Restaurants

Develop a program that allows managing a restaurant chain.

All the classes are located in the package `restaurantChain`. The class `MainClass` in package *main* provides some examples of usage of the main methods.

The JDK documentation is available in a local server.

R1: Restaurant chain

The class `Chain` represents a chain of restaurants and provides the following methods.

The method `addRestaurant()` allows adding a new restaurant in the chain, and accepts two arguments: the name of the restaurant and the name of the tables available in the restaurant; if the name is already taken an *InvalidName* exception is raised.

The method `getRestaurant()` accepts the name of a restaurant and returns the relative object of type `Restaurant`; if the name does not correspond to a restaurant present in the chain an *InvalidName* exception is raised.

R2: Restaurants

The class *Restaurant* represents a restaurant and provide the following methods.

The method `getName()` returns the name of the restaurant.

The method `addMenu()` allows adding a menu to the restaurant listing, it accepts two arguments: the name of the menu and the relative price; if the name of the menu is already taken an *InvalidName* exception is raised.

The method `reserve()` allows a group of customers to reserve some tables; it receives two arguments: the name of the group representative and the number of customers in the group; if the name of the representative is already taken an *InvalidName* exception is raised.
The customers are seated at tables with four seats each, that can are reserved for the group. If the available tables are not enough, the operation is rejected.
The method returns the number of tables reserved for the group (zero if not enough tables are available).

The method `getRefused()` returns the total number of customers that could not be accommodated due to lack of tables.

The method `getUnusedTables()` returns the number of tables in the restaurant that are currently not reserved.

R3: Orders

The class *Restaurant* allows managing the orders and the payments.

The method `order()` allows a group of customers, sitting the tables, to place an order. The methods accepts the name of the group representative and a sequence of menu names; if the name of the representative is not among those who reserved the tables throws an *InvalidName* exception.
The order is accepted only if the number of menus (present in the sequence) is greater or equal to the number of customers belonging to the group; the method returns a boolean value indicating whether the order has been accepted.
If the names of the menus in the order are not all in the listing of the restaurant, the method throws an *InvalidName* exception.

The method `getUnordered()` returns the list, in alphabetic order, of the names of the group representatives having tables that have not yet placed an order.

The method `pay()` accepts the name of a group representative and allow him/her to pay the order (that is the sum of the ordered menus); if the name of the representative is not among those who reserved some tables an *InvalidName* exception is raised.
The methods returns the total amount payed (zero if the group has placed no orders yet).

The method `getUnpaid()` returns the list, in alphabetic order, of the names of the representatives of the groups that have not payed yet.

The method `getIncome()` returns the total amout received in payments by the restaurant.

R4: Reports

The following methods of the *Chain* class allow to compare the trends of different restaurants belonging to the chain.

The method `sortByIncome()` returns the list of restaurants sorted by decreasing amount of received payments.

The method `sortByRefused()`b returns the list of restaurants sorted by increasing number of people whose reservation could not be accepted due to lack of tables.

The method `sortByUnusedTables()` returns the list of restaurants sorted by ascending number of unused tables.