ATM exercise

1: Account model

Your task is to create an **Account** class, which will be the model for a bank account.

Constant data

- Transactions limit (max. number of transactions without fees)
- Fee per transaction after transaction limit is reached
- Overdrawn transaction base fee
- Overdrawn transaction fee per dollar overdrawn
- Account number (unique to each account)

Variable data

- Name
- Balance (how much money is in the account)
- Total number of transactions
- Total amount owed in fees
- Total amount paid in fees

Functionalities: Actions

- Deposit an amount of money into account
  - Validate the amount to deposit before applying action
  - Don’t apply the action directly; instead call the private setter method for the balance
  - Return a Boolean error code
- Withdraw an amount of money from account
  - Validate the amount to withdraw before applying action
  - Don’t apply the action directly; instead call the private setter method for the balance
  - Return a Boolean error code
- Pay an amount of owed fees
  - Validate the amount to pay before applying action
  - Return a Boolean error code
- Set name
- [Private: Set balance]
  - Apply the new value
  - If the account is overdrawn, charge a fee (= base fee + fee per dollar overdrawn)
  - If the transactions limit is exceeded, charge a fee

Functionalities: Data access

- Field getter methods for all private fields
- Is the account overdrawn? (Overdrawn = Balance is negative)
- Get the number of remaining allowed transactions
- Get the amount of unpaid fees owed
- Get a string description of the account