Segunda entrega proyecto de curso "Glimmer Manga&Cafe"

Link github: https://github.com/AlejandroFonseca25/glimmer-manga-cafe

1. **Requerimientos funcionales:**

    1. Registrar un cliente al sistema. Para ello debe ingresarse el nombre completo, número y tipo de identificación, fecha de nacimiento, género, número telefónico, correo electrónico y contraseña. No pueden repetirse ni la ID, ni el número telefónico ni el correo electrónico entre clientes. Podrá ser realizado tanto por clientes como por empleados (Completado).

    2. Registrar un empleado al sistema. Para ello debe ingresarse el nombre completo, número y tipo de identificación, fecha de nacimiento, género, número telefónico, correo electrónico, contraseña y cargo que desempeña. No pueden repetirse ni la ID, ni el número telefónico ni el correo electrónico entre empleados. Función solamente disponible para el administrador.

    3. Permitir acceso a clientes y empleados previamente registrados. Para ello necesita su número de identificación y contraseña. Solo se permite el acceso si los dos anteriores son atributos de la misma persona.

    4. Permitir acceso a un administrador al sistema. Para ello la ID y la contraseña deben ser iguales a Strings predeterminados en el sistema. El ingreso de estos campos se hace en el login de empleados. el ID es "admin11037" y la contraseña "accessV3" (Completado).

    5. Reservar una habitación. Para ello debe seleccionar la cantidad de tiempo que desea (3, 6, 12 y 24 horas) y la habitación que desea (A-J). El costo de 3 horas es $30.000, el de 6 horas es $45.000, el de 12 horas es $70.000, y el de 24 horas $110.000. Solo puede reservar la habitación si está disponible y libre previamente. El costo será agregado a la deuda del cliente. Si la deuda sobrepasa los $300.000, no podrá realizar la reserva. Función solamente disponible para clientes.

    6. Extender el tiempo de reserva de una habitación por un máximo de 3 horas. Para ello debe seleccionar la cantidad de tiempo extra que desea. Esta acción solo puede hacerse una vez por habitación reservada. El costo de cada hora es de $10.000, y será agregado a la deuda del cliente.

Si la deuda sobrepasa los $300.000, no podrá realizar la reserva. Función solamente disponible para clientes.

7. Reservar mangas. Para ello puede buscar el manga por su nombre e ID, o puede seleccionarlo directamente de una lista. Una persona puede reservar un máximo de 5 mangas al mismo tiempo. Una vez reservado el manga, su estado cambia a "no disponible" y no puede ser reservado por otra persona hasta que sea retornado. Puede retornarlo en el momento que desee. Función solamente disponible para clientes.

8. Comprar comida. Para ello puede buscar el manga por su nombre o identificación, o puede seleccionarlo directamente de una lista. Después, puede seleccionar la cantidad deseada de cada comida. El costo será agregado a la deuda del cliente. Si la deuda sobrepasa los $300.000, no podrá registrar la compra. Función solamente disponible para clientes.

9. Actualizar la información del cliente. Para ello primero se busca el cliente por identificación o por nombre, y luego se editan los campos que se deseen actualizar. Los empleados diferentes al administrador pueden actualizar toda la información perteneciente a cliente (nombre y apellido, número y tipo de identificación, fecha de nacimiento, género, número telefónico, correo electrónico y contraseña). Además, se puede actualizar el estado del cliente, entre activo, inactivo, y vetado. Un usuario vetado no puede ingresar a la aplicación. Función solamente disponible para empleados.

10. Añadir mangas y comida al sistema. Para ello primero se selecciona cual de los dos va a ser añadido, y después se llenan los campos correspondientes. Un manga tiene nombre, fecha de publicación y estantería en la que se encuentra. Una comida tiene nombre, marca, cantidad y precio. Un manga no puede tener el mismo nombre que otro, lo mismo aplica para la comida. Los mangas se guardaran en un árbol binario, y la comida en dos listas circulares doblemente enlazadas: Comida y bebida. Función solamente disponible para empleados.

11. Ver la información de productos y clientes. Para ello se escoge que lista desea desplegar (Mangas, comidas o clientes). La lista mostrará el ID generado por el sistema y el nombre del producto o cliente. Al hacer click en un elemento en especifico, se mostrara toda la información del elemento. Función solamente disponible para empleados.

12. Remover mangas y comida del sistema. Para ello primero se busca el producto por el número de identificación generado por el sistema. Si el número coincide con un manga o una comida, mostrará el nombre y el tipo de producto (manga o comida) en pantalla. Para eliminarlo, se utiliza un botón. Si no encuentra un producto, lanzará un error. Función solamente disponible para empleados.

13. Registrar pagos. Para ello se busca el usuario al cual se le recibirá el pago por medio de su número de identificación. Luego, se introduce la suma que será pagada. Si la suma a pagar supera la deuda, quedará como balance positivo para futuros servicios. Función solamente disponible para empleados.

14. Deshabilitar/Habilitar habitaciones. Para ello se selecciona la habitación a deshabilitar con la interfaz y se deshabilita con un interruptor. Esta función sólo se puede realizar si la habitación está libre (No tiene a un cliente utilizandola). Función solamente disponible para el administrador.

15. Actualizar la información de un empleado. Para ello se selecciona el empleado por medio de una lista o por una búsqueda de su nombre o identificación de empleado. Luego se editan los campos que se deseen actualizar (Nombre, apellido, identificación de empleado, tipo de identificación, número telefónico, correo electrónico y cargo), finalmente se da click en el botón “save changes”. No pueden coincidir dos identificaciones de empleado, número telefónico o correo. Función solamente disponible para el administrador.

16. Cerrar la sesión. Para ello se presiona un boton que esta siempre presente en las interfaces, y que enviará al usuario a la ventana de login. Podrá ser realizado tanto por clientes como por empleados.

17. Mostrar la fecha y hora. La hora local está situada en cada interfaz a excepción del login. Es manejada por un hilo aparte.

18. Completar las búsquedas que el usuario realice. Muestra sugerencias para completar una búsqueda en cualquier campo, de acuerdo a los caracteres ingresados previamente. Es manejada por un hilo aparte.

19. Reproducir música personalizada. El usuario puede ingresar un url de YouTube para reproducir su audio. Este perdura durante toda la ejecución del programa. Es manejada por un hilo aparte.

2. Requerimientos no funcionales:

1. Los clientes serán guardados en un ArrayList.
2. Los empleados serán guardados en un árbol binario.
3. Las habitaciones están guardadas en un Array de tamaño 10.
4. Las máquinas están guardadas en un Array de tamaño 2.
5. Los mangas serán guardados en un árbol binario.
6. Las bebidas serán guardadas en una lista circular doblemente enlazada.
7. Las comidas serán guardadas en una lista circular doblemente enlazada.
8. Los dulces serán guardadas en una lista circular doblemente enlazada.
9. La información de cada persona es consistente y siempre debe ser desplegada en tiempo real cuando ingresen al sistema.
10. El programa debe ser persistente por serializado. La clase manager será serializada. Las comidas presentaron persistencia por medio de archivos de texto plano.
11. Los empleados y mangas podrán ser ordenados por burbuja, inserción y selección. También, empleados y clientes podrán ser ordenados por las interfaces Comparator y Comparable.
12. Las comidas y clientes tienen la capacidad de ejecutar búsquedas binarias.
13. Los requerimientos funcionales “Registrar empleado al sistema”, “Deshabilitar/habilitar habitación” y “Actualizar información de un empleado” sólo pueden ser ejecutados en la sesión del administrador.
14. Los requerimientos funcionales “Reservar una habitación”, “comprar comida”, “Reservar mangas” y “extender el tiempo de una habitación” sólo pueden ser ejecutados en la sesión de un cliente.

3. Diagramas de clase:

Al final del documento se encuentran todos los diagramas.

4. Diseño de las pruebas unitarias:

Scenarios Setup:

Name	Class	Scenario
setup1	ManagerTest/AdminTest	A manager, with 3 registered users: 1. n = “Alejandro” s = “Fonseca Forero” password = “2450” birthdate = “5/4/2000” dn = “1006227418” dt = “CC” phone = “3187653153” gender = “Male” 2. n = “Xian” s = “Lin Yao” password = “6547” dn = “365238” dt = “Passport” birthdate = “5/5/2001” phone = “4558769319” email = “XiYao@hotmail.com” gender = “Female” 3.n = “Soila” s = “Perez Ossa” password = “51247” dt = “CR” dn = “ACE105930” phone = ““ email = “soila@hotmail.com” gender = “Female”
setup2	AdminTest	A manager, with 2 registered Employees: 1. n = “Vordt” s = “of the Boreal Valley” password = “7453” employeeID = “1234” dn = “154879675” dt = “CC” phone = “314856323” password = “12345” birthdate = “10/4/2000” gender = “Male” charge = “Security Guard”

		2. n = “Lothric” s = “of Lordran” password = “5486” employeeID = “2314” dn = “564156451” dt = “CC” phone = “314586513” password = “12345” birthdate = “1/2/2000” gender = “Male” charge = “Manager”

Test objective:

To verify the correct functionality of the addClient method in manager, checking if the client was added to the properly.

Class	Method	Scenario	Input values	Result
Manager	testAddClient		n = “Albert” s = “Mockup” dn = “123547886221” dt = “CC” t = “3187653153” email = “asdf@gmail.com”	true
Manager	testAddClient		n = “” s = “Mockup” dn = “123547886221” dt = “” t = “3187653153” email = “asdf@gmail.com”	false

Test objective:
To verify the correct functionality of the confirmMangas method, checking if the mangas were added to the instance of the object of client that requested them.

Class	Method	Scenario	Input values	Result

ClientGUI	testConfirmMangas	setup1	mangas = man1, man2, man3, man4, man5	true

Test objective:
To verify the correct functionality of the confirmFoods method, checking if the food was added to the instance of the object of client that requested them.

Class	Method	Scenario	Input values	Result
ClientGUI	testConfirmFoods	setup1	foods = f1, f2, f3	true

Test objective:
To verify the correct functionality of the extendPlan method, checking if the plan was extended successfully

Class	Method	Scenario	Input values	Result
ClientGUI	testExtendPlan	setup1	planChoice	true

Test objective:
Verify if the addClient method of employee and admin (both being pretty much the same) works properly, checking if the user was added properly to the list

Class	Method	Scenario	Input values	Result
EmployeeGUI/AdminGUI	testAddClient	setup1	n = “Albert” s = “Mockup” dn = “123547886221” dt = “CC” t = “3187653153” email = “asdf@gmail.com”	true
EmployeeGUI/AdminGUI	testAddClient	setup1	n = “” s = “Mockup” dn = “123547886221” dt = “” t = “3187653153” email = “asdf@gmail.com”	false

Test objective:
Verify if the updateClientEmployee method of employee works properly, checking if the user was updated properly in the list

Class	Method	Scenario	Input values	Result
EmployeeGUI	testUpdateClientEmployee	setup1	n = “Wesker” s = “Albert” dn = “123547886221” dt = “CC” t = “3187653153” email = “Wesalbrt@gmail.com”	true

Test objective:
Verify if the addProduct method of employee and admin works properly, checking if the product was added properly in the list

Class	Method	Scenario	Input values	Result
EmployeeGUI/ AdminGUI	testAddProduct		productName = “Cheeto’s” shelf = “A”	true
EmployeeGUI/ AdminGUI	testAddProduct		productName = “Cheerio’s” shelf = “”	false

Test objective:
Verify if the removeProduct method of employee and admin works properly, checking if the product was removed properly of the list

Class	Method	Scenario	Input values	Result
EmployeeGUI/ AdminGUI	testRemoveProduct	testAddProduct	productName = “Cheeto’s”	true
EmployeeGUI/ AdminGUI	testRemoveProduct	testAddProduct	productName = “Cheerio’s”	false

Test objective:

Verify if the searchProductID method of employee and admin works properly, checking if the product was found properly

Class	Method	Scenario	Input values	Result
EmployeeGUI/AdminGUI	testSearchProductID	testAddProduct	productID= “1”	true
EmployeeGUI/AdminGUI	testSearchProductID	testAddProduct	productName = “23”	false

Test objective:
Verify if the pay method of employee and admin works properly, checking if the payment was subtracted from the instance of client correctly

Class	Method	Scenario	Input values	Result
EmployeeGUI/AdminGUI	testPay	setup1	clientID = “1006227418” payAmount = 12000	true
EmployeeGUI/AdminGUI	testPay	setup1	clientID = “54841” payAmount = 12000	false

Test objective:
Verify if the add method of admin for employees works properly, checking if the employee was added properly to the list

Class	Method	Scenario	Input values	Result
AdminGUI	testAddEmployee	setup2	n = “Albert” s = “Wesker” password = “12345” employeeID = “2153” dn = “32564537” dt = “CC” phone = “314545613” password = “32516safd”	true

			birthdate = “15/7/2000” gender = “Male” charge = “Janitor”	
AdminGUI	testAddEmployee	setup2	n = “Albert” s = “” password = “12345” employeeID = “” dn = “32564537” dt = “” phone = “314545613” password = “32516safd” birthdate = “” gender = “Male” charge = “”	false

Test objective:
Test the search method of admin with employees

Class	Method	Scenario	Input values	Result
AdminGUI	testSearchEmployee	setup2	employeeID = “1234”	true
AdminGUI	testSearchEmployee	setup2	employeeID = “123456”	false

Test objective:
Test the update client method of admin, checking that the data is correctly overwritten

Class	Method	Scenario	Input values	Result
AdminGUI	testUpdateClientAdmin	setup1	clientID = “365238” clientStatus = “Banned”	true
AdminGUI	testUpdateClientAdmin	setup1	clientID = “0000” clientStatus = “Inactive”	false

Test objective:

Test the updateClientAdmin method of admin, checking that the data is correctly overwritten

Class	Method	Scenario	Input values	Result
AdminGUI	testUpdateClientAdmin	setup1	clientID = “365238” clientStatus = “Banned”	true
AdminGUI	testUpdateClientAdmin	setup1	clientID = “0000” clientStatus = “Inactive”	false

Test objective:
Test the updateRoomAdmin method of admin, checking that the data is correctly overwritten

Class	Method	Scenario	Input values	Result
AdminGUI	testUpdateRoomAdmin		roomID = “1” roomStatus = “disabled”	true
AdminGUI	testUpdateRoomAdmin		roomID = “25” roomStatus = “enabled”	false

Test objective:
Test the updateRoomAdmin method of admin, checking that the data is correctly overwritten

Class	Method	Scenario	Input values	Result
AdminGUI	testUpdateEmployee	setup2	employeeID= “2153” n = “Weskah” s = “Aruberuto” password = “12345”	true

			employeeID = “2153” dn = “32564537” dt = “CC” phone = “314545613” password = “32516safd” birthdate = “15/7/2000” gender = “Male” charge = “Janitor”	
AdminGUI	testUpdateEmployee	setup2	employeeID= “21562”	false

tests
ManagerTest
+setup1():void
+setup2:void
+testAddClient():void
ClientGUITest
+setup1():void
+setup2():void
+testConfirmMangas():void
+testConfirmFoods():void
+testExtendedPlan():void
AdminGUITest
+setup1():void
+setup2():void
+testAddClient():void
+testUpdateClientEmployee():void
+testAddProduct():void
+testRemoveProduct():void
+testSearchProductID():void
+testPay():void
+testAddEmployee():void
+testSearchEmployee():void
+testUpdateClientAdmin():void
+testUpdateRoom():void
+testUpdateEmployee():void
-adminGUI
-manager
-clientGUI
model
Manager
+Manager()
+addClient(String firstName, String lastName, String ID, String IDType, LocalDate birthdate, String gender, String phone, String email, String password)
+getClients():ArrayList<Clients>
+getHeadEmployee():Employee
+getHeadManga():Manga
+getMachines():Machine[]
+getRooms():Room[]
+setClients(ArrayList<Client>):void
+setHeadEmployee(Employee):void
+setHeadManga(Manga):void
+setMachines(Machine[]):void
+setRooms(Room[]):void
-machines
-rooms
-m1
-clients:arrayList
-headEmployee
-headManga
-mangas
<<Abstract>>
Machine
-money : int
+Machine()
+getMoney():int
+setMoney(int money):void
FoodMachine
-firstFood : FoodType
+FoodMachine()
+getFirstFood():FoodType
+setFirstFood(FoodType firstFood):void
CandyMachine
-firstCandy : Candy
+CandyMachine()
getFirstCandy():Candy
setFirstCandy(Candy firstCandy):void
-firstFood
-firstCandy
<<Abstract>>
FoodType
-brand : String
-name : String
-price : int
-quantity : int
-systemID : String
+FoodType(String name, String brand, int price, int quantity)
+getBrand():String
+getName():String
+getNext():FoodType
+getPrev():FoodType
+getPrice():int
+getQuantity():int
+getSystemID():String
+setBrand(String):void
+setName(String):void
+setNext(FoodType):void
+setPrev(FoodType):void
+setPrice(int):void
+setQuantity(int):void
+setSystemID(String):void
-next
-prev
Drink
-carbonated : boolean
-milliliters : double
+Drink(String name, String brand, int price, int quantity, boolean carbonated, double milliliters)
+getMilliliters():double
+isCarbonated():boolean
+setCarbonated(boolean carbonated):void
+setMilliliters(double milliliters):void
Food
-gluten : boolean
-grams : double
+Food(String name, String brand, int price, int quantity, boolean gluten, double grams)
+getGrams():double
+isGluten():boolean
+setGluten(boolean gluten):void
+setGrams(double grams):void
Room
-enabled : boolean
-name : String
-occupied : boolean
-time : long
+Room(String name)
+getName():String
+getTime():long
+isEnabled():boolean
+isOccupied():boolean
+setEnabled(boolean):void
+setName(String):void
+setOccupied(boolean):void
+setTime(long):void
Candy
-brand : String
-name : String
-price : int
-quantity : int
-sugarQuantity : double
-systemID : String
+Candy(String name, String brand, int price, int quantity, double sugarQuantity)
+getBrand():String
+getName():String
+getNext():Candy
+getPrev():Candy
+getPrice():int
+getQuantity():int
+getSugarQuantity():double
+getSystemID():String
+setBrand(String brand):void
+setName(String name):void
+setNext(Candy next):void
+setPrev(Candy prev):void
+setPrice(int price):void
+setQuantity(int quantity):void
+setSugarQuantity(double sugarQuantity):void
+setSystemID(String systemID):void
-prev
-next
Manga
-name:String
-publishingDate:LocalDate
-genre:String
-bookshelf:char
-available:boolean
-SystemId:String
+Manga(name:String, pDate:LocalDate, genre: String, bookshelf:char, SystemId:String)
-left
-right
Employee
-birthdate : LocalDate
-charge : String
-email : String
-employeeID : String
-firstName : String
-gender : String
-iD : String
-IDType : String
-lastName : String
-password : String
-phone : String
-systemID : String
+Employee(String firstName, String lastName, String iD, String IDType, LocalDate birthdate, String gender, String charge, String phone, String email, String password)
+getBirthdate():LocalDate
+getCharge():String
+getEmail():String
+getEmployeeID():String
+getFirstName():String
+getGender():String
+getID():String
+getIDType():String
+getLastName():String
+getLeft():Employee
+getPassword():String
+getPhone():String
+getRight():Employee
+getSystemID():String
+setBirthdate(LocalDate birthdate):void
+setCharge(String charge):void
+setEmail(String email):void
+setEmployeeID(String employeeId):void
+setFirstName(String firstName):void
+setGender(String gender):void
+setID(String ID):void
+setIDType(String IDType):void
+setLastName(String lastName):void
+setLeft(Employee left):void
+setPassword(String password):void
+setPhone(String phone):void
+setRight(Employee right):void
+setSystemID(String systemID):void
Client
-balance : double
-birthdate : LocalDate
-email : String
-firstName : String
-gender : String
-iD : String
-IDType : String
-lastName : String
-password : String
-phone : String
-status : String
-systemID : String
Client(String firstName, String lastName, String iD, String IDType, LocalDate birthdate, String gender, String phone, String email, String password)
+getBalance():double
+getBirthdate():LocalDate
+getEmail():String
+getFirstName():String
+getGender():String
+getID():String
+getIDType():String
+getLastName():String
+getMangas():Manga[]
+getPassword():String
+getPhone():String
+getStatus():String
+getSystemID():String
+setBalance(double balance):void
+setBirthdate(LocalDate birthdate):void
+setEmail(String email):void
+setFirstName(String firstName):void
+setGender(String gender):void
+setID(String ID):void
+setIDType(String IDType):void
+setLastName(String lastName):void
+setMangas(Manga[] mangas):void
+setPassword(String password):void
+setPhone(String phone):void
+setStatus(String status):void
+setSystemID(String systemID):void
ui
Main
+main(args:String[]):void
+start(stage:Stage):void
MainPane.fmxl
MainGUI
-clientGUI : ClientGUI
-clientID : TextField
-clientPassword : PasswordField
-employeeLoginID : TextField
-employeeLoginPassword : PasswordField
-logoImageView : ImageView
-mainPane : BorderPane
+MainGUI(manager : Manager)
+getMainPane() : BorderPane
+loadEmployeeLogin(event : ActionEvent) : void
+loadLogin(event : ActionEvent) : void
+loadSignUp(event : ActionEvent) : void
+signInClient(event : ActionEvent) : void
+signInEmployee(event : ActionEvent) : void
-clientGUI
-mainGUI
-adminGUI
-employeeGUI
AdminGUI
-adminNameUPLEFT : Label
-amountToPay : TextField
-clientActiveRB : RadioButton
-clientAgeADD : DatePicker
-clientAgeUPDT : TextField
-clientBannedRB : RadioButton
-clientEmailADD : TextField
-clientEmailUPDT : TextField
clientFemaleRB : RadioButton
-clientFemaleRBADD : RadioButton
-clientFirstNameADD : TextField
-clientFirstNameUPDT : TextField
-clientGender : ToggleGroup
-clientGenderADD : ToggleGroup
-clientIDADD : TextField
-clientIdTypeADD : ChoiceBox<String>
-clientIDUPDT : TextField
-clientIdTypeADDfromADM : ChoiceBox<String>
-clientIDUPDT : TextField
-clientInactiveRB : RadioButton
-clientLastNameADD : TextField
-clientLastNameUPDT : TextField
-clientMaleRB : RadioButton
-clientMaleRBADD : RadioButton
-clientPasswordADD : PasswordField
-clientPasswordUPDT : TextField
-clientPhoneADD : TextField
-clientPhoneNumberUPDT : TextField
-clientPhoneUPDT : TextField
-clientStatus : ToggleGroup
-dateHourTextField : Label
-employeeAgeUPDT : TextField
-employeeChargeMAN : TextField
-employeeEmailMAN : TextField
-employeeFemaleRB : RadioButton
-employeeFirstNameUPDT : TextField
-employeeGUI : EmployeeGUI
-employeeIDMAN : TextField
-employeeIDTypeMAN : TextField
-employeeIDTypeUPDT : ChoiceBox<String>
-employeeIDUPDT : Label
-employeeLastNameUPDT : TextField
-employeeMaleRB : RadioButton
-employeeNameMAN : TextField
-employeePasswordUPDT : TextField
-employeePhoneMAN : TextField
-employeePhoneUPDT : TextField
-employeePositionUPDT : TextField
-employeeSystemIDMAN : TextField
-employeeTabView : TableView<String>
-employeeUserIDMAN : TextField
-gender : ToggleGroup
-genderEmployee : ToggleGroup
-iDType : ChoiceBox<String>
-productIDSearch : TextField
-productName : TextField
-productShelf : TextField
-productType : ChoiceBox<String>
-roomDisabled : RadioButton
-roomEnabled : RadioButton
-roomGroup : ToggleGroup
-roomIDTextField : TextField
-searchEmployeeTextField : TextField
-userIDPAY : TextField
+AdminGUI(mainGUI : MainGUI, manager : Manager, employee : EmployeeGUI) : void
+addClient(event : ActionEvent) : void
+addProduct(event : ActionEvent) : void
+loadAddClient(event : ActionEvent) : void
+loadAdminInterface(event : ActionEvent) : void
+loadEmployeeInterface(event : ActionEvent) : void
+loadEmployeeManagement(event : ActionEvent) : void
+loadLogin(event : ActionEvent) : void
+loadPayments(event : ActionEvent) : void
+loadRoomManagement(event : ActionEvent) : void
+loadSupplyManagement(event : ActionEvent) : void
+loadSystemInformation(event : ActionEvent) : void
+loadUpdateClientAdmin(event : ActionEvent) : void
+pay(event : ActionEvent) : void
+removeProduct(event : ActionEvent) : void
+saveChangesEmployee(event : ActionEvent) : void
+searchEmployee(event : ActionEvent) : void
+searchProductID(event : ActionEvent) : void
+updateClientAdmin(event : ActionEvent) : void
+updateEmployee(event : ActionEvent) : void
+updateRoomsAdmin(event : ActionEvent) : void
EmployeeGUI
-amountToPay : TextField
-clientAgeUPDT : TextField
-clientEmailADD : TextField
-clientEmailUPDT : TextField
-clientFemaleRB : RadioButton
-clientFemaleRBADD : RadioButton
-clientFirstNameADD : TextField
-clientFirstNameUPDT : TextField
-clientGender : ToggleGroup
-clientGenderADD : ToggleGroup
-clientIDADD : TextField
-clientIdTypeADD : ChoiceBox<String>
-clientIDUPDT : TextField
-clientLastNameADD : TextField
-clientLastNameUPDT : TextField
-clientMaleRB : RadioButton
-clientMaleRBADD : RadioButton
-clientPasswordADD : PasswordField
-clientPhoneADD : TextField
-clientPhoneUPDT : TextField
-dateHourTextField : Label
-employeeNameUPLEFT : Label
-gender : ToggleGroup
-iDType : ChoiceBox<String>
-productIDSearch : TextField
-productName : TextField
-productShelf : TextField
-productType : ChoiceBox<String>
-userIDPAY : TextField
+EmployeeGUI(main : MainGUI, manager : Manager, dminGUI : AdminGUI)
+addClient(event : ActionEvent) : void
+addProduct(event : ActionEvent) : void
+loadAddClient(event : ActionEvent) : void
+loadAdminInterface(event : ActionEvent) : void
+loadEmployeeInterface(event : ActionEvent) : void
+loadLogin(event : ActionEvent) : void
+loadPayments(ActionEvent) : void
+loadSupplyManagement(event : ActionEvent) : void
+loadSystemInformation(event : ActionEvent) : void
+loadUpdateClientEmployee(event : ActionEvent) : void
+pay(event : ActionEvent) : void
+removeProduct(event : ActionEvent) : void
+searchProductID(event : ActionEvent) : void
+setAdminGUI(event : AdminGUI) : void
+signUp(event : ActionEvent) : void
+updateClientEmployee(event : ActionEvent) : void
-adminGUI
-employeeGUI
ClientGUI
-availableFoodList : ListView<String>
-availableMangasList : ListView<String>
-chosenFoodList : ListView<String>
-chosenMangasList : ListView<String>
-clientEmailSelfADD : TextField
-clientNameUPLEFT : Label
-dateHourTextField : Label
-extensionPlans : ChoiceBox<String>
-roomA : Button
-roomB : Button
-roomC : Button
-roomD : Button
-roomE : Button
-roomF : Button
-roomG : Button
-roomH : Button
-roomI : Button
-roomJ : Button
-timeLeftClientTextField : Label
+ClientGUI(mainGUI : MainGUI, manager : Manager)
+addFood(event : ActionEvent) : void
+addMangas(event : ActionEvent) : void
+confirmFood(event : ActionEvent) : void
+confirmMangas(event : ActionEvent) : void
+extendPlan(event : ActionEvent) : void
+loadClientInterface(event : ActionEvent) : void
+loadExtendTime(event : ActionEvent) : void
+loadFoods(event : ActionEvent) : void
+loadLogin(event : ActionEvent) : void
+loadMangas(event : ActionEvent) : void
+loadRooms(event : ActionEvent) : void
+removeFood(event : ActionEvent) : void
+removeMangas(event : ActionEvent) : void
+rentRoom(event : ActionEvent) : void