Jacob Alspaw

EECS 132

**Testing Report**

Account Class

**Constructors**

1. The first constructor sets the line fee as type double, the rate as type double, the interest rate as type double, and the expected monthly usage as type int.

> Account a = new Account(1.0, 2.0, 3.0, 4.0, 5)

> a.getLineFee()

1.0

> a.getRate()

2.0

> a.getSalesTaxRate()

3.0

> a.getInterestRate()

4.0

> a.getPastYearUsage()

60

1. The second constructor sets the line fee as type double, the rate as type double, the interest rate as type double, the expected monthly usage as type int, the customer as type customer, and the date as type date.

> Account a = new Account(1.0, 2.0, 3.0, 4.0, 5, new Customer("Alspaw", "Adress"), new Date(1, 1))

> a.getLineFee()

1.0

> a.getRate()

2.0

> a.getSalesTaxRate()

3.0

> a.getInterestRate()

4.0

> a.getPastYearUsage()

60

> a.getCustomer()

Customer@123d14c

> a.getAnniversaryDate()

Date@e0108a

> a.isOpen()

true

**Methods**

1. void setLineFee(double lineFee): This method will set the monthly line fee of a specific account for having service and not return a value.

> Account a = new Account(1, 0, 0, 0, 0)

> a.setLineFee(2)

> a.getLineFee()

2.0

1. double getLineFee(): This method will return the monthly line fee for having service as a double.

> Account a = new Account(1, 0, 0, 0, 0)

> a.setLineFee(2)

> a.getLineFee()

2.0

1. void setRate(double rate): This method will set the rate of the account and will not return a value.

> Account a = new Account(0, 1, 0, 0, 0)

> a.setRate(2)

> a.getRate()

2.0

1. double getRate(): This method will return the rate of the account as a double.

> Account a = new Account(0, 1, 0, 0, 0)

> a.setRate(2)

> a.getRate()

2.0

1. void setSalesTaxRate(double taxRate): This method will set the sales tax rate as a percentage and will not return a value.

> Account a = new Account(0, 0, 1, 0, 0)

> a.setSalesTaxRate(2)

> a.getSalesTaxRate()

2.0

1. double getSalesTaxRate(): This method will return the sales tax rate in percentage form as a double.

> Account a = new Account(0, 0, 1, 0, 0)

> a.setSalesTaxRate(2)

> a.getSalesTaxRate()

2.0

1. void setInterestRate(double interestRate): This method sets the interest percentage that is applie int getDay(): returns the day of the date. The day should be a value between 1 and 31.

> Account a = new Account(0, 0, 0, 1, 0)

> a.setInterestRate(2)

> a.getInterestRate()

2.0

1. double getInterestRate(): This method will return the interest rate of the account as a double.

> Account a = new Account(0, 0, 0, 1, 0)

> a.setInterestRate(2)

> a.getInterestRate()

2.0

1. Void setMonthUsage(int monthUsage): This method sets the amount of the service used by the account this month and will not return a value.

> a.setMonthUsage(5)

> a.getMonthUsage()

5

1. Int getMonthUsage(): This method will return the amount of the service used by the account this month as an integer.

> a.setMonthUsage(5)

> a.getMonthUsage()

5

1. Int getPastYearUsage(): This method will return the amount of the service used by the account in the previous year as an integer.

> Account a = new Account(1, 1, 1, 1, 5)

> a.getPastYearUsage()

60

1. Double getBalance(): This method will return the current money balance on the account as a double.

> a.getBalance()

0.0

1. double getMonthBill(): This method returns the amount billed to the customer this month as a double.

> a.getMonthBill()

0.0

1. double getAmountPaidThisMonth(): This method will return the amount the customer has paid so far this month as a double.

> a.getAmountPaidThisMonth()

0.0

1. void receivePayment(double amount): This method decreases the balance by amount and increases the amount paid this month by amount and returns no value.

> a.getBalance()

0.0

> a.receivePayment(1)

> a.getBalance()

-1.0

1. void openAccount(Date date, Customer customer): This method sets the account's customer to customer and sets the account anniversary date to date while returning no value.

> a.openAccount(new Date(1, 1), new Customer("Alspaw", "Address"))

> a.getCustomer()

Customer@34c1b9

> a.getAnniversaryDate()

Date@2b74c4

1. boolean isOpen(): This method returns true if the account is open and false otherwise.

> Account a = new Account(1, 1, 1, 1, 5)

> a.isOpen()

false

> Account a = new Account(1, 1, 1, 1, 5, new Customer("Alspaw", "Address"), new Date(1, 1))

> a.isOpen()

true

1. Date getAnniversaryDate():This method will return the anniversary date for the account in Date type.

> Account a = new Account(1, 1, 1, 1, 5, new Customer("Alspaw", "Address"), new Date(2, 2))

> a.getAnniversaryDate()

Date@51b1fb

1. Customer getCustomer(): This method will return the customer for the account in Customer type.

> Account a = new Account(1, 1, 1, 1, 5, new Customer("Alspaw", "Address"), new Date(2, 2))

> a.getCustomer()

Customer@144d2cc

1. void endOfYearProcessing(): This method will initiate end of year processing for the account. If the account is not open, nothing is done. Otherwise the method does the following processing. If the account is not on a monthly plan, the expected monthy usage is set to the current year usage divided by 12. If the user is on a monthly plan, the expected monthly usage is the current year usage plus the difference between the current year usage and the previous year usage, the total divided by 12. Then, the previous year usage is set to the current year usage, and the current year usage is set to 0. The method will not return a value.

> Account a = new Account(1, 1, 1, 1, 5, new Customer("Alspaw", "Address"), new Date(2, 2))

> a.getPastYearUsage()

60

> a.getMonthUsage()

0

1. void endOfMonthProcessing(): This method will initiate end of month processing. If the account is not open, nothing is done. Otherwise, the method does the following processing. First, the current year usage is increased by the month usage. Second, the account balance is adjusted. The balance is increased by the line fee, by the product of the month usage and the rate, and by the sales tax applied to each of these two charges. In addition, if the total amount paid this month was less than the month bill, the interest rate is applied to the difference and this amount is added to the balance. Finally, the month bill is calculated. If the account is not on a monthly plan, the month bill is equal to the current balance or zero if the current balance is negative. If the account is on a monthly plan, the month bill is equal to the sum of the following: (a) the difference between the previous month bill and the amount paid this month, (b) the interest applied if there was a positive difference between the month bill and the amount paid, (c) the line fee, (d) the product of the rate and the expected monthly usage, and (e) the sales tax applied to both the line fee and the product of the rate and the expected monthly usage. The method will not return a value.

> Account acct = new Account(10, 0.10, 0.05, 0.20, 500)

> acct.openAccount(null, null);

> acct.getBalance()

0.0

> acct.setMonthUsage(1000)

> acct.endOfMonthProcessing()

> acct.getMonthBill()

115.5

> acct.getBalance()

115.5

/\*\* Could not get other scenarios to work\*/

1. boolean isOnMonthlyPlan(): This method will return true if the account is on a monthly plan and false otherwise.

> Account a = new Account(5, 5, 5, 5, 5)

> a.turnOnMonthlyPlan()

> a.isOnMonthlyPlan()

true

1. void turnOffMonthlyPlan(): This method sets the account to no longer be on a monthly plan. If the account is open, it sets the expected monthly usage to be the past year usage divided by 12. The method will not return a value.

> Account a = new Account(5, 5, 5, 5, 5)

> a.turnOffMonthlyPlan()

> a.isOnMonthlyPlan()

false

1. void turnOnMonthlyPlan(): This method sets the account to be on a monthly plan. If the account is open, it sets the expected monthly usage to be the difference between the past year usage and the current year usage (or zero if this difference is negative) divided by the number of months from the current date (where can you find it?) to the anniversary date (use the appropriate method below to retrieve the month difference). The method will not return a value.

> Account a = new Account(5, 5, 5, 5, 5)

> a.turnOnMonthlyPlan()

> a.isOnMonthlyPlan()

true

1. processDate(Date date): This method processes the date. If the account is not open, nothing is done. Otherwise, if the date's day and month are both equal to the anniversary date's day and month, the endOfYearProcessing method should be called. Then, if the date's day is equal to the anniversary date's day, the endOfMonthProcessing method should be called. The method will not return a value.

> Account a = new Account(5, 5, 5, 5, 5, new Customer("Alspaw", "Address"), new Date(2, 2))

> a.processDate(new Date(3, 3))

Date Class

**Constructor**

1. The first constructor sets the day as type int and month as type int and initializes the Date object with the given inputs.

> Date d = new Date(2, 2)

> d.getDay()

2

> d.getMonth()

2

**Methods**

1. int getDay(): This method returns the day of the date. The day should be a value between 1 and 31 and is stored as type int.

> Date d = new Date(3, 2)

> d.getDay()

3

1. int getMonth(): This method returns the month of the date. The month should be between 1 and 12 and is stored as type int.

> Date d = new Date(3, 2)

> d.getMonth()

2

1. void incrementDay(): This method adds 1 to the day of the date. If the day exceeds the number of days for the month, the day is set to 1 and the month is incremented. (Note that we will not be keeping track of years and so you should ignore leap years.) If the month exceeds 12, the month is set to 1 and the year is incremented. The method returns no value.

> Date d = new Date(3, 2)

> d.incrementDay()

> d.getDay()

4

> d.getMonth()

2

1. int monthsUntil(Date date): This method returns the number of months until the input Date, rounded up as type int. A full month is considered to be from the day or one month to the identical day of the next month. For example, if this date is April 10, the number of months until July 12 is 4, the number of months until September 10 is 5, and the number of months until February 2 is 10. Note that the last days of each month must be handled specially so the number of months from February 28 to March 31 is 1, the number of months from February 28 to March 29 is 1, but the number of months from February 27 to March 28 is 2.

> Date d = new Date(2, 2)

> d.monthsUntil(new Date(4,4))

3

> Date d = new Date(5, 2)

> d.monthsUntil(new Date(4,4))

2

> Date d = new Date(5, 2)

> d.monthsUntil(new Date(4,1))

11

Customer Class

**Constructor**

1. The first constructor sets the last name as type string, the address as type string.

> Customer c = new Customer("Alspaw", "Address")

> c.getLastName()

"Alspaw"

> c.getAddress()

"Address"

1. One had input String firstName, String lastName, String address, Date date and sets the appropriate values.

> Customer c = new Customer("Jacob", "Alspaw", "Address", new Date(5,5))

> c.getFirstName()

"Jacob"

> c.getLastName()

"Alspaw"

> c.getAddress()

"Address"

> c.getDate()

Date@193bcf

**Methods**

1. void setFirstName(String firstName): This method sets the first name associated with the customer as type string and returns no value.

> Customer c = new Customer("Jacob", "Alspaw", "Address", new Date(5,5))

> c.setFirstName("Changed")

> c.getFirstName()

"Changed"

1. String getFirstName(): This method returns the first name associated with the customer as type string.

> Customer c = new Customer("Jacob", "Alspaw", "Address", new Date(5,5))

> c.setFirstName("Changed")

> c.getFirstName()

"Changed"

1. void setLastName(String lastName): This method sets the last name associated with the customer as type string and returns no value.

> Customer c = new Customer("Jacob", "Alspaw", "Address", new Date(5,5))

> c.setLastName("Changed")

> c.getLastName()

"Changed"

1. String getLastName(): This method returns the last name associated with the customer as type string.

> Customer c = new Customer("Jacob", "Alspaw", "Address", new Date(5,5))

> c.setLastName("Changed")

> c.getLastName()

"Changed"

1. void setAddress(String address): This method sets the address associated with the customer as type string and returns no value.

> Customer c = new Customer("Jacob", "Alspaw", "Address", new Date(5,5))

> c.setAddress("Changed")

> c.getAddress()

"Changed"

1. String getAddress(): This method returns the address associated with the customer as type string.

> Customer c = new Customer("Jacob", "Alspaw", "Address", new Date(5,5))

> c.setAddress("Changed")

> c.getAddress()

"Changed"

1. void setElectricAccount(Account electricAccount): This method assigns an account for electricity usage for this customer as type account and returns no value.

> c.setElectricAccount(new Account(1, 2, 3, 4, 5))

> c.getElectricAccount()

Account@1ea760c

1. Account getElectricAccount(): This method returns the account for electicity usage by this customer as account type.

> c.setElectricAccount(new Account(1, 2, 3, 4, 5))

> c.getElectricAccount()

Account@1ea760c

1. void setGasAccount(Account gasAccount): This method assigns an account for natural gas usage for this customer as account type and returns no value.

> c.setGasAccount(new Account(1, 2, 3, 4, 5))

> c.getGasAccount()

Account@292acf

1. Account getGasAccount(): This method returns the account for natural gas usage by this customer as account type.

> c.setGasAccount(new Account(1, 2, 3, 4, 5))

> c.getGasAccount()

Account@292acf

1. void setWaterAccount(Account waterAccount): This method assigns an account for water usage for this customer as account type and returns no value.

> c.setWaterAccount(new Account(1, 2, 3, 4, 5))

> c.getWaterAccount()

Account@138c00d

1. Account getWaterAccount(): This method returns the account for water usage by this customer as type account.

> c.setWaterAccount(new Account(1, 2, 3, 4, 5))

> c.getWaterAccount()

Account@138c00d

1. void setDate(Date date): This method sets a Date instance associated with this customer as type date and returns no value.

> Customer c = new Customer("Jacob", "Alspaw", "Address", new Date(5,5))

> c.setDate(new Date(6, 6))

> c.getDate()

Date@d496be

1. Date getDate(): This method returns a Date instance associated with this customer as type date.

> Customer c = new Customer("Jacob", "Alspaw", "Address", new Date(5, 5))

> c.setDate(new Date(6, 6))

> c.getDate()

Date@d496be

1. void incrementDate(): This method calls the associated method of the Date class to increment the date. Calls the processDate method of each account that exists for the customer. This method returns no value.

> Customer c = new Customer("Jacob", "Alspaw", "Address", new Date(5, 5))

> c.setElectricAccount(new Account(1, 2, 3, 4, 5))

> c.setWaterAccount(new Account(1, 2, 3, 4, 5))

> c.setGasAccount(new Account(1, 2, 3, 4, 5))

> c.incrementDate()