ASSIGNMENT 1 DOCUMENTATION

# **PUBLIC INTERFACE**

A Valid Call is a well-formed Call object that has:

* Valid src\_number (str)
* Valid dst\_number (str)
* Valid time (datetime.datetime)
* Valid duration (>=0) (int)
* Valid src\_loc (tuple[float, float])
* Valid dst\_loc (tuple[float, float])

# TermContract class

(Inherited from Contract class)

## Description:

A term-based phone contract with an initial deposit and included monthly free minutes.

* If the customer has not explicitly cancelled the contract, then the same contract will renew and deposit will carry on to next month.
* If the customer cancels before end of term, then the deposit is forfeited.
* If the customer finished the term, then the deposit is refunded and subtract from the bill.

## Public Methods

**\_\_init\_\_(start: datetime.datetime, end: datetime.datetime) -> None**

Parameters

* start: term start date
* end: term end date

Usage

* The initializer for this class. Must pass in start as a datetime object and end as a datetime object to specify the start and end date of this contract

**new\_month(month: int, year: int, bill: Bill) -> None**

Parameters

* month: the month to advance to
* year: the year to advance to
* bill: the Bill of that month

Usage

* This method advances the term contract to a new month and year specified in the argument, renewing the monthly free minutes and bills the monthly fee
* Only bill the deposit if advancing to the start date (month, year) of the contract

Preconditions

* The bill must exist for that month and year

**bill\_call(call: Call) -> None**

Parameters

* call: the Call to bill from

Usage

* Deducts the call duration from free minutes first, and adds the free minutes from the bill
* Bills extra minutes at contract’s minimum cost if free minutes are used up

Preconditions

* call must be a valid Call (see definition at top)

**cancel\_contract() -> float**

Usage

* If cancelling after the term has ended, deposit will be subtracted from the cost
* If cancelled before the term has ended, deposit will be forfeited
* This method will return the total the customer owes to the company for cancelling the contract

# MTMContract class

(Inherited from Contract class)

## Description:

A month-to-month phone contract with a fixed monthly cost and no free minutes

## Public Methods

**\_\_init\_\_(start: datetime.datetime) -> None**

Parameter

* start: the start date of the monthly contract

Usage

* The initializer for this class. Must pass in start date of the contract as a datetime object to specify the start of the monthly contract

**new\_month(month: int, year: int, bill: Bill) -> None**

Parameters

* month: the month to advance to
* year: the year to advance to
* bill: the Bill associated with this month and year

Usage

* Sets the bill for the new month at the contract’s monthly fee

Precondition

* bill must exist for this month and year

**bill\_call(call: Call) -> None**

Parameters

* call: the Call to bill from

Usage

* Bills the call with the contract’s per minute rate of a call

Precondition

* call must be a valid Call (see definition at top)

**cancel\_contract() -> Float**

Usage

* Returns the amount due with no penalties

# PrepaidContract

(Inherited from Contract class)

## Description

A prepaid phone contract where users pay in advance and usage deducts from a balance. The user must pay an initial deposit, and left-over balance will be carried to next month.

* If the user has less than $10 in credit, then the user must top up $25

## Public Methods

**\_\_init\_\_(start: datetime.datetime, balance: float) -> None**

Parameters

* start: the start date of the contract
* balance: the initial balance the user must deposit

Usage

* The initializer for the prepaid contract class. Must pass in the start as a datetime object and the initial balance the user must top up to start the contract

**new\_month(month: int, year: int, bill: Bill) -> None**

Parameters

* month: the month to advance to
* year: the year to advance to
* bill: the Bill associated with this month and year

Usage

* Advances the contract to the new month and year and carries over the previous balance
* if balance is below $10, tops up $25 (not added to the bill)

Precondition

* must have a bill associated with the month and year specified

**bill\_call(call: Call) -> None:**

Parameters:

* call: the Call to bill from

Usage:

* deducts the call cost from the balance with the prepaid per minutes rate

Precondition

* call must be a valid Call (see definition at top)

**cancel\_contract() -> Float:**

Usage

* cancels the prepaid contract.
* If the user has credit balance left, the amount will be forfeited. (return 0)
* If the user has a negative credit balance, the method will return the amount the user owes to the company

# **Implementation Details**

# TermContract class

(Inherited from Contract class)

## Attributes:

\_end: the end date of the contract

\_curr\_month: stores the current month of the contract

\_curr\_year: stores the current year of the contract

\_free\_minutes: stores the free minutes remaining for this montn

\_monthly\_fee: stores the monthly fee of the contract

\_deposit: stores the required deposit for the term

\_mins\_cost: stores the per minute cost of a call or per sms

## Methods

**\_\_init\_\_(start: datetime.datetime, end: datetime.datetime) -> None**

* The initializer for this class. The start will define the start date of the contract. The end will define the end date of the contract

**(Override) new\_month(month: int, year: int, bill: Bill) -> None**

* Update \_curr\_month and \_curr\_year to the month and year specified
* Set the bill of the contract to the bill passed in from the argument and sets the type of the bill to be ‘TERM’ with a rate at \_mins\_cost
* If the month and year specified is the start date of the contract, add a fixed cost of \_deposit to the bill
* If the month and year advance to is before the contract start date or contract is cancelled start is None, then the input is invalid and exit out of method

**(Override) bill\_call(call: Call) -> None**

* Calculates the free minutes remaining by subtracting the duration of the call (converted in minutes) from the free minutes available \_free\_minutes
* If the free minutes remaining is negative, then the overused amount is added to the bill
* If the free there is free minutes remaining, calculate the free\_minutes used and bill it as well as updating the \_free\_minutes to account for the duration used.

**(Override) cancel\_contract() -> float**

* Set the start date to None (indicate that the contract is cancelled) and get the cost of the bill . If the user cancels after the end of the term, the \_deposit will be refunded and subtracted from the cost. The total that user owes to the company will be returned as a float

# MTMContract class

(Inherited from Contract class)

## Attributes

\_monthly\_fee: the monthly fee of the MTM contract

\_mins\_cost: the per minute cost of the MTM contract

## Methods

**\_\_init\_\_(start: datetime.datetime) -> None**

* Initalizes a new month-to-month contract with the start date specified

**(Override) new\_month(month: int, year: int, bill: Bill) -> None**

* Sets the bill to the bill passed in, and set the type of the bill to be ‘MTM’, with a rate of \_mins\_cost
* Add the cost per month of the contract to the fix cost of the bill of this month
* If the month and year advance to is before the contract start date or contract is cancelled start is None, then the input is invalid and exit out of method

# PrepaidContract

(Inherited from Contract class)

## Attributes

\_balance: current balance of the prepaid contract, negative balance represents the credit the user has

\_mins\_cost: the per minute cost of a call with prepaid contract

\_intial\_deposit: stores the initial credit customer put in to start this contract

## Methods

**\_\_init\_\_(start: datetime.datetime, balance: float) -> None**

* Sets the start date of the prepaid contract to the start, and set \_balance to the negative of balance. And sets \_initial\_deposit to balance

**(Override) new\_month(month: int, year: int, bill: Bill) -> None**

* Sets the bill of the contract to the bill of the new month and year passed in. And also sets the type of the bill to be ‘PREPAID’ with a rate of \_mins\_cost
* If the balance is greater than -10 (has less than $10 in credit), then add $25 in credit (subtract 25 from balance)
* Carry over the bill by add fixed cost of \_balance to the bill
* If the month and year advance to is before the contract start date or contract is cancelled start is None, then the input is invalid and exit out of method

**(Override) bill\_call(call: Call) -> None:**

* Update the balance by adding the cost of the call by converting the duration of the call to minutes and round up to the nearest minute and multiply \_mins\_cost
* Update bill by adding billed minutes of the duration of the call rounded up to nearest minutes

**(Override) cancel\_contract() -> float:**

* Set start to None to indicate end of contract
* If \_balance is less than or equal to 0, then the user has left over credit, which is forfeited and should return 0 (user owes $0 to the company). Fixed cost of the bill bill.fixed\_cost is reset to 0 and bill.billed\_min is reset to 0
* Else return the amount \_balance user owes to the company