Program Testing Document

PROJECT #1

KAMERON DAMASKA
EECS 132

Class: Account

This class records the amount of money an individual owes.

Account Constructors

First constructor

This constructor takes a String input. The number associated with the account is set to the String value

```
> Account x = new Account("11")
> x.getAccountNumber()
"11"
```

Second constructor

This constructor takes a String input and an int input. The number associated with the account is set to the String input and the balance limit is set to the int input.

```
> Account x = new Account("11", 5000);
> x.getAccountNumber()
"11"
> x.getBalanceLimit()
5000
```

Account methods

getAccountNumber

This method should return the account number, as a String, for an account.

getAccountNumber test:

```
>Account One = new Account("100")
> One.getAccountNumber()
"100"
```

getBalance

This method should return the current balance, stored as a double, for the given account.

charge

This method should increase the current balance of an account by the amount inputted. The input is a double.

credit

This method should decrease the current balance of an account by the amount inputted. The input is a double.

getBalance, charge, and credit test:

```
> One.getBalance()
0.0
> One.charge(400)
> One.getBalance()
400.0
> One.credit(400)
> One.getBalance()
0.0
> One.credit(400.25)
> One.getBalance()
-400.25
```

setBalanceLimit

Sets what the maximum balance of an account should be. It takes an int input and does not return anything.

getBalanceLimit

This method should recall what value the account's balance limit was set at. It takes no inputs and returns an int value equal to setBalanceLimit.

setBalanceLimit and getBalanceLimit test:

```
> One.setBalanceLimit(5000);
> One.getBalanceLimit()
5000
>One.setBalanceLimit(4000);
>One.getBalanceLimit()
4000
```

Class: LibraryAccount

This class tracks an individual's financial record at a library. It records overdue materials and any fines towards the individual.

LibraryAccount Constructors

First Constructor

This constructor takes in a String input, and creates an object with the account number equal to the string input

```
> LibraryAccount x = new LibraryAccount("1001")
> x.getAccountNumber()
"1001"
```

Second Constructor

This constructor takes 4 inputs: A String, an int, and two double. The number associated with the account is set to the String input. The balance limit is set to the int input. The fine for books and reserved books are set to the double values, respectively.

```
> LibraryAccount x = new LibraryAccount("11", 5000,
11.5, 12.5);
> x.getBalanceLimit()
5000
> x.getAccountNumber()
"11"
> x.getBookFine()
11.5
> x.getReserveFine()
12.5
```

LibraryAccount methods

setBookFine

This method should set the value of the field bookFine to the double value inputted.

getBookFine

This method should recall the double value for the field bookFine.

getBookFine and setBookFine test:

```
> LibraryAccount One = new LibraryAccount("11")
> One.setBookFine(500);
> One.getBookFine()
500.0
> One.setBookFine(400);
> One.getBookFine()
400.0
```

setReserveFine

This method should set the value of the field reserveFine to the double value inputted.

getReserveFine

This method should recall the double value for the field reserveFine.

getReserveFine and setReserveFine test:

```
> One.setReserveFine(500);
> One.getReserveFine()
500.0
> One.setReserveFine(400);
> One.getReserveFine()
400.0
```

incrementOverdueBooks

This method should increase the int value stored in the field overdue by 1.

decrementOverdueBooks

This method should decrease the int value stored in the field overdue by 1.

getNumberOverdueBooks

This method should return the int value stored in the field overdue.

incrementOverdueBooks, decrementOverdueBooks, and getNumberOverdueBooks test:

```
> One.incrementOverdueBooks();
> One.getNumberOverdueBooks()
1
> One.decrementOverdueBooks();
> One.getNumberOverdueBooks()
0
```

incrementOverdueReserve

This method should increase the int value stored in the field reserveOverdue by 1.

decrementOverdueReserve

This method should decrease the int value stored in the field reserveOverdue by 1.

getNumberOverdueReserve

This method should return the int value stored in the field reserveOverdue.

incrementOverdueReserve, decrement-OverdueReserve, getNumberOverdueReserve test:

```
> One.incrementOverdueReserve()
> One.getNumberOverdueReserve()
1
> One.decrementOverdueReserve()
>
> One.getNumberOverdueReserve()
0
```

canBorrow

This method should return a boolean. If the value of balance is less than or equal to the value of balanceLimit, the method should return true. Otherwise, it should return false.

canBorrow test:

```
> One.getBalance()
5000.0
> One.setBalanceLimit(5000)
> One.canBorrow()
true
> One.setBalanceLimit(4999)
> One.canBorrow()
false
```

endOfDay

This method should increase the value of balance by the total amount of fines from overdue and overdue reserved books.

endOfDay test:

```
> One.getNumberOverdueReserve()
5
> One.getNumberOverdueBooks()
4
> One.getReserveFine()
500.0
> One.getBookFine()
500.0
> One.getBalance()
5000.0
> One.endOfDay();
> One.getBalance()
9500.0
```

```
5000 + 5 * 500 + 4 * 500 = 9500
```

Class: CreditAccount

This class tracks the amount an individual borrows and any interest charged.

CreditAccount Constructor

The constructor takes in a string value and a double value. A new CreditAccount object is created: the name is set to the String value and the interest rate is set to the double value.

```
> CreditAccount x = new CreditAccount("101", 1.14);
> x.getAccountNumber()
"101"
> x.getInterestRate()
1.14
```

CreditAccount Methods

setInterestRate

This method should take an int input and sets the field interestRate equal to the value inputted.

getInterestRate

This method should recall the int value stored in interestRate.

setInterestRate and getInterestRate test

```
> one.setInterestRate(.3);
> one.getInterestRate()
0.3
> one.setInterestRate(.4);
> one.getInterestRate()
0.4
```

getMonthlyPayment

This method returns the double value stored in monthlyPayment.

getAmountPaidThisMonth

This method returns the double value stoed in amountPaid.

endOfMonth

This method first checks to see if the value in monthlyPayment is greater than amountPaid. If so, the account is charged for an amount equal to balance multiplied by 1/12th of interestRate. If not, the account is not charged. Following this, monthlyPayment is set equal to the balance, and amountPaid is reset to 0.

credit

This method overwrites the method in the parent class Account. The method takes a double value called credit. The credit is added to the amountPaid and subtracted from the balance.

getMonthlyPayment, getAmountPaidThisMonth, endOfMonth, and credit test

```
> tuition.charge(1000);
> tuition.getBalance()
1000.0
> tuition.credit(500);
> tuition.getBalance()
500.0
> tuition.getAmountPaidThisMonth()
500.0
> tuition.endOfMonth();
> tuition.getMonthlyPayment()
500.0
> tuition.getAmountPaidThisMonth()
0.0
> tuition.getBalance()
500.0
> tuition.charge(500);
> tuition.getBalance()
1000.0
> tuition.getMonthlyPayment()
500.0
> tuition.getAmountPaidThisMonth()
0.0
> tuition.endOfMonth();
> tuition.getMonthlyPayment()
1016.6666666666666
> tuition.getBalance()
1016.6666666666666
```

Class: Student Account

This class represents a comprehensive financial account for a student. The account is divided into sections for tuition, dining charges, and library fees.

StudentAccount Constructor

This constructor takes a two String inputs. The first String input is sets the account number, and the second String input sets the name associated with the account.

```
> StudentAccount x = new StudentAccount("11", "John");
> x.getAccountNumber()
"11"
> x.getName()
"John"
```

Methods

setName

This method takes a String input that changes the student name associated with the account.

getName

This method returns the String value stored that represents the student's name

setAddress

This method takes a String input that sets/changes the address of the student.

getAddress

This method returns the String value stored that represents the student's address.

setName, getName, setAddress, getAddress test:

```
> StudentAccount student101 = new StudentAccount("101",
"Jonathan Smith");
> student101.getName()
"Jonathan Smith"
> student101.setName("John Smith")
> student101.getName()
"John Smith"
> student101.setAddress("1600 Washington Avenue");
> student101.getAddress()
"1600 Washington Avenue"
```

setLibraryAccount

This method takes a single LibraryAccount input value. This sets/changes the library account associated with this student account. The library account can be null.

getLibraryAccount

This method returns the library account associated with this student account.

setLibraryAccount, getLibraryAccount test:

```
> LibraryAccount johnAccount = new
LibraryAccount("101");
> student101.setLibraryAccount(jeffAccount);
> student101.getLibraryAccount()
LibraryAccount@6bc327a5
> student101.setLibraryAccount(null);
> student101.getLibraryAccount()
null
```

setTuitionAccount

This method takes a single CreditAccount input value. This sets/changes the tuition account associated with this student account. The tuition account can be null.

getTutionAccount

This method returns the tuition account associated with this student account.

setTuitionAccount, getTuitionAccount test:

```
> CreditAccount johnTuitionAccount = new
CreditAccount("101", .4);
> student101.getTuitionAccount()
null
> student101.setTuitionAccount(johnTuitionAccount);
> student101.getTuitionAccount()
CreditAccount@59b2b7c2
```

setDiningAccount

This method takes a single CreditAccount input value. This sets/changes the dining account associated with this student account. The tuition account can be null.

getDiningAccount

This method returns the dining account associated with this student account.

setDiningAccount, getDiningAccount test:

```
> CreditAccount johnDiningAccount = new
CreditAccount("101", .4);
> student101.getDiningAccount()
null
> student101.setDiningAccount(johnDiningAccount);
> student101.getDiningAccount()
CreditAccount@3a4f1d22
```

getBalance

This method overwrites the parent class's getBalance method. The method takes no inputs and sums of the balances in the library account, tuition account, and dining account. The method also subtracts the account's refund amount from this sum and returns the double value.

getBalance test:

```
> johnDiningAccount.getBalance()
3000.0
> johnTuitionAccount.getBalance()
5166.666666666667
> johnLibraryAccount.getBalance()
2500.0
> student101.getBalance()
10666.666666666667
```

```
> student101.getBalance()
10666.666666666667
> student101.getBalance()
10666.666666666667
> student101.getBalance()
10666.666666666667
```

charge

This overwrites the parent class's charge method. The method subtracts the input amount from the account's refund amount. If the difference is positive and the tuition account exists, the tuition account's balance will increase by the difference. If the tuition account does not exist, the account's refund amount will be set to the negation of the difference.>

charge test:

```
> x.getBalance() //x is a studentAccount object
5000.0
> lib.getBalance()
2000.0
> tuition.getBalance()
1500.0
> dining.getBalance()
1500.0
> x.charge(200);
> x.getBalance()
5200.0
> tuition.getBalance()
1700.0
> x.setTuitionAccount(null);
> x.getBalance()
3500.0
> x.charge(500);
> lib.getBalance()
2000.0
> dining.getBalance()
1500.0
> x.getBalance()
4000.0
> x.setTuitionAccount(tuition);
> x.getBalance()
5700.0
> x.charge(400);
> x.getBalance()
6100.0
```

Credit

This overwrites the parent class's credit method. The method processes a payment to the account, and decreases the balances in the accounts in the following order: tuition, dinig, library. Reduces the tuition account first. If any money is left after reducing the tuition account, the money reduces the dining account, then the library account. Any money left over will be added to the refund amount.
For the tuition and dining accounts, the balance will be reduced until the amount paid equals the necessary monthly payment. For the library account, the balance will not go below 0.

credit tests (with no null values):
Account balance:

```
> dining.getMonthlyPayment()
1500.0
> dining.getAmountPaidThisMonth()
0.0
> tuition.getMonthlyPayment()
1500.0
> tuition.getAmountPaidThisMonth()
0.0
> lib.getBalance()
2000.0
> x.getBalance()
5000.0
```

When credit > The amount needed to be paid off in tuition, it will pay off the Tuiton and start paying off Dining Account:

```
> x.credit(2000);
> tuition.getMonthlyPayment()
1500.0
> tuition.getAmountPaidThisMonth()
1500.0
> dining.getMonthlyPayment()
1500.0
> dining.getAmountPaidThisMonth()
500.0
> x.getBalance()
3000.0
```

When Credit > the amount needed to pay off dining, it will pay off the Dining Account and start paying off the Library Account:

```
> x.credit(1500);
> tuition.getMonthlyPayment()
1500.0
> tuition.getAmountPaidThisMonth()
1500.0
> dining.getMonthlyPayment()
1500.0
> dining.getAmountPaidThisMonth()
1500.0
> lib.getBalance()
1500.0
> x.getBalance()
1500.0
```

When credit > the amount needed to pay off library, it will pay off the library account, after all accounts are paid off, balance goes negative (refund amount is positive):

```
> x.credit(2000)
> tuition.getMonthlyPayment()
1500.0
> tuition.getAmountPaidThisMonth()
1500.0
> dining.getMonthlyPayment()
1500.0
> dining.getAmountPaidThisMonth()
1500.0
> lib.getBalance()
0.0
> x.getBalance()
-500.0
```

Credit test (library, tuition, and dining accounts are null)

```
> x.getBalance()
5000.0
> lib.getBalance()
2000.0
> x.setLibraryAccount(null);
> x.getBalance()
3000.0
> tuition.getBalance()
1500.0
> x.setTuitionAccount(null);
```

```
> x.getBalance()
1500.0
> dining.getBalance()
1500.0
> x.setDiningAccount(null);
> x.getBalance()
0.0
> x.credit(500);
> x.getBalance()
-500.0
```