Public coffee machine at a university

Requirements

Non-functional requirements:

- available 24/7
- flexible payment system (by cash or by card)
- precise adherence to the recipe

Functional requirements:

- allow user to get takeaway drink in a paper cup
- allow user to choose the drink
- allow user to set the amount of sugar
- notify user about the status of the drink preparation via sensor screen
- notify user about the payment status via sensor screen
- notify user about the amount of money deposited
- notify administrator of all actions with machine

Use cases

Use case 1
Title: Takeaway drink
Primary actor: Client
Success scenario: The client chooses a drink. The system displays the cost on the screen and offers to choose a payment method - by card or in cash. The client chooses the payment method, pays for the drink. The system accepts payment, displays on the screen that the payment was successful. The system prepares a drink, displays the preparation status on the screen. The client takes the finished drink. In the case of cash payment, if the client has deposited an amount exceeding the cost of the drink, the client receives the balance of funds.

Use case 2
Title: Generate reports
Primary actor: Administrator
Success scenario: The administrator connects to the coffee machine system, selects the report, receives the result on the screen of his computer.

Use case 3
Title: Change settings
Primary actor: Administrator
Success scenario: The administrator has the ability to remotely adjust the recipe of drinks in the coffee machine, payment options of the coffee machine.

Use case 4
Title: Service of coffee machine
Primary actor: Employee
Success scenario: The employee has the keys to the coffee machine. Once opened, the employee enters a password to notify the system and prevent unauthorized access to the machine. After that, the employee has the opportunity to replenish the stock of machine ingredients and consumables, test the operation of the coffee machine, withdraw cash from the bill acceptor, add bill tape. The system notifies the administrator of all employee actions.

Extensions:
1. If the coffee machine does not have a connection for card payment, the system should display an appropriate message to the client and notify the administrator of the problem.
2. If there is not enough cash in the coffee machine to dispense change or too many cash inside, the system should display an appropriate message for the client and notify the administrator of the problem
3. If there are not enough ingredients or consumables in the coffee machine to prepare a drink, the system should display an appropriate message for the client and notify the administrator of the problem
4. If the coffee machine is broken, the system should display an appropriate message for the client (if it is possible) and notify the administrator of the problem

Class Diagram

Account
-account: long
+getAccount(): long +setAccount(long account)

BillAcceptor
-billAcceptor: long
+getBillAcceptor(): long +setBillAcceptor(long billAcceptor)

Payment	
-payment: String	
+getPayment(): String +setPayment(String payment)	
Cash	**Card**
-payment: String = “Cash”	-payment: String = “Card”

User		
-id: int		
+getId(): int +setId(int id)		
Client	**Employee**	**Administrator**
-deposit: long	-name: String -password: String	-name: String -password: String
+getDeposit(): long +setDeposit(long deposit)	+getName(): String +setName(String name) +getPassword(): String +setPassword(String password)	+getName(): String +setName(String name) +getPassword(): String +setPassword(String password) +addUser(): User

Menu
-message: List <String>
+getMessage(): String +setMessage(String message) +addMessage(String message) +removeMessage():List <String>

Drink
-drink: String -recipe: Recipe -price: long
+getDrink(): String +setDrink(String drink) +getRecipe(): Recipe +setPecipe(Recipe recipe) +getPrice(): long +setPrice(long price)

Chocolate	Americano	Cappuccino	Espresso	Latte
-drink: String = “Chocolate"	-drink: String = “Americano"	-drink: String = “Cappuccino"	-drink: String = “Espresso"	-drink: String = “Latte"

Recipe
-ingredients: List <Ingredient>
+addIngredient(Ingredient ingredient): List <Ingredient> +removeIngredient(Ingredient ingredient): List <Ingredient>

Ingredient
-size: long -name: String
+getSize(): long +setSize(long size) +getName(): String +setName(String name)

Water	Coffee	Milk	Chocolate	Sugar
-name: String=”Water”	-name: String=”Coffee”	-name: String=”Milk”	-name: String=”Chocolate”	-name: String=”Sugar”

Consumable
-amount: int -name: String
+getAmount: int +setAmount(int amount) +getName(): String +setName(String name)

PaperCup	Spoon	BillTape
-name: String=”PaperCup”	-name: String=”Spoon”	-name: String=”BillTape”

Machine

-user: User
-drink: Drink
-consumables: List <Consumable>
-ingredients: List <Ingredient>
-paymentType: Payment
-billAcceptor: BillAcceptor
-account: Account
-menu: Menu

+chooseDrink(menu): Drink
+addSugar()
+subtractSugar()
+choosePaymentType(menu): Payment

+prepareDrink()
+giveChange(Deposit deposit): long

+increaseConsumableValue(Consumable consumable): List <Consumable>
+decreaseConsumableValue(Drink drink): List <consumable>
+increaseIngredientValue(Ingredient ingredient): List <Ingredient>
+decreaseIngredientValue(Drink drink): List <Ingredient>
+changeAccountValue(long value): Account
+changeBillAcceptor(long value): BillAcceptor

+checkPaymentMethod(): boolean
+checkIngredientValue(): boolean
+checkConsumableValue(): boolean
+checkConnectionForPayment(): boolean
+checkBillAcceptor(): boolean
+checkDepositMoney(): boolean
+checkEmployeeAccess(): boolean
+checkAdministratorAccess(): boolean

+printScreenMessage(): boolean
+sendReport(): boolean
+printBill(): boolean

UML class diagram with the relationship between classes