Decision coverage of withdraw

Made by Chinese-broccoli

Author: Mason Chang, Yueyang Liu, Ruikang Luo.

```
    def processWithdraw(self, toAccount, amount, fromAccount):
        match, idx = self.binarySearch(fromAccount)
      ① if match:
            decreaseAccount = self.masterList[idx].split(' ')
            name = decreaseAccount[2].strip()
            oldAmount = decreaseAccount[1]
            newAmount = int(oldAmount) - int(amount)
      ②    if newAmount < 0 :
                print(f"Transaction: WDR {toAccount} {amount} {fromAccount} *** I
                      s invalid!")
                print("Due to: insufficient balance.")
                return False
      ③    else:
                updateInfo = f"{fromAccount} {newAmount} {name}\n"
      ④        if self.checkUpdateInfo(updateInfo):
                    self.masterList[idx] = updateInfo
      ⑤        else:
                    print(f"Transaction: WDR {toAccount} {amount} {fromAccount} *
                          ** is invalid!")
                    print("Due to: Invalid length.")
                    return  False
      ⑥ else:
            print(f"Transaction: WDR {toAccount} {amount} {fromAccount} *** is in
                  valid!")
            print("Due to: no matching account found!")
            return False
        return True
```

```
    def binarySearch(self,targetAccount):
        targetAccount = int(targetAccount)
        start = 0
        end = self.length - 1
        # prepare for the return when not find the account number
        middle = (start + end) // 2
      ⑦ while start <= end:
            middle = (start + end) // 2
```

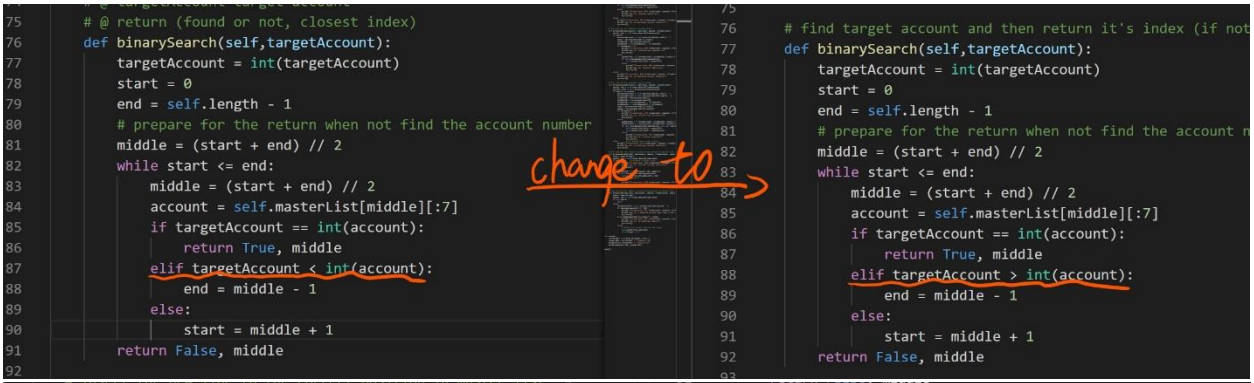
```
            account = self.masterList[middle][:7]
⑧           if targetAccount == int(account):
                return True, middle
⑨           elif targetAccount > int(account):
                end = middle - 1
⑩           else:
                start = middle + 1
        return False, middle
```

Test No.	Input transaction	Decision	Terminal output	Description
T1.0	WDR 0000000 100000 1000327 ***	① True ⑥ False ② False ③ True ④ True ⑤ False ⑦ True ⑧ True ⑨ False ⑩ True	None	Successfully withdraw, account is in the bottom of the master file
T1.1	WDR 0000000 100000 1000329 ***	① True ⑥ False ② False ③ True ④ True ⑤ False ⑦ True ⑧ True ⑨ False ⑩ True	None	Successfully withdraw, account is in the top of the master file
T1.2	WDR 0000000 10000 1000330 ***	① False ⑥ True ⑦ False ⑧ False	Transaction: WDR 0000000 10000 1000330 *** is invalid! Due to: no matching account found!	Unsuccessfully withdraw, due to no matching account found
Notes: up to now, both side of direction (⑨ , ⑩), whether within given range (⑦), and matching or not (⑧) have tested. For simplicity, they will not be stated in the below table.				
T1.3	WDR 0000000 100000 1000328 ***	① True ⑥ False ② True ③ False	Transaction: WDR 0000000 100000 1000328 *** is invalid! Due to: insufficient balance!	Unsuccessfully withdraw, due to insufficient balances

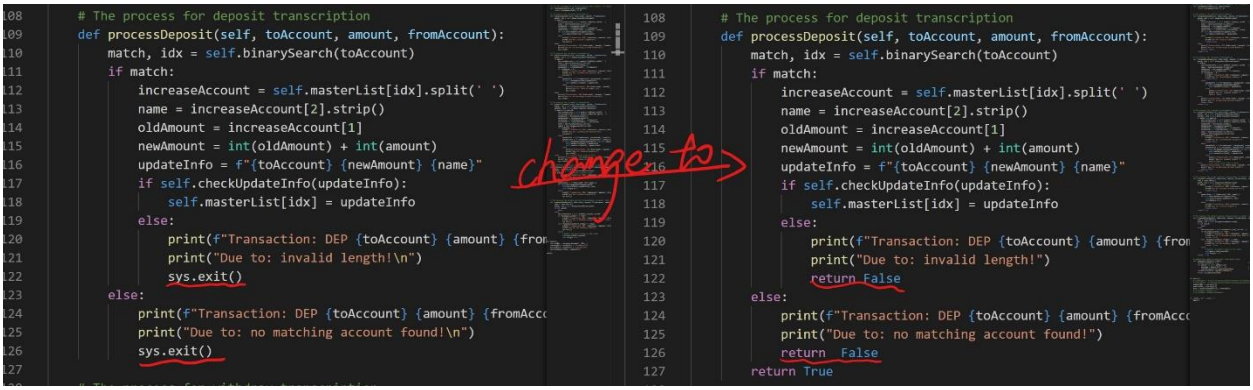
T1.4	WDR 0000000 10000 1000328 ***	① True ⑥ False ② False ③ True ④ False ⑤ True	Transaction: WDR 0000000 10000 1000328 *** is invalid! Due to: Invalid length!	Unsuccessfully withdraw, due to the account information is longer than 46 characters.
Sum up: both sides of ① is tested at T1.0 and T1.2; ② is tested at T1.0 and T1.3; ③ is tested at T1.0 and T1.3; ④ is tested at T1.0 and T1.4; ⑤ is tested at T1.0 and T1.4; ⑥ is tested at T1.0 and T1.2;				

Failure spotted at T1.0 and T1.2

T1.0, found that the account list should be stored in descending order



T1.2, found that sys.exit() can't be used in pytest. Changing sys.exit() to return False (For simplicity, will not post more similar example.)




```python
    # The process for deposit transcription
    def processDeposit(self, toAccount, amount, fromAccount):
        match, idx = self.binarySearch(toAccount)
        if match:
            increaseAccount = self.masterList[idx].split(' ')
            name = increaseAccount[2].strip()
            oldAmount = increaseAccount[1]
            newAmount = int(oldAmount) + int(amount)
            updateInfo = f"{toAccount} {newAmount} {name}"
            if self.checkUpdateInfo(updateInfo):
                self.masterList[idx] = updateInfo
            else:
                print(f"Transaction: DEP {toAccount} {amount} {fro
                print("Due to: invalid length!\n")
                sys.exit()
        else:
            print(f"Transaction: DEP {toAccount} {amount} {fromAcc
            print("Due to: no matching account found!\n")
            sys.exit()
```

```python
    # The process for deposit transcription
    def processDeposit(self, toAccount, amount, fromAccount):
        match, idx = self.binarySearch(toAccount)
        if match:
            increaseAccount = self.masterList[idx].split(' ')
            name = increaseAccount[2].strip()
            oldAmount = increaseAccount[1]
            newAmount = int(oldAmount) + int(amount)
            updateInfo = f"{toAccount} {newAmount} {name}"
            if self.checkUpdateInfo(updateInfo):
                self.masterList[idx] = updateInfo
            else:
                print(f"Transaction: DEP {toAccount} {amount} {fro
                print("Due to: invalid length!")
                return False
        else:
            print(f"Transaction: DEP {toAccount} {amount} {fromAcc
            print("Due to: no matching account found!")
            return False
        return True
```