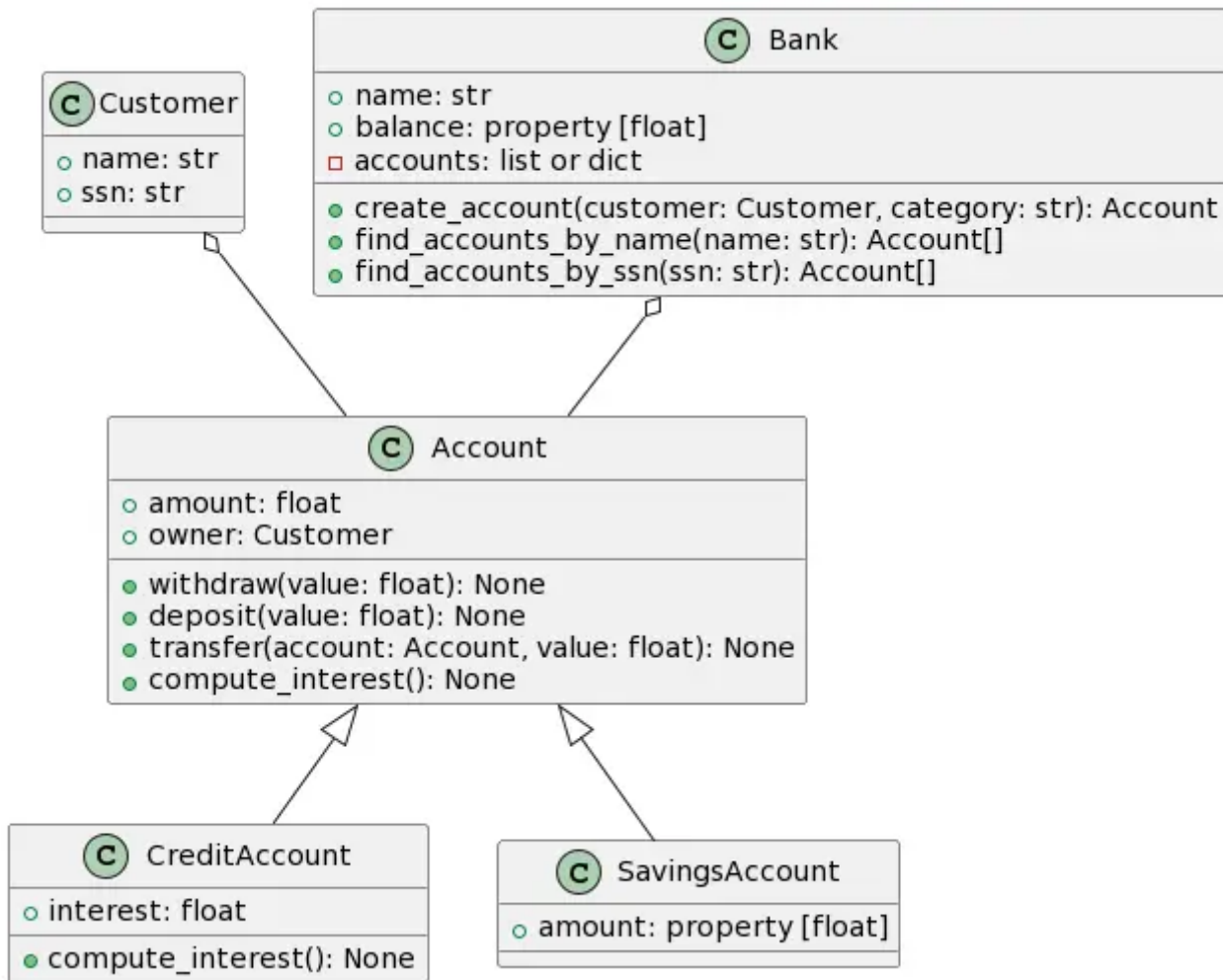


ACIT2515

Lab: bank and bank accounts

Class diagram



1. Use the `Customer` class

A customer has:

- a name
- a SSN (Social Security Number)

The code for this class is already provided.

2. Build the `Account` class

A bank account has:

- "owner": a `Customer` instance

- an amount (float, default value: 0)
- a method `deposit`
 - it receives an `int` or `float` argument: the amount to be deposited. If the argument is negative, raise a `AttributeError` exception.
 - it adds the deposited amount to the account amount
- a method `withdraw`
 - it receives an `int` or `float` argument: the amount to be withdrawn. If the argument is negative, raise a `AttributeError` exception.
 - it removes the amount provided from the account
- a method `transfer`, to make transfers between accounts
 - it has two arguments: `account` and `amount`
 - it must raise a `TypeError` exception if the account is not an instance of `Account`
 - the method withdraws `amount` from the current instance and deposits it into the `account`

3. Build child classes

3.1. `CreditAccount`

- The credit account inherits from `Account`. Its amount is typically negative.
- Its constructor receives an additional argument: the interest rate (a number between 0 and 100).
- This is the interest rate in %. Store it as an attribute of your class.
- The `compute_interest` method, **if the amount is negative**:
 - charges the interest to the account: $\text{amount} = \text{amount} * (100 + \text{interest_rate}) / 100$
 - then charges \$10 to the account (administration fees)

3.3. `SavingsAccount`

- The savings account inherits from `Account`.
- A savings account cannot have an amount less than 0. Use a `@property`!

4. Build the `Bank`

- A bank has a name (received by the constructor)
- `create_account`: creates an account in the bank. Receives two mandatory arguments
 - `category`: can be either "account", "credit", or "savings"
 - `owner`: a `Customer` instance
 - and an optional argument: `interest_rate` (default value = 0)
 - credit accounts must use this value for the interest rate
 - this method creates an account of the specified type, associates it with the provided owner
 - the method **returns** the account created, but you need to make sure your bank keeps track of the accounts (you could use a dictionary, or a list)
- `find_accounts_by_ssn()`: receives one argument

- first argument is a string: the social security number of a customer
- the method returns the list of account(s) associated with the given customer
- `find_accounts_by_name()`: receives one argument
 - first argument is a string: (part of) the name of the customer
 - the method returns the list of account(s) belonging to customers whose name matches the string provided
- `balance`: a **property**
 - its value is the sum of all amounts of all bank accounts managed by the bank

Grading rubric

- 1 mark for each test that passes (13 total)
- 4 marks for PEP8 syntax, docstrings, and code quality

TOTAL = 17 marks