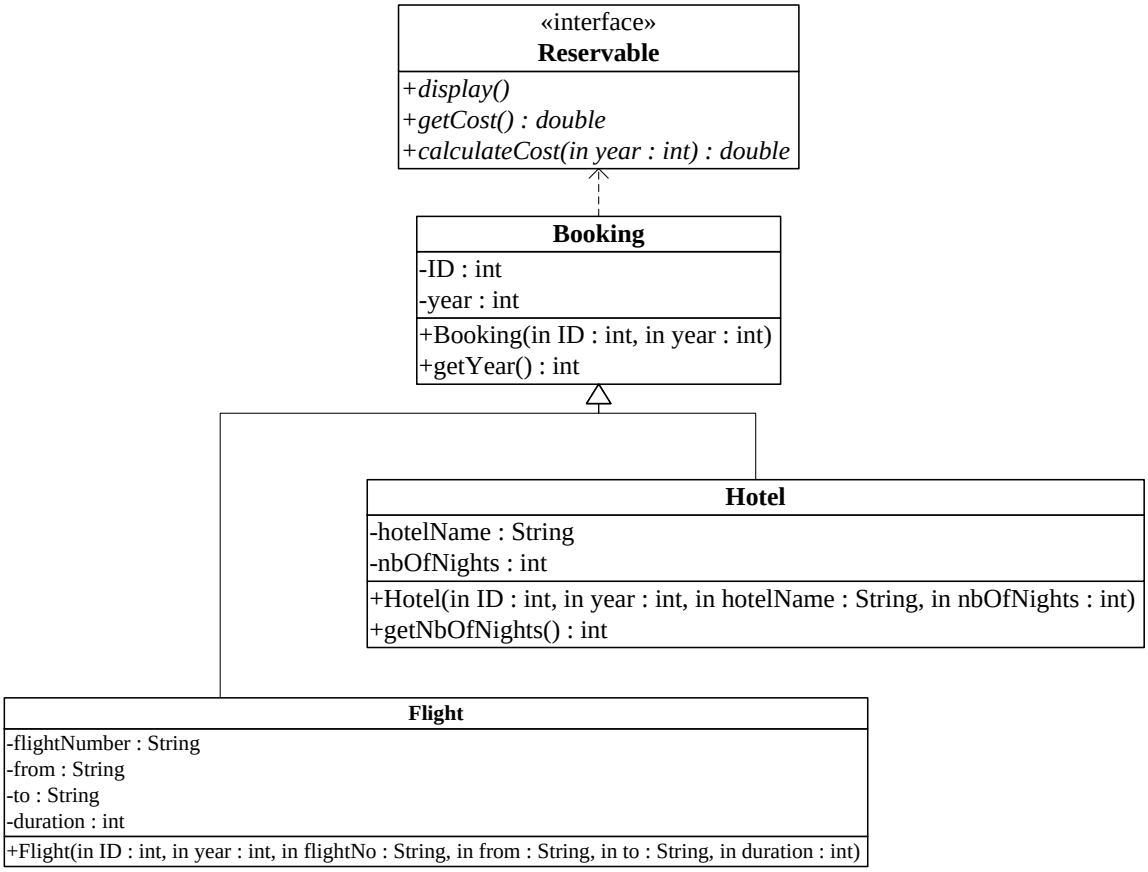
Customer class***:***

- Attributes:
  - ***name***: the customer name.
  - ***address :*** the address of the customer.
- Methods:
  - ***Customer (name:*** *String****, address:*** *String****)***: constructor.
  - ***addBooking (b: Booking***): this method adds the Booking ***b*** to the customer.
  - ***getHotels(y:*** *int****, n:*** *int****):*** this method returns a BookingList object containing all Hotel Bookings in the year ***y*** and having the number of nights greater than ***n***.

QUESTION: Translate into Java code the class ***Customer.***

King Saud University
College of Computer and Information Sciences
Department of Computer Science
CSC113 – Computer Programming II – Midterm 2 Exam – Spring 2016

Exercise1:



Reservable Interface***:***

- Methods:
  - ***display()*:** this method displays **<u>all</u>** attributes of the Reservable object.
  - ***getCost()*:** this method returns the cost of the Reservable object calculated by the method calculateCost and using the attribute year of the Reservable object.
  - ***calculateCost(year: int)*:** this method calculates and returns the cost of the Reservable object. It is calculated as follows:
    - For ***Flight Booking***: cost = year / 10 + Flight Duration * 10 .
    - For ***Hotel Booking:*** if the Hotel Booking is done in or before 2010, the cost is 2000 SAR. For any year after 2010 the cost is 10 % greater than the cost of the previous year.

      Cost for current year = 2000 SAR if current year is less or equal to 2010.

      Otherwise: cost for current year = 1.1 * cost for previous year.

Booking class***:***

- Attributes:
  - ***ID***: the ID of the Booking.
  - ***year:*** the year of the Booking.
- Methods:
  - ***Booking (ID:*** *int****, year:*** *int****)***: constructor
  - ***getYear()*****:** this method returns the year of the Booking.

Hotel class

- Attributes:
  - ***hotelName:*** the name of the Hotel.
  - ***nbOfNights:*** the number of nights spent in the Hotel.
- Methods:
  - ***Hotel (ID:*** *int****, year:*** *int****, hotelName:*** *String****, nbOfNights:*** *int****)***: constructor.
  - ***getNbOfNights()***: this method returns the number of nights spent in the Hotel.

Flight class

- Attributes:
  - ***flightNumber:*** the Flight number.
  - ***from***: the name of the departure Airport.
  - ***to:*** the name of the arrival Airport.
  - ***duration:*** the Flight's duration (in minutes).
- Methods:
  - ***Flight (ID:*** *int****, year:*** *int****, flightNo:*** *String****, from:*** *String****, to:*** *String****, duration:*** *int****)***: constructor.

QUESTION: Translate into Java code:

- the Interface ***Reservable***,
- the class ***Booking***
- and the class ***Hotel***.
- For the method ***calculateCost*** , propose 2 solutions (an **iterative solution** and a **recursive solution**).

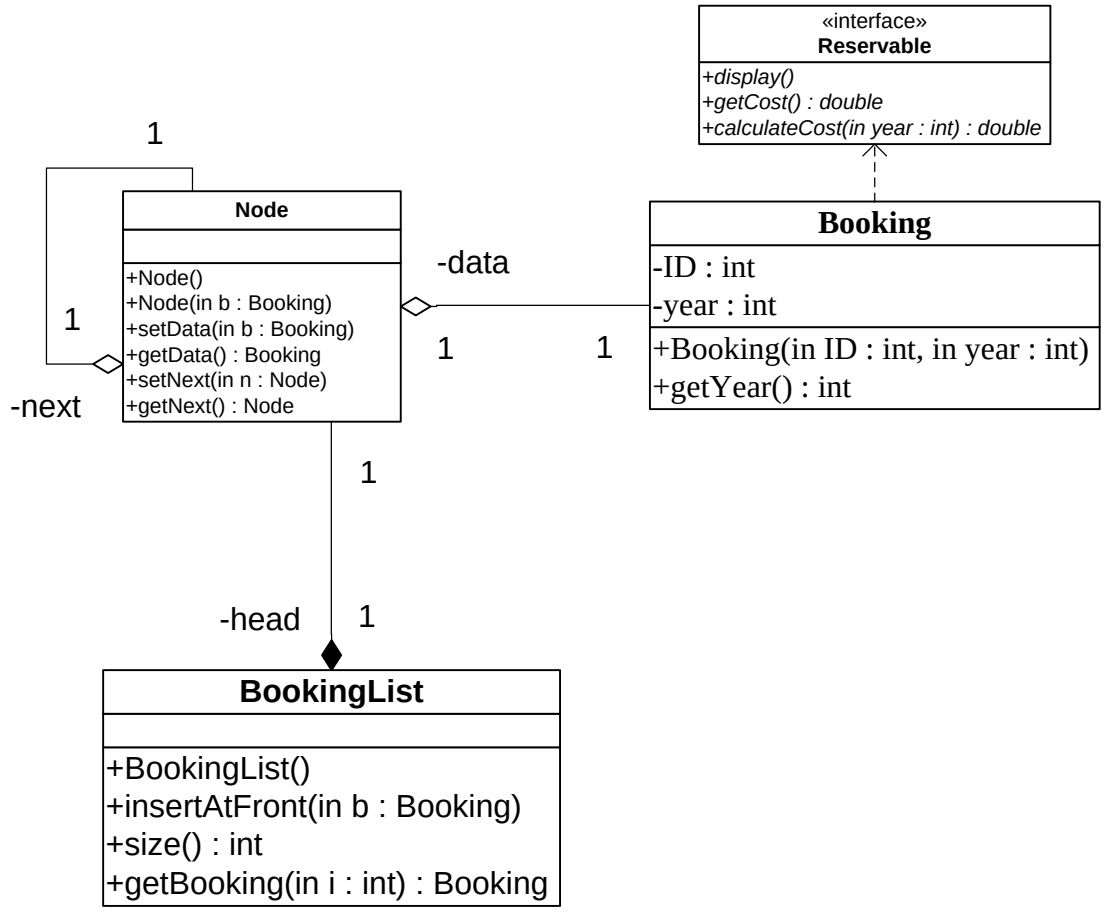
Solution of exercise1:

```
public interface Reservable {        …….. 0.5 ..... 2
        void display();              …….. 0.5
        double getCost();            …….. 0.5
        double calculateCost(int year); …….. 0.5
}

public abstract class Booking implements Reservable { …….. 1 + 1
..... 7
                private int ID;
                private int year;

                public Booking(int ID, int year) { …….. 1
                        this.ID = ID;
                        this.year = year;
                }

                public int getYear() { …….. 1
                        return year;
                }

                public double getCost() { …….. 1
                        return calculateCost(year); …….. 1
                }

                public void display() { …….. 1
                        System.out.println("ID: " + ID);
                        System.out.println("Year: " + year);
                }
}
```

```
public class Hotel extends Booking{ …….. 1          ……. 12
        private String hotelName;
        private int nbOfNights;

        public Hotel(int ID, int year, String hotelName, int nbOfNights){
                super(ID, year);  …….. 1
                this.hotelName = hotelName; …….. 0.5
                this.nbOfNights = nbOfNights; …….. 0.5
        }

        public int getNbOfNights() { …….. 1
                return nbOfNights;
        }

       public double calculateCost(int year) { ……..Iterative Solution
                double result = 2000; …….. 0.5
                for (int y = 2011; y <= year; y++) …….. 1
                        result += 1.1 * result; …….. 1
                return result; …….. 0.5
        }


        public double calculateCost(int year) { ……Recursive Solution
                if (year <= 2010) return 2000; …….. 1
                return 1.1 * calculateCost(year-1); …….. 1+1
        }

        public void display(){…….. 1
                super.display ();          …….. 1
                System.out.println("Hotel name: " + hotelName);
                System.out.println("# of nights: " + nbOfNights);
        }
}
```

Exercise 2:

Let's consider the same class ***Booking*** described in exercise 1.



BookingList class***:***

- Methods:
  - ***BookingList ()***: constructor.
  - ***insertAtFront (b: Booking***): this method insert the Booking ***b*** at the beginning of the list.
  - ***size ()*:** this method returns the number of elements of the list.
  - ***getBooking(i: int*):** this method returns the Booking object stored in the node at position ***i***. The position of the first node is 1. If the parameter ***i*** is less than 1 or greater than the number of elements of the list, this method throws an ***Exception*** with the message "Position out of bounds".

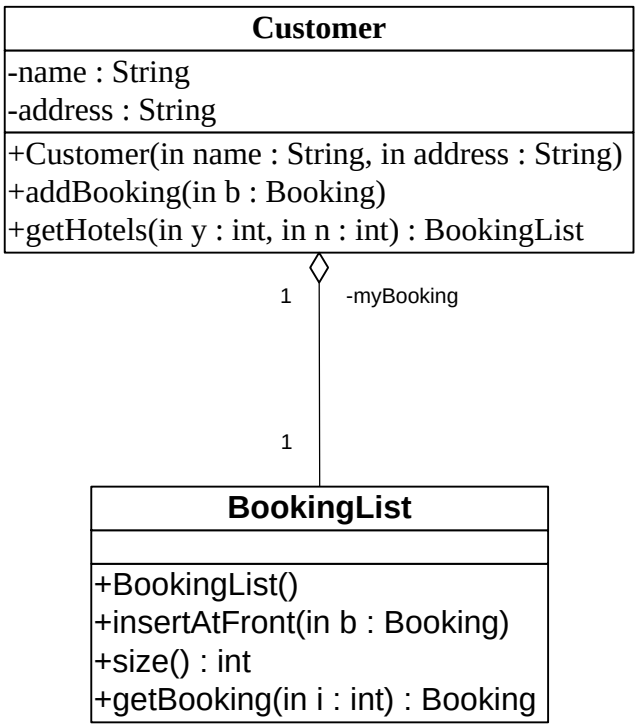
QUESTION: Translate into Java code the class ***BookingList.***

Solution of exercise 2:

```
public class BookingList {          ..... 17
        private Node head; ..... 1

        public BookingList(){
              head = null; ..... 1
        }

        public void insertAtFront(Booking b) {
                Node nn = new Node(b); ..... 1
                nn.setNext(head); ..... 1
                head = nn; ..... 1
        }
        public void insertAtFront(Booking b) {
                Node nn = new Node();       nn.setData(b);       ..... 1
                nn.setNext(head); ..... 1
                head = nn; ..... 1
        }

        public int size() {
                int count = 0; ..... 1
                Node p = head; ..... 1
                while (p != null) { ..... 1
                        count++; ..... 1
                        p = p.getNext();..... 1
                }
                return count; ..... 1
        }

       public Booking getBooking(int i) throws Exception{ ..... 1
                if (i < 1 || i > size())
                     throw new Exception("Position out of bounds"); ..... 1
                Node p = head; ..... 1
                for (int j=1; j < i; j++) ..... 1
                        p = p.getNext();       ..... 1
                return p.getData(); ..... 1
        }
}
```

Exercise 3:

Let's consider the same class ***BookingList*** described in exercise 2.



Customer class***:***

- Attributes:
  - ***name***: the customer name.
  - ***address :*** the address of the customer.
- Methods:
  - ***Customer (name:*** *String****, address:*** *String****)***: constructor.
  - ***addBooking (b: Booking***): this method adds the Booking ***b*** to the customer.
  - ***getHotels(y:*** *int****, n:*** *int****):*** this method returns a BookingList object containing all Hotel Bookings in the year ***y*** and having the number of nights greater than ***n***.

QUESTION: Translate into Java code the class ***Customer.***

Solution of exercise 3:

```
public class Customer{   ..... 13
        private String name;
        private String address;
        private BookingList myBooking;  ..... 1

        public Customer(String name, String address){
                this.name = name;
                this.address = address;
                myBooking  = new BookingList();  ..... 1
        }

        public void addBooking(Booking b){
                myBooking.insertAtFront(b);  ..... 1
        }

        public BookingList getHotels(int y, int n) throws Exception{ ..... 1
                BookingList result = new BookingList();..... 1
                for (int i=1; i <= myBooking.size(); i++){ ..... 1
                        Booking b = myBooking.getBooking(i); ..... 1
                        if ( b instanceof Hotel ..... 1
                            && b.getYear() == y ..... 1
                            && ((Hotel)b).getNbOfNights() > n) ..... 1+1
                                result.insertAtFront(b); ..... 1
                }
                return result; ..... 1
        }

        public BookingList getHotels(int y, int n) {
                BookingList result = new BookingList();..... 1
                for (int i=1; i <= myBooking.size(); i++){ ..... 1
                        try { ..... 0.5
                            Booking b = myBooking.getBooking(i); ..... 1
                            if (      b instanceof Hotel ..... 1
                                    && b.getYear() == y ..... 1
```

```
                                && ((Hotel)b).getNbOfNights() > n) ..... 1+1
                                    result.insertAtFront(b); ..... 1
                        }
                        catch(Exception e) { System.out.println(e.getMessage());} ..... 0.5
                    }
                    return result; ..... 1
            }
}
```