Bank Account Hierarchy

Your job is to create Python child classes that model bank accounts.

A. Start with *BankAccount.py* (Copy or rename if you like) and change it so the following rules apply:

1. The class has four instance variables, which you may name as you like; the revised class has the same instance variables as the previous. Make these changes in the processing of the instance fields:
    1. **Use class variables to autogen the acctID field**
    2. Before setting the balance property, check if the argument passed to the CTOR is negative. **If so, set argument to 0 and print 'invalid balance amount' message, then set the balance property.**
2. Properties coded for the balance field. The balance must be **non-negative** (change from previous class).
3. A make_deposit method that accepts a deposit amount as an argument. The argument must exceed 0 (same as previous class).
4. A make withdrawal method that accepts a withdrawal amount. The argument must exceed 0 (no change from previous class) **and overdrafts are allowed** (change from previous class)
5. The revised class **has no transfer method**
6. A print method that displays the field names and values. **Overload a dunder method to facilitate printing** (change from previous)
7. For items 2, 3, 4: if the field or argument is invalid, print a message describing the issue.

    A solution is **BackAcctInheritance.py**

B. Create a class named **Savings**. This class:

1. Is a child class of BankAccount

2. Contains an **instance variable num_deposits** that counts the number of deposits made for an account

3. Has a **make_deposit** method that:

    Leverages the functionality of the make_deposit method in the parent class

    Only allows deposits of 9_999 or less to apply. If this method is called with a deposit amount exceeding 9_999, print a diagnostic and *do not make a deposit*

    If deposit amount <= 9_999, apply the deposit. Remember, this class keeps track of the number of deposits made!

4. A print method that displays the field names and values. **Overload a dunder method to facilitate printing** and leverage the print method in the parent class. Try to match the result of the print method to the sample outputs shown on the last page (need not be exact!)

A solution is **SavingsSoln.py**

C. Create a class named **Checking**. This class:

1. Is a child class of BankAccount

2. Contains an **instance variable num_withdrawals** that counts the number of withdrawals made for an account

3. Has a **make_withdrawal** method that:

    Leverages the functionality of the make_withdrawal method in the parent class

    If the resulting balance after withdrawal is negative, **apply an NSF fee of 35$**. IOW, overdrafts are permitted but a feel applies.

    If deposit amount <= 9_999, apply the deposit. Remember, this class keeps track of the number of withdrawals made!

4. A print method that displays the field names and values. **Overload a dunder method to facilitate printing** and leverage the print method in the parent class. Try to match the result of the print method to the sample outputs shown on the last page (need not be exact!)

A solution is **CheckingSoln.py**

You have a starter/driver class, **useBankAccountsSkel.py**, that has instructions on how to produce the outputs shown on the next page. Note the first line of output is the 'invalid initial balance' message you coded to reject the negative balance for one of the accounts.

```
# Any required imports go here

# Create some accounts
accts = [Savings("saHolder1", 123_456), Checking("chHolder1", 34_456),
        Savings("holder2", -234), Savings("saHolder3", 23_456),
        Checking("chHolder2", 4_456), Checking("chholder3", 234)]
# Print accts[0] (a Savings acct) and accts[1] (a Checking account)

# Deposit 15_000 in a accts[0]; reprint account


# Deposit 5_000 in a accts[0]; reprint account

# Withdraw 2_000 in accts[1]; reprint account
```

And your expected output:

```
Invalid initial balance amount -234. Initial balance set to 0
**************************************************
Account type: Savings
Account info for account ID: 1
Holder: saHolder1      Date opened: 2022-04-21 09:02:53.942472  Balance: 123,456
Num deposits: 0
**************************************************
**************************************************
Account type: Checking
Account info for account ID: 2
Holder: chHolder1      Date opened: 2022-04-21 09:02:53.942472  Balance: 34,456
Num withdrawals: 0
**************************************************
Deposit amount of 15000 exceeds max (9999) for this type of account
Deposit not done; balance unchanged
**************************************************
Account type: Savings
Account info for account ID: 1
Holder: saHolder1      Date opened: 2022-04-21 09:02:53.942472  Balance: 123,456
Num deposits: 0
**************************************************
**************************************************
Account type: Savings
Account info for account ID: 1
Holder: saHolder1      Date opened: 2022-04-21 09:02:53.942472  Balance: 128,456
Num deposits: 1
**************************************************
**************************************************
Account type: Checking
Account info for account ID: 2
Holder: chHolder1      Date opened: 2022-04-21 09:02:53.942472  Balance: 32,456
Num withdrawals: 1
**************************************************
```