Practical Test 4

1. Setting up for Practical Test 4

Create and work in a **FOP/PracTest4** directory for the test.

2. Download and modify a Python program

Download **accounts.py** and **testAccounts.py** from the Assessments under Practical Test 4. Note this implementation has a Portfolio class to hold a collection of bank accounts

Modify **testAccounts.py** to:

1. Create an account "Castle", a/c number "999999-1", balance $1000
2. Create an account "Shrubbery", a/c number "999999-2", balance $100
3. Print the balances of the accounts
   <test your code to here before going on>
4. Deposit $100 into the Castle account
5. Withdraw $10 from the Shrubbery account
6. Withdraw $1000 from the Shrubbery account
7. Print the balances of the accounts
   <test your code to here before going on>

Modify **accounts.py** to:

1. Complete the code for getNumAccounts() and getTotalBalance()

Modify **testAccounts.py** to:

1. Add another account “Grail”, a/c number "999999-3", balance $100 (can be included at start of program, with other accounts)
2. Withdraw $1000 from the Grail account
3. Use and test getNumAccounts() and getTotalBalance() – printing out their results

Modify **accounts.py** to:

1. **Add exception handling**:
   Modify **withdraw()** to check if the balance is able to cover the withdrawal amount.
   a. If not, raise an **InsufficientFundsError exception** (in the withdraw method) and leave the balance unchanged. Insufficient funds error is defined in accounts.py

Modify **testAccounts.py** to:

1. **Add exception handling**:
   Handle the possible exceptions for each withdrawal in **testAccounts.py** **(try/except)** – print an explanatory message if an exception is thrown.

3. Update the README file

Copy or create a README file for the test, update it to include your files.

4. Submission

Submit you test via Blackboard using the link on the Assessment page. Go to your FOP directory and type: zip –r PracTest4_ID PracTest4

End of Test