1. Create a class DigitalWallet to represent a user account for a service like Paypal.

Each DigitalWallet has the following properties:

- `accountId` (long)
- `balance` (non-negative long, represents a number of pennies – default value of zero)
- `transactions` (an object of type TransactionHistory, more details below)

There should be a getters and setters for all class variables.

For security purposes, no customer is allowed to withdraw more than 50% of their balance at one time. Therefore the class needs a single variable to track that percentage: double `maxWithdrawalPct`. That percentage may change over time and should be available for modification via a getter and setter.

DigitalWallet will have two constructors:

- The first constructor has just one parameter. That parameter is `accountId`
- The second constructor takes the `accountId` and an `initialDeposit`.

Both constructors initialize all properties appropriately. Name your constructor parameters exactly as above.

DigitalWallet will have public methods:

- `boolean canWithdraw(long amount)` (returns true if the amount is withdrawable from this wallet)
- `void withdraw(long amount)` (deducts from `balance` and calls addTransaction() on the transaction history. The argument to addTransaction should be a negative value. You must not allow the `amount` parameter to be negative. Also, disallow withdrawals over the permitted amount as controlled by `maxWithdrawalPct`.)
- `void deposit(long amount)` (adds to `balance` and calls addTransaction() on the transaction history. The argument to addTransaction should be a positive value. You must not allow the `amount` parameter to be negative.)

Print an error message if any of the rules are violated.

Your DigitalWallet class is composed with a TransactionHistory. TransactionHistory's constructors and methods are:

- `public TransactionHistory()` (no-arg constructor)
- `public TransactionHistory(long amt)` (constructor taking a first transaction.)
- `public void addTransaction(long amt)` (adds a transaction to history.)
- `public long[] getTransactions()` (returns an array of all transactions on the DigitalWallet. The array contains positive values to indicate deposit and negative values to indicate withdrawal.)

You will need to use a TransactionHistory instance in DigitalWallet, you do not need to implement it.
2. Write a class TestDigitalWallet that has only the main method.

The main method should instantiate two DigitalWallet’s ‘w1’ and ‘w2’.

For ‘w1’, instantiate the DigitalWallet with the one-parameter constructor and pass an account id ‘1’. Deposit $100, then withdraw $50. Then set the maxWithdrawalPct to 75%. Finally print the results of ‘canWithdraw(4000)’. Put in comments what the output of the print statement would be.

Next, instantiate ‘w2’ using the account id ‘2’ and an initial deposit of $200. Make 2 withdrawals of $50. Finally confirm that the sum of the transaction history equals a call to getBalance().
3. Write one more method for DigitalWallet.

Below you will be provided with the method header. You must fill in the body of the method. The method will perform a transfer of funds between two accounts.

If you had two DigitalWallet instances 'w1' and 'w2', in order to transfer funds between the two wallets you would make the following call:

```java
w1.transferFunds(w2, 4000);
```

Here is the method, fill in the body (note: use the this keyword):

```java
public void transferFunds (DigitalWallet wallet, long amount)
{
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
}
```