131 lines (100 sloc) | 4.1 KB Raw Blame

```python
from datetime import datetime

# Custom Exceptions for Bank Transactions!!
class TransactionDeclined(Exception):
    pass

class InsufficientBalance(Exception):
    pass

class MaxWithdrawLimtExceeded(Exception):
    pass


class BankAccount:
    interest_rate = 4.0
    def __init__(self, fname, lname, amount):
        self.fname = fname
        self.lname = lname
        self.balance = float(amount)
        self._transactions = [ ]
        self._transactions.append(f"{datetime.now()} ***Initial Deposit*** {s

    def deposit(self, amount):
        self.balance += float(amount)
        self._transactions.append(f'{datetime.now()} Deposited Amount: {amoun

```

```
27      def withdraw(self, amount):
28          if amount <= self.balance:
29              self.balance -= amount
30              self._transactions.append(f'{datetime.now()} Withdrawn Amount: {a
31          else:
32              raise InsufficientBalance()
33
34      def statement(self):
35          for line in self._transactions:
36              print(line)
37          print(f"Total Account Balance: {self.balance}")
38
39      def roi(self):
40          self.balance = self.balance + self.balance * (self.interest_rate / 10
41
42
43  class SavingsAccount(BankAccount):
44      interest_rate = 4.0
45      def withdraw(self, amount):
46          if amount > 10000:
47              raise MaxWithdrawLimtExceeded('Can not withdrawn more than 10000
48          super().withdraw(amount)
49
50
51  class SalaryAccount(BankAccount):
52      MAX_DRAFT_AMOUNT = 10000
53      def __init__(self, name):
54          self.is_first_month_salary = True
55          self.draft_amount = 0
56          super().__init__(name, 0.00)
57
58      def deposit(self, amount):
59          if self.is_first_month_salary:
60              self.is_first_month_salary = False
61              super().deposit(amount + 1000)
62          else:
63              super().deposit(amount)
64
65      def overdraft(self, amount):
66          if self.draft_amount + amount <= self.MAX_DRAFT_AMOUNT:
67              self.draft_amount += amount
68              super().deposit(amount)     # handover the amount to deposit meth
69          else:
70              raise ValueError(f"Max available draft amount is {self.MAX_DRAFT_
71
```

```
72
73  class SeniorCitizenAccount(BankAccount):
74      interest_rate = 5.5
75
76      def withdraw(self, amount):
77          if amount > 20000:
78              raise MaxWithdrawLimtExceeded('Can not withdraw more than 20000 p
79          super().withdraw(amount)
80
81
82  class SukanyaSamrudhiAccount(BankAccount):
83      interest_rate = 9.5
84
85      def deposit(self, amount):
86          if amount < 1000:
87              raise ValueError('Min Amount Should be 1000rs')
88          super().deposit(amount)
89
90      # Completely overriding the parent class method "withdraw"
91      def withdraw(self, amount):
92          raise TransactionDeclined("Can not withdraw")
93
94
95  class PenaltyAccount:
96      def withdraw(self, amount):
97          self.balance -= 200  # Penalty for withdrawing from PensionAccount
98          super().withdraw(amount)
99
100 class RetirementAccount(PenaltyAccount, BankAccount):
101     pass
102
103 # ====================================================
104 # Alternate solution
105 class PenaltyAccount:
106     penalty = 0
107     def withdraw(self, amount):
108         self.balance -= self.__class__.penalty  # Penalty for withdrawing fro
109         super().withdraw(amount)
110
111 class RetirementAccount(PenaltyAccount, BankAccount):
112     penalty = 200       # Over-riding class variable
113
114 class MaxTransactionLmit(PenaltyAccount, BankAccount):
115     penalty = 1000  # Over-riding class variable
116
```

```python
# ====================================================
# Alternate solution
class PenaltyAccount:
    def __init__(self, penalty_amount):
        self.penalty_amount = penalty_amount

    def withdraw(self, amount):
        self.balance -= self.penalty_amount
        super().withdraw(amount)

class PensionAccount(PenaltyAccount, BankAccount):
    def __init__(self, penalty_amount, name, balance):
        PenaltyAccount.__init__(self, penalty_amount)       # Initialise the
        BankAccount.__init__(self, name, balance)       # Initialise the cons
# ====================================================
```