**Especificación TP C**



Gonzalo de Achaval

Ignacio de la Vega



EJERCICIO 1

ADT: ***Invoice***

Description: Represents the invoice of a Cart.

*Constructors*:

* **newInvoice:** id x toPay x maxCapacity -> Invoice
  + Creates an invoice with its respective values.
  + Precondition:
    - Id number must be a positive integer which is specific to each invoice.
    - toPay must be a positive number or 0.
    - MaxCapacity must be a positive integer.
  + Postcondition: an invoice is created.

*Modifiers*:

* **addInvoiceLine:** Invoice x InvoiceLine -> void
  + Adds an invoiceLine to an Invoice’s list containing them.
  + Precondition:
    - Receives a non null invoice and invoice line
    - InvoiceLine capacity must not be filled
  + Postcondition: invoice with added invoiceLine

*Destroyer*:

* **freeInvoice**: Invoice -> void
  + Frees the memory allocated for an invoice.
  + Precondition:
    - Receives a non null invoice
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

ADT: ***Label***

Description: Represents the label of an Appliance

*Constructors*:

* **newLabel:** id x name-> Label
  + Creates an appliance with its respective values.
  + Precondition:
    - Id number must be a positive integer which is specific to each label.
    - name must be a non null array of characters
  + Postcondition: a label is created.

*Destroyer*:

* **freeLabel**: Label -> void
  + Frees the memory allocated for a specific label
  + Precondition:
    - receives a non null label.
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

ADT: ***Invoice Line***

Description: Represents the invoiceLine of an Appliance.

*Constructors*:

* **newInvoiceLine:** quantity x article -> InvoiceLine
  + Creates an invoice line with its respective values.
  + Precondition:
    - Quantity must be a positive integer
    - Article must be a non null array of characters.
  + Postcondition: an invoiceLine is created.

*Destroyer*:

* **freeInvoiceLine**: InvoiceLine -> void
  + Frees the memory allocated for an invoiceLine.
  + Precondition:
    - receives a non null invoiceLine
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

ADT: ***Cart***

Description: Represents a Cart that can produce Invoices.

*Constructors*:

* **newCart:** id -> Cart
  + Creates a Cart with a maximum capacity of 10 appliances.
  + Precondition:
    - Id number must be a positive integer which is specific to each Cart.
  + Postcondition: a Cart is created.

*Destroyer*:

* **freeCart**: Cart -> void
  + Frees the memory allocated for a Cart.
  + Precondition:
    - receives a non null Cart
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

*Modifiers*:

* **addToCart:** Cart x Appliance -> void
  + Adds an appliance to the Cart.
  + Precondition:
    - Recieves a non null appliance and cart
  + Postcondition: cart with added appliance.
* **growCart**: Cart -> void
  + Duplicates the maximum capacity of a Cart and does the same for its allocated memory.
  + Precondition:
    - Recieves a non null cart
    - Language support to dynamically manage memory
  + Postcondition:
    - Cart with increased capacity and allocated memory.
* **eraseAppliance**: Cart x Appliance -> void
  + Erases an appliance from the cart
  + Precondition:
    - Recieves a non null cart and appliance
    - Appliance must exist in cart
  + Postcondition:
    - Cart with an appliance less.
* **finishShopping:** Cart -> Invoice
  + Returns an invoice with the total prices of every appliance
  + Precondition:
    - Non null cart
  + Postcondition:
    - new Invoice

ADT: ***LineCart***

Description: Represents a LineCart

*Constructors*:

* **newLineCart:** id x Appliance -> LineCart
  + Creates a LineCart containing an appliance
  + Precondition:
    - Id number must be a positive integer which is specific to each Cart.
    - Non null appliance
  + Postcondition: a LineCart is created

*Destroyer*:

* **freeLineCart**: LineCart -> void
  + Frees the memory allocated for a LineCart.
  + Precondition:
    - receives a non null LineCart
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

ADT: ***Appliance***

Description: Represents an Appliance with all of its attributes.

*Constructors*:

* **newAppliance:** name x model x price x discount x Provider -> Appliance
  + Creates an Appliance with its defined attributes
  + Precondition:
    - Name and model should be an array of characters.
    - Price and discount positive numbers
    - Discount maximum = 100.
  + Postcondition: an Appliance is created

*Destroyer*:

* **freeAppliance:** Appliance -> void
  + Frees the memory allocated for an Appliance.
  + Precondition:
    - receives a non null Appliance
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

*Analyzers:*

* **compareTo:** Appliance X Appliance -> int
  + Compares two appliances by its id
  + Precondition:
    - receives two non null Appliances
  + Postcondition: 1 if the Appliances’ id are equal, 0 if not.

ADT: ***Catalogue***

Description: Represents a Catalogue with all of its attributes.

*Constructors*:

* **newCatalogue:** id X name X discount-> Catalogue
  + Creates a Catalogue with its defined attributes
  + Precondition:
    - id and name should be an array of characters.
    - Discount positive numbers
    - Discount maximum = 100.
  + Postcondition: a Catalogue is created

*Destroyer*:

* **freeCatalogue:** Catalogue -> void
  + Frees the memory allocated for a Catalogue.
  + Precondition:
    - Receives a non null Catalogue
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

*Modifiers*:

* **addAppliance:** Appliance X Catalogue -> void
  + Adds an appliance to the array of appliances that catalogue contains.
  + Precondition:
    - Receives a non null appliance and catalogue
  + Postcondition:
    - Catalogue with added appliance
* **growCatalogue**: Catalogue -> void
  + Duplicates the maximum capacity of a Catalogue and does the same for its allocated memory.
  + Precondition:
    - Recieves a non null catalogue
    - Language support to dynamically manage memory
  + Postcondition:
    - Catalogue with increased capacity and allocated memory.
* **removeAppliance**: Catalogue x Appliance -> void
  + Erases an appliance from the catalogue
  + Precondition:
    - Recieves a non null catalogue and appliance
    - Appliance must exist in catalogue
  + Postcondition:
    - Catalogue with an appliance less.

ADT: ***Provider***

Description: Represents a Provider with all of its attributes.

*Constructors*:

* **newProvider:** description X name X direction X city X phone X web X Manufacturer-> Provider
  + Creates a Provider with its defined attributes
  + Precondition:
    - id, name, description, direction, city, phone and web should be an array of characters.
    - Non null Manufacturer
  + Postcondition: a Provider is created

*Destroyer*:

* **freeProvider:** Provider -> void
  + Frees the memory allocated for a Provider.
  + Precondition:
    - Receives a non null Provider
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

*Modifiers*:

* **askForAppliances:** Provider X Stock -> void
  + Asks a manufacturer for 15 appliances in case the provider doesn’t have any. Adds 10 to the stock and subtracts 5 from the provider.
  + Precondition:
    - Non null provider and invoiceLine
  + Postcondition:
    - Provider with updated appliances.

ADT: ***Manufacturer***

Description: Represents a Manufacturer with all of its attributes.

*Constructors*:

* **newManufactuer:** description X name X direction X city X phone X web -> Manufacturer
  + Creates a Manufacturer with its defined attributes
  + Precondition:
    - id, name, description, direction, city, phone and web should be an array of characters.
  + Postcondition: a Manufacturer is created

*Destroyer*:

* **freeManufacturer:** Manufacturer -> void
  + Frees the memory allocated for a Manufacturer.
  + Precondition:
    - Receives a non null Manufacturer
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

*Modifiers*:

* **createAppliance:** Manufacturer X Provider X quantity -> void
  + Creates an appliance. Represents it in the manufacturer and provider by adding the quantity to a variable.
  + Precondition:
    - Non null Manufacturer and Provider
    - Quantity > 0
  + Postcondition:
    - Manufacturer and Provider with updated appliances.

ADT: ***Stock***

Description: Represents stock of an article

*Constructors*:

* **newStock:** id x article -> Stock
  + Creates a Stock with its defined attributes
  + Precondition:
    - ID and article must be an array of characters.
  + Postcondition: a Stock is created

*Destroyer*:

* **freeStock:** Stock -> void
  + Frees the memory allocated for a Stock.
  + Precondition:
    - Receives a non null Stock
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

EJERCICIO 2

ADT: ***Accessory***

Description: Represents a purchasable accessory for a camera

*Constructors*:

* **newAccessory:** accessoryType x comment x code -> Accessory
  + Creates an accessory with its respective values.
  + Precondition:
    - Code must be a positive integer which is UNIQUE to each product (accessory or camera)
    - Accessory type must be 1 or 2 (more can be added if desired)
    - Comment must be an array of chars.
  + Postcondition: an accessory is created.

*Destroyer*:

* **freeAccessory**: Accessory -> void
  + Frees the memory allocated for an Accessory.
  + Precondition:
    - Receives a non null Accessory
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

ADT: ***Camera***

Description: Represents a purchasable camera

*Constructors*:

* **newCamera:** megaPixels x LCDScreen x opticZoom x type x code -> Camera
  + Creates an accessory with its respective values.
  + Precondition:
    - Code must be a positive integer which is UNIQUE to each product (accessory or camera)
    - Camera type must be 1 or 2 (more can be added if desired)
    - MegaPixels, LCDScreen and opticZoom must be positive numbers. Only opticZoom can be 0.
  + Postcondition: a camera is created.

*Destroyer*:

* **freeCamera**: Camera -> void
  + Frees the memory allocated for a Camera
  + Precondition:
    - Receives a non null Camera.
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

*Modifiers*:

* **addAccessoryToCamera:** Camera x Accessory -> void
  + Adds an accessory to an array of accessories the camera contains, which has a maximum capacity that can only be modified internally to the code.
  + Precondition:
    - Receives a non null Camera and a non null accessory.
  + Postcondition: camera with added accessory

ADT: ***Manufacturer***

Description: Represents someone who manufactures products (accessories and cameras).

*Constructors*:

* **newManufacturer:** name x code -> Manufacturer
  + Creates a manufacturer with its respective values.
  + Precondition:
    - Code must be a positive integer which is unique to each manufacturer.
    - Name must be an array of chars.
  + Postcondition: a manufacturer is created.

*Destroyer*:

* **freeManufacturer**: Manufacturer -> void
  + Frees the memory allocated for a Manufacturer
  + Precondition:
    - Receives a non null Manufacturer
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

ADT: ***Product***

Description: Represents a product: a camera or an accessory.

*Constructors*:

* **newProduct:** name x code x price x photo x provider x manufacturer -> Product
  + Creates a product with its respective values.
  + Precondition:
    - Name must be an array of chars.
    - Photo must be an array of chars with an URL to that photo
    - Price and code must be positive integers
    - Provider and manufacturer must be non null
  + Postcondition: a product is created.

*Destroyer*:

* **freeProduct**: Product -> void
  + Frees the memory allocated for a Product
  + Precondition:
    - Receives a non null Product
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

ADT: ***Provider***

Description: Represents someone who provides products to clients (accessories and cameras), made by a manufacturer.

*Constructors*:

* **newProvider:** CIF x name x phone x fax x address x location x province x country x postalCode -> Provider
  + Creates a provider with its respective values.
  + Precondition:
    - Every attribute must be an array of chars.
    - CIF must be 3 letter long and respect the [*incoterm*](https://en.wikipedia.org/wiki/Incoterms) convention
    - Postal code must correspond with address
    - Address, location, province and country must exist
  + Postcondition: a provider is created.

*Destroyer*:

* **freeProvider**: Provider -> void
  + Frees the memory allocated for a Provider
  + Precondition:
    - Receives a non null Provider
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

ADT: ***Registered User***

Description: Represents a user who can buy products

*Constructors*:

* **newRegisteredUser:** name x phone x address x location x province x country x postalCode -> RegisteredUser
  + Creates a registered user with its respective values.
  + Precondition:
    - Every attribute must be an array of chars.
    - Postal code must correspond with address
    - Address, location, province and country must exist
  + Postcondition: a registered user is created.

*Destroyer*:

* **freeRegisteredUser**: RegisteredUser -> void
  + Frees the memory allocated for a RegisteredUser
  + Precondition:
    - Receives a non null RegisteredUser
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

ADT: ***Sale***

Description: Represents the purchase made by a registered user in a given time.

*Constructors*:

* **newSale:** code x discount -> Sale
  + Creates a Sale with its respective values.
  + Precondition:
    - Code must be a unique positive integer
    - Discount must be a positive number between 0 (included) and 100 (excluded)
  + Postcondition: a sale is created.

*Destroyer*:

* **freeSale**: Sale -> void
  + Frees the memory allocated for a Sale
  + Precondition:
    - Receives a non null Sale
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

*Modifiers*:

* **addProduct**: Sale x Product -> void
  + Adds a bought product to the current sale
  + Precondition: receives a non null Sale and Product
  + Postcondition: sale with added product
* **growSaleLineArray:** Sale -> void
  + Grows the array containing product that the sale has.
  + Precondition:
    - Receives a non null sale.
    - Array containing product fully filled.
  + Postcondition:
    - Max capacity of the array duplicated
    - Memory reallocated.
* **removeProduct**: Sale x productCode -> void
  + Precondition:
    - Receives a non null sale
    - productCode must correspond to an existing product.
  + Postcondition: sale with removed product

*Analyzers*:

* **endShopping**: Sale -> double
  + Sums the total to be payed and saves it in the total attribute. Defines the time t of the sale.
  + Precondition:
    - Receives a non null Sale
  + Postcondition: positive double or 0.

ADT: ***Sale Line***

Description: Represents a group of the same product.

*Constructors*:

* **newSaleLine:** product x quantity -> SaleLine
  + Creates a SaleLine with its respective values.
  + Precondition:
    - Product must be non null
    - Quantity must be a positive integer.
  + Postcondition: a saleLine is created.

*Destroyer*:

* **freeSaleLine**: SaleLine -> void
  + Frees the memory allocated for a SaleLine
  + Precondition:
    - Receives a non null SaleLine
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

EJERCICIO 3

ADT: ***Borrow***

Description: Represents the borrowing of Material from the Library to a Person.

*Constructors*:

* **newBorrow:** price x returnDays -> Borrow
  + Creates a Borrow structure with that will cost the person a certain price *p*, that has to be returned *r* returnDays after time of creation *t*. Has a borrowCode *b* that is unique.
  + Precondition:
    - Price must be a positive number
    - ReturnDays must be a positive integer.
  + Postcondition: a Borrow struct is created.

*Destroyer*:

* **freeBorrow**: Borrow -> void
  + Frees the memory allocated for a Borrow
  + Precondition:
    - Receives a non null Borrow
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

ADT: ***Library***

Description: Represents a Library that contains Material which can be borrowed to Persons.

*Constructors*:

* **newLibrary:** - -> Library
  + Creates a Library structure
  + Precondition: -
  + Postcondition: a Library is created.

*Destroyer*:

* **freeLibrary**: Library -> void
  + Frees the memory allocated for a Library
  + Precondition:
    - Receives a non null Library
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

*Modifiers*:

* **addMaterial:** Library x Material -> void
  + Adds material to the array of material contained by the library
  + Precondition: recieves non null Library and Material
  + Postcondition: Library with added Material.
* **addPerson:** Library x Person -> void
  + Adds a Person to the array of persons contained by the library
  + Precondition: recieves non null Library and Person
  + Postcondition: Library with added Person.
* **addBorrow:** Library x Borrow -> void
  + Adds a Borrow to the array of borrows contained by the library
  + Precondition: recieves non null Library and Borrow
  + Postcondition: Library with added Borrow.
* **generateBorrowCode**: Library -> int
  + Generates a unique borrowCode with the aid of an internal attribute contained in Library.
  + Precondition: receives a non null Library
  + Postcondition: positive integer.
* **removeMaterial:** Library x materialCode -> void
  + Removes material from the library
  + Precondition: materialCode has to correspond to an existing material in the library
  + Postcondition: library with removed material
* **removePerson:** Library x personCode -> void
  + Removes person from the library
  + Precondition: personCode has to correspond to an existing person in the library
  + Postcondition: library with removed person
* **removeBorrow:** Library x idBorrow -> void
  + Removes Borrow from the library
  + Precondition: idBorrow has to correspond to an existing borrow in the library
  + Postcondition: library with removed borrow

ADT: ***Material***

Description: Represents Material contained by a Library that can be a Book or a Magazine.

*Constructors*:

* **newBook**: code x author x title x year x editorial -> Material
  + Creates a Book with its corresponding attributes. MaterialType is 1.
  + Precondition:
    - Author, title and editorial must be char arrays
    - Year must be a positive integer
    - Code must be a unique positive integer
  + Postcondition: a Book is created.
* **newMagazine**: code x title x year x editorial -> Material
  + Creates a Magazine with its corresponding attributes. MaterialType is 2.
  + Precondition:
    - Title and editorial must be char arrays
    - Year must be a positive integer
    - Code must be a unique positive integer
  + Postcondition: a Magazine is created.

*Destroyer*:

* **freeMaterial**: Material -> void
  + Frees the memory allocated for a Material
  + Precondition:
    - Receives a non null Material
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

*Modifiers*:

* **enlistMaterial**: Material -> void
  + Changes a Material status to available
  + Precondition:
    - Receives a non null Material
  + Postcondition: available Material
* **takeOutMaterial**: Material -> void
  + Changes a Material status to not available
  + Precondition:
    - Receives a non null Material
  + Postcondition: not available Material

ADT: ***Person***

Description: Represents a Person that can be a Student or a Teacher

*Constructors*:

* **newStudent:** name x mail x phone x code x enrollment -> Person
  + Creates a Student with its corresponding attributes. PersonType is 1.
  + Precondition:
    - Name and mail must be char arrays
    - Phone code and enrollment must be a positive integer
    - Code must be unique to each Person.
  + Postcondition: a Student is created.
* **newTeacher:** name x mail x phone x code x employeeNumber -> Person
  + Creates a Teacher with its corresponding attributes. PersonType is 2.
  + Precondition:
    - Name and mail must be char arrays
    - Phone code and employee number must be a positive integer
    - Code must be unique to each Person.
  + Postcondition: a Teacher is created.

*Destroyer*:

* **freePerson**: Person -> void
  + Frees the memory allocated for a Person
  + Precondition:
    - Receives a non null Person
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

*Modifiers*:

* **takeMaterial**: Material x Library x Borrow x Person -> void
  + Person *p* that takes Material *m* from Library *l* creating a Borrow *b*.
  + Precondition: Receives a non null Material, Library, Borrow and Person
  + Postcondition: Person with added Material. Library with less Material.
* **leaveMaterial**: Material x Library x Borrow x Person -> void
  + Person *p* that retrieves Material *m* to the Library *l*, marked by a Borrow *b.*
  + Precondition: receives a non null Material, Library, Borrow and Person
  + Postcondition: Person with less Material. Library with added Material.

EJERCICIO 4

ADT: ***Client***

Description: Represents a Client of the Hotel.

*Constructors*:

* **newClient:** name x dni -> Client
  + Creates a Client with its corresponding attributes.
  + Precondition:
    - Name must be char array
    - DNI must be a positive integer unique to each client.
  + Postcondition: a Client is created.

*Destroyer*:

* **freeClient**: Client -> void
  + Frees the memory allocated for a Client
  + Precondition:
    - Receives a non null Client
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

ADT: ***Room***

Description: Represents a Room of a Hotel.

*Constructors*:

* **newRoom:** roomNumber x pricePerDay x roomType -> Client
  + Creates a Client with its corresponding attributes.
  + Precondition:
    - Name must be char array
    - DNI must be a positive integer unique to each client.
  + Postcondition: a Client is created.

*Destroyer*:

* **freeRoom**: Room -> void
  + Frees the memory allocated for a Room
  + Precondition:
    - Receives a non null Room
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

ADT: ***Reservation***

Description: Represents the reservation of a Room in a Hotel.

*Constructors*:

* **newReservation:** Client x roomNumber x daysToStay -> Reservation
  + Creates a Reservation with its corresponding attributes.
  + Precondition:
    - Days to stay must be a positive integer
    - RoomNumber must correspond to an existing, available room
    - Client must be non null
  + Postcondition: a Reservation is created.

*Destroyer*:

* **freeReservation**: Reservation -> void
  + Frees the memory allocated for a Reservation
  + Precondition:
    - Receives a non null Reservation
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

ADT: ***Receptionist***

Description: Represents the Receptionist of a Hotel that is in charge of reservations.

*Constructors*:

* **newReceptionist:** name x dni -> Receptionist
  + Creates a Receptionist with its corresponding attributes.
  + Precondition:
    - Name must be an array of chars
    - Dni must be a positive integer unique to each Receptionist
  + Postcondition: a Receptionist is created.

*Destroyer*:

* **freeReceptionist**: Receptionist -> void
  + Frees the memory allocated for a Receptionist
  + Precondition:
    - Receives a non null Receptionist
    - Language support to dynamically manage memory
  + Postcondition: memory freed.
* **deleteReservation:** clientDNI x Receptionist -> void
  + Receptionist deletes reservation of a client with a certain DNI
  + Precondition:
    - Receives non null Receptionist
    - Client DNI must correspond to an existing client with a previously made reservation
  + Postcondition: reservation and memory allocated for it deleted.

*Modifiers*:

* **checkIn:** Client x Receptionist x Hotel -> int
  + Client checks in at the hotel through the receptionist.
  + Precondition:
    - Receives non null Client, Receptionist and Hotel
  + Postcondition: 0 if the Client did not have a reservation and therefore can’t check in. 1 if the check in was successful.
* **makeReservation:** Client x roomNumber x Receptionist x daysToStay-> void
  + Client makes a reservation of a certain room, through the Receptionist, for a desired amount of days.
  + Precondition:
    - Receives non null Client and Receptionist
    - Room number must correspond to a non-booked room
    - DaysToStay must be a positive integer
  + Postcondition: reservation made

ADT: ***Invoice***

Description: Represents the Invoice of a Client’s stay in a Hotel.

*Constructors*:

* **newInvoice:** invoiceNumber x nitHotel x hotelName x clientName x clientDNI x price -> Invoice
  + Creates an Invoice with its corresponding attributes.
  + Precondition:
    - Client name and hotelName must be an array of chars
    - clientDni must be a positive integer that corresponds to a client that has stayed in the Hotel
    - Unite NIT specific to each country (*Argentina: CUIT*)
    - Invoice number unique
    - Price positive number
  + Postcondition: a Receptionist is created.

*Destroyer*:

* **freeInvoice**: Invoice -> void
  + Frees the memory allocated for an Invoice
  + Precondition:
    - Receives a non null Invoice
    - Language support to dynamically manage memory
  + Postcondition: memory freed

ADT: ***Hotel***

Description: Represents a Hotel where client can stay.

*Constructors*:

* **newHotel:** name x nitHotel x roomsMaxCapacity -> Hotel
  + Creates a Hotel with its corresponding attributes.
  + Precondition:
    - Name must be an array of chars
    - Unique NIT, specific to each country (*Argentina: CUIT*)
    - RoomsMaxCapacity positive integer
  + Postcondition: a Hotel is created.

*Destroyer*:

* **freeHotel**: Hotel -> void
  + Frees the memory allocated for a Hotel.
  + Precondition:
    - Receives a non null Hotel
    - Language support to dynamically manage memory
  + Postcondition: memory freed

*Modifiers*:

* **addRoom**: Hotel x Room -> void
  + Adds a room the the Hotel.
  + Precondition:
    - Receives a non null Hotel and Room
  + Postcondition: hotel with added room
* **addInvoice**: Hotel x Invoice -> void
  + Adds an Invoice the the Hotel.
  + Precondition:
    - Receives a non null Hotel and Invoice
  + Postcondition: hotel with added invoice
* **payRoom:** Client x Receptionist x Hotel -> Invoice
  + Pays for a previously reserved room, generating an Invoice (if the operation was successful)
  + Precondition:
    - Receives a non null Client, Receptionist, and Hotel
  + Postcondition:
    - Invoice if the operation was done correctly, NULL if it wasn’t.

*Analyzers*:

* **getRoom**: Hotel x roomNumber -> Room
  + Searches for the room associated to a roomNumber
  + Precondition:
    - Receives a non null Hotel
    - roomNumber must correspond to an existing Room in the Hotel.
  + Postcondition: room
* **getInvoiceCode**: Hotel -> int
  + Creates a unique code for an invoice
  + Precondition:
    - Receives a non null Hotel
  + Postcondition: positive unique integer

EJERCICIO 5

ADT: ***Admin***

Description: Represents an Admin of the system.

*Constructors*:

* **newAdmin:** name x dni -> Admin
  + Creates a Admin with its corresponding attributes.
  + Precondition:
    - Name must be char array
    - DNI must be a positive integer unique to each client.
  + Postcondition: an Admin is created.

*Destroyer*:

* **freeAdmin**: Admin -> void
  + Frees the memory allocated for an Admin
  + Precondition:
    - Receives a non null Admin
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

ADT: ***Client***

Description: Represents a Client of the Hotel.

*Constructors*:

* **newClient:** name x numberID -> Client
  + Creates a Client with its corresponding attributes.
  + Precondition:
    - Name must be char array
    - NumberID must be a positive integer unique to each client.
  + Postcondition: a Client is created.

*Destroyer*:

* **freeClient**: Client -> void
  + Frees the memory allocated for a Client
  + Precondition:
    - Receives a non null Client
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

ADT: ***Movie***

Description: Represents a Movie that can be rented.

*Constructors*:

* **newMovie:** name -> Movie
  + Creates a Movie with a name, and a unique numberID.
  + Precondition:
    - Name must be char array
  + Postcondition: a Client is created.

*Destroyer*:

* **freeMovie**: Movie -> void
  + Frees the memory allocated for a Movie
  + Precondition:
    - Receives a non null Movie
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

*Modifier:*

* **rentMovie**: Movie x Id x rentalDays -> void
  + Rent a movie for a certain amount of rentalDays
  + Precondition:
    - Receives a non null Movie and Id
    - RentalDays must be a positive integer
  + Postcondition: movie rented and no longer available

ADT: ***Excess***

Description: Represents the excess of time a client took to return a Movie.

*Constructors*:

* **newExcess:** costOfMovieRentPerDay -> Excess
  + Creates an Excess, with the cost of renting the corresponding movie per day.
  + Precondition:
    - costOfMovieRentPerDay must be a positive number
  + Postcondition: an Excess is created

*Destroyer*:

* **freeExcess**: Excess -> void
  + Frees the memory allocated for an Excess
  + Precondition:
    - Receives a non null Excess
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

*Analyzer*:

* **moviesWithoutReturn:** DataBase -> int
  + Gets, through the dataBase, the amount of movies that have yet not been returned
  + Precondition: dataBase must be non null.
  + Postcondition: positive integer

*Modifier:*

* **leaveMovie:** Movie x Excess -> void
  + Leave a movie with a certain excess.
  + Precondition: Movie and Excess must be non null
  + Postcondition: returned movie available for rent

ADT: ***Id***

Description: Id that has a unique number

*Constructors*:

* **newID:** numberId -> Id
  + Creates an Id
  + Precondition: -
  + Postcondition: an ID is created

*Destroyer*:

* **freeID**: ID -> void
  + Frees the memory allocated for an ID
  + Precondition:
    - Receives a non null ID
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

ADT: ***Database***

Description: Database that manages clients and movies.

*Constructors*:

* **newDataBase:** - -> DataBase
  + Creates a DataBase.
  + Precondition: -
  + Postcondition: a DataBase is created

*Destroyer*:

* **freeDataBase**: DataBase -> void
  + Frees the memory allocated for a DataBase
  + Precondition:
    - Receives a non null DataBase
    - Language support to dynamically manage memory
  + Postcondition: memory freed.

*Modifiers*:

* **addMovie:** Movie x Database -> void
  + Adds material to the array of movies contained by the database
  + Precondition: receives non null Movie and DataBase
  + Postcondition: Database with added Movie.
* **addClient:** Client x Database -> void
  + Adds a Client to the array of clients contained by the database.
  + Precondition: receives non null Client and Database
  + Postcondition: Database with added Client.
* **growMovieArray:** Database -> void
  + Grows the array containing movies that the database contains.
  + Precondition:
    - Receives a non null Database.
    - Array containing product fully filled.
  + Postcondition:
    - Max capacity of the array duplicated
    - Memory reallocated.
* **growClientArray:** Database -> void
  + Grows the array containing clients that the database contains.
  + Precondition:
    - Receives a non null Database.
    - Array containing product fully filled.
  + Postcondition:
    - Max capacity of the array duplicated
    - Memory reallocated.

*Analyzer*:

* **getRentMovieClient:** Id x Database -> Movie
  + Finds the movies a client rented in the dataBase.
  + Precondition:
    - Receives a non null Database and Id
  + Postcondition:
    - Array containing Movies, or empty if the client hasn’t rented any.
* **getMoviesAvailable:** Database -> Movie
  + Finds every movie that is available to be rented in the Database
  + Precondition:
    - Receives a non null Database
  + Postcondition:
    - Array containing Movies, or empty if there are not available movies to rent.
* **getMovie:** movieName x DataBase -> Movie
  + Finds a movie through its title, in the Database.
  + Precondition:
    - Receives a non null Database
    - movieName is an array of chars
    - movieName must correspond to an existing movie.
  + Postcondition: Movie
* **getIdCode:** DataBase -> int
  + Creates unique codes for IDs.
  + Precondition:
    - Receives a non null Database
  + Postcondition: positive unique integer.