Bank Account manager

```
class Account:
    def __init__(self,acct_nbr,opening_deposit):
        self.acct_nbr = acct_nbr
        self.balance = opening_deposit

    def __str__(self):
        return f'${self.balance:.2f}'

    def deposit(self,dep_amt):
        self.balance += dep_amt

    def withdraw(self,wd_amt):
        if self.balance >= wd_amt:
            self.balance -= wd_amt
        else:
            print('Funds Unavailable')

class Savings(Account):
    def __init__(self,acct_nbr,opening_deposit):
        # Run the base class __init__
        super().__init__(acct_nbr,opening_deposit)

    # Define a __str__ method that returns a string specific to Savings accounts
    def __str__(self):
        return f'Savings Account #{self.acct_nbr}\n  Balance: {Account.__str__(self)}'


class Business(Account):
    def __init__(self,acct_nbr,opening_deposit):
        # Run the base class __init__
        super().__init__(acct_nbr,opening_deposit)

    # Define a __str__ method that returns a string specific to Business accounts
    def __str__(self):
        return f'Business Account #{self.acct_nbr}\n  Balance: {Account.__str__(self)}'
```

```python
class Customer:
    def __init__(self, name, PIN):
        self.name = name
        self.PIN = PIN
        self.accts = {'C':[],'S':[],'B':[]}

    def __str__(self):
        return self.name

    def open_checking(self,acct_nbr,opening_deposit):
        self.accts['C'].append(Checking(acct_nbr,opening_deposit))

    def open_savings(self,acct_nbr,opening_deposit):
        self.accts['S'].append(Savings(acct_nbr,opening_deposit))

    def open_business(self,acct_nbr,opening_deposit):
        self.accts['B'].append(Business(acct_nbr,opening_deposit))

    def get_total_deposits(self):
        total = 0
        for acct in self.accts['C']:
            print(acct)
            total += acct.balance
        for acct in self.accts['S']:
            print(acct)
            total += acct.balance
        for acct in self.accts['B']:
            print(acct)
            total += acct.balance
        print(f'Combined Deposits: ${total:.2f}') # added precision formatting here

def make_dep(cust,acct_type,acct_num,dep_amt):
    for acct in cust.accts[acct_type]:
        if acct.acct_nbr == acct_num:
            acct.deposit(dep_amt)

def make_wd(cust,acct_type,acct_num,wd_amt):
    for acct in cust.accts[acct_type]:
        if acct.acct_nbr == acct_num:
            acct.withdraw(wd_amt)
```

```
class Checking(Account):
   def __init__(self,acct_nbr,opening_deposit):
       super().__init__(acct_nbr,opening_deposit)

   def __str__(self):
       return f'Checking Account #{self.acct_nbr}\n  Balance: {Account.__str__(self)}'


class Savings(Account):
   def __init__(self,acct_nbr,opening_deposit):
       super().__init__(acct_nbr,opening_deposit)

   def __str__(self):
       return f'Savings Account #{self.acct_nbr}\n  Balance: {Account.__str__(self)}'


class Business(Account):
   def __init__(self,acct_nbr,opening_deposit):
       super().__init__(acct_nbr,opening_deposit)

   def __str__(self):
       return f'Business Account #{self.acct_nbr}\n  Balance: {Account.__str__(self)}'
```