A1 Documentation

The class contract involves implementing classes for different types of customer contracts (Month-to-Month, Term, and Prepaid). Each class accurately manages and reflects the specific billing behaviors, rates, and conditions associated with its contract type, such as handling monthly fees, call charges, deposits, and any included benefits like free minutes.

MTMContract (Month-to-Month): Manages contracts without a fixed term, applying a standard monthly fee and per-minute charges for calls. This contract type offers flexibility, allowing customers to cancel any time without penalties.

TermContract: For customers committed to a fixed term, this contract includes a deposit, lower monthly fees, and possibly free minutes. The system must handle the deposit's return if the contract is completed or retained if terminated early.

PrepaidContract: Customers pay in advance for their usage, with the system tracking the balance. If the balance runs low, it must be topped up. This contract type requires careful management of the balance for billing and top-up purposes.

Each class must correctly implement methods for advancing to a new month (new_month), billing calls (bill_call), and contract cancellation (cancel_contract), reflecting the unique aspects of the contract type.

TermContract:

Overview: TermContract is a sublass of the Contract class and hence it inherits the methods from the parent class. It implements a term-based contract, including a term_deposit variable and a free_minutes variable. This class includes logic for handling a term deposit, offering free minutes, and managing monthly fees. It outlines how contracts transition between months, account for call durations against free minutes, and determine final charges upon cancellation, particularly how the term deposit is treated based on the cancellation date. This sublass has the following four methods:

1) The __init__ or initializer function.
2) The new_month method
3) The bill_call method
4) The cancel_contract method

The term contract class has the following attributes:

1) start: Starting date for the contract.
2) end_date: end date for the contract.
3) bill: Bill for this contract for the last month.
4) current_date: The date of the latest month that is added. 5) free_minutes: The number of free calling minutes.

These are the attribute types of the term contract class:

1) start: datetime.datetime
2) end_date: datetime.datetime
3) bill: Optional[bill]
4) current_date: tuple[int, int]
5) free_minutes: int
6) term_deposit: float

The TermContract class has the following representation invariants:

1) term_deposit >= 0
2) start > current_date > end_date
3) free_minutes > 0

Methods of TermContract class:

1) The Initializer Method:

The initializer methods of the subclass inherits the method from the parent class. The code uses Contract.__init__(self, start) to inherit the self.start attribute from the parent class Contract. The initial value of self.term deposit has been set equal to the TERM_DEPOSIT and the initial value of self.end_date is equal to end_date attribute.

2) The new_month Method:

Description:

A new month has begun corresponding to <month> and <year>. This may be the first month of the contract. Is stores the <bill> argument in this contract and set the appropriate rate per minute and fixed cost. This method takes in a month as an integer, year as an integer, and bill as a type of bill. The method does not return any value, it just modifies the self.bills attribute. As the new month

has started, the method sets <current_month> and <current_year> as given month and year. If this is the first month of the term_contract, the term_deposit will be added to the bill.

Implementation:
The new_month method of the TermContract class overrides the new_month method of the contact parent class. It records the current month and year, marking the progression of the contract timeline. The method receives a Bill object, which it assigns to the contract to handle this month's billing. This object will track call costs and fixed charges for the month. The contract provides a certain number of free minutes (TERM_MINS) each month, which are reset to ensure the customer can use them during the new billing period. If this month is the start of the contract, a term deposit (TERM_DEPOSIT) is added to the bill as a fixed cost. This deposit is a unique aspect of term contracts, representing a pre-paid amount that may be refundable under certain conditions. A fixed monthly fee (TERM_MONTHLY_FEE) is added to the bill, representing the basic charge for maintaining the contract.

3) The bill_call Method:

Description:

The bill_call method is used to bill a given call object. This method takes call as an input and does not return anything, it just adds the <call> to the bill. This method has a precondition that a bill has already been created for the month+year when the <call> was made. In other words, self.bill has been already advanced to the right month+year.

Implementation:

It converts the call duration from seconds to minutes, rounding up to the nearest whole minute, if necessary, to determine how many minutes to bill. If the customer has enough free minutes to cover the call, it deducts the call's minutes from the free minutes balance and records the minutes as used but not billed. If the call lasts longer than the available free minutes, it calculates the remaining minutes after using up the free minutes. These excess minutes are then added to the bill as chargeable.

In this method, a variable named call_mins is defined, which is assigned the value of ceil(call.duration / 60.0). If the number of free minutes is greater than call_mins, then call_mins is subtracted from free_minutes. Conversely, if the number of free minutes is less than call_mins, free_minutes is subtracted from call_mins. Subsequently, the method add_billed_minutes is utilized to add billed minutes by adding call_mins to bill.

4) The cancel_contract Method:

Description:

The cancel_contract method returns the amount the client owes in order to close the contract and phone line associated with it. If the client cancels after the <term date> they receive the entire term deposit amount. The method returns a float value. The method has a Precondition that a bill has already been created for the month+year when this contract is being cancelled. In other words, it can be safely assumed that self.bill exists for the right month+year when the cancellation is requested.

Implementation:

The cancel_contract method calculates the final amount due when a phone line contract is ended. It checks if the contract's cancellation occurs after its scheduled end date, comparing the current date (self.current_date) to the contract's end date (self.end_date). If the cancellation is indeed late, it applies a refund by subtracting the term deposit from the bill (self.bill.add_fixed_cost(-self.term_deposit)). This action adjusts the final bill to reflect the deposit return. Finally, the method returns the adjusted total bill (self.bill.get_cost()), which is the amount owed by the customer for all accumulated charges up to the point of cancellation.

MonthToMonth Contract Class:

Overview: MonthToMonth Contract is a subclass of the Contract class and hence it inherits the methods from the parent class. It represents a simpler contract without a term commitment or free minutes, charging a flat monthly fee and per-minute costs for calls. Since it inherits from the parent class, it has a bill attribute.

THE MTM contract class has the following methods:

1)The __init__ or initializer function.

2)The new_month method

The MTM contract class has the following attributes:

1) start: Starting date for the contract.
2) bill: Bill for this contract for the last month.

These are the attribute types of the MTM contract class:

1) start : datetime.datetime
2) bill: Optional[bill]

Methods of MTMContract class:

1)The Initializer Method:

The initializer methods of the subclass inherit the method from the parent class. Contract.__init__(self, start) is used to inherit the self.start attribute from the parent class, Contract.

2)The new_month Method:

Description:

A new month has begun corresponding to <month> and <year>. This may be the first month of the contract. It stores the <bill> argument in this contract and set the appropriate rate per minute and fixed cost. This method takes in a month as an integer, year as an integer, and bill as a type bill. The method does not return any value, it just modifies the self.bills attribute. As the new month has started, the method sets <current_month> and <current_year> as given month and year. The MTM contract has fewer unique elements and it does not have any return value.

Implementation:

The new_month method of the MTMContract class overrides the new_month method of the contact parent class. It associates the contract with a new Bill object (self.bill = bill), setting the call rate for the month (self.bill.set_rates("MTM", MTM_MINS_COST)) based on the Month-toMonth (MTM) contract's predefined cost per minute. Additionally, it adds a fixed monthly charge to the bill (self.bill.add_fixed_cost(MTM_MONTHLY_FEE)) to cover the service cost for the new month. This ensures the contract is accurately billed for usage and the monthly fee.

3)The bill_call Method:

Description:

The bill_call method is called from the parent class since the bill_call method is not required in the MTM Class. Through inheritance in the initializer from the parent class, the MTM contract class automatically inherits the bill_call method. This method does not return a value.

Implementation:

The MTM contract class does not use the bill_call method and it just inherits it from the parent class. This class directly bills the client based on call duration.

4)The cancel_contract Method:

Description:

The cancel_contract method is inherited from the parent class through the initializer as there is no need for cancel_contract in the MTM contract class.

Implementation:

There is no cancel_contract method for the MTM contract class as it does not need one. It just inherits it from the super class.

PrePaid Contract:

Overview: TermContract is a sublass of the Contract class and hence it inherits the methods from the parent class. It implements a pre-paid-basis contract. It manages a prepaid service, where the balance is adjusted for call costs and monthly charges. It includes logic for a monthly credit topup if the balance falls below a threshold and adjusts the bill accordingly.

This sublass has the following four methods:

1)The __init__ or initializer function.

2)The new_month method

3)The bill_call method

4)The cancel_contract method

The term contract class has the following attributes:

1)start: Starting date for the contract.

2)bill: Bill for this contract for the last month.

3)balance: The amount of credits that this contract has left.

These are the attribute types of the term contract class:

1) start: datetime.datetime
2) balance: float
3) bill: Optional[bill]

Methods of PrepaidContract class:

1)The Initializer Method:

The initializer methods of the subclass inherit the method from the parent class. Contract.__init__(self, start) is used to inherit the self.start attribute from the parent class, Contract. The value of self.balance is set to (-1 * balance).

2)The new_month Method:

Description:

A new month has begun corresponding to <month> and <year>. This may be the first month of the contract. Is stores the <bill> argument in this contract and set the appropriate rate per minute and fixed cost. This method takes in a month as an integer, year as an integer, and bill as a type bill. The method does not return any value, it just modifies the self.bills attribute. As the new month has started, the method sets <current_month> and <current_year> as given month and year.

Implementation:

The method starts by associating the contract with a new Bill object for the current month, effectively resetting the billing information to reflect new charges. It specifies the cost per minute for calls (PREPAID_MINS_COST), ensuring that the bill will calculate call charges based on the rates applicable to prepaid contracts. If the account's balance is greater than -$10, indicating the credit is running low, it deducts $25 from the balance, effectively topping up the account to ensure continued service. Finally, the current balance (which can be negative, indicating credit) is added as a fixed cost to the bill. This operation adjusts the bill to reflect the prepaid account's credit or debit status as the new month begins.

3)The bill_call Method:

Description:

The bill_call method bills a given call object. This method accepts call as input and does not return any value; instead, it adds the call to the bill. A precondition for this method is that a bill has already been created for the month and year in which the call was made. It is assumed that self.bill has already been advanced to the appropriate month and year.

Implementation:

The bill_call method processes a phone call for a prepaid contract by first determining the call's duration in minutes, using the ceiling function to round up as needed. The calculated duration is added to the bill as billed minutes to record the call's cost. The method concurrently updates the contract's balance by calculating the call's cost, which is done by multiplying the rounded duration by the contract's cost per minute. The bill is thus reflective of the call charges, and the prepaid balance is adjusted for the cost of the call. Within this method, a variable named var is set to the result of ceil(call.duration / 60.0). The add_billed_minutes method is then applied to var, and PREPAID_MINS_COST multiplied by var is added to self.balance.

4)The cancel_contract Method:

Description:

The cancel_contract method returns the amount the client owes in order to close the contract and phone line associated with it. If the client cancels after the <term date> they receive the entire term deposit amount. The method returns a float value. The method has a Precondition that a bill has already been created for the month+year when this contract is being cancelled. In other words, it can be safely assumed that self.bill exists for the right month+year when the cancellation is requested.

Implementation:

The cancel_contract method computes the final amount due upon termination of a phone line contract. The method evaluates self.balance, and if it is greater than 0, the method returns self.balance, which represents the final amount owed by the client. If the balance is zero or negative, indicating that prepaid credit is still available, the owed amount is none, and the method returns zero, indicating that the customer has no outstanding debt.