CS213 Fall 2020 **Dr. Lily Chang**

Project #2 (60 points)

Due Date

• Thursday, October 8, by 11:59pm.

Submission

1. Group Submission (one copy per team)

(a) You must designate a submitter (one of the team members) and submit the zipped project folder to

Canvas.

(b) You must include both team members' names in the comment block on top of EVERY Java file.

(c) Your project folder must include the following subfolders/files for grading.

• Source folder, including all Java files [30 points]

o JUnit Test for the Checking class, MoneyMarket class, and Date class. [15 points]

• Javadoc folder, including all the files generated. [5 points]

• Class Diagram, drawn with either Visual Paradigm or Draw.io. [5 points]

2. Individual Submission (everyone must submit a copy)

• Personal time log, using the template posted on Canvas. [5 points]

Project Description

In this project, you will develop a Transaction Management System to process banking transactions. You

CANNOT use any Java library classes, EXCEPT the Scanner class, StringTokenizer class, and DecimalFormat class,

or 0 points!

The banking transactions will be processed through the standard I/O (console input and output.) The transactions may

include open a new account, close an existing account, deposit funds to an existing account, withdraw funds from an existing account, print the list of accounts and print the account statements. The system maintains an account database,

which may include 3 different types of bank accounts listed in the table below. Same person can open different types

of accounts. The interest rates and fee schedules are different based on the account types and account options.

Account Type Monthly Fee / waived if

balance is >=

Annual

Interest Rate Account Options

Checking $25 / $1500 0.05% Fee is waived for an account with direct deposit

Savings $5 / $300 0.25% loyal customer gets promotional interest rate 0.35%

Money Market $12 / $2500 0.65% Fee cannot be waived if the number of withdrawals

per statement exceeds 6

Requirements

1. This is a group assignment. You MUST work in pair in order to get the credit for this program.

2. You MUST follow the software development ground rules, or you will lose points for not having a good

programming style.

3. You are required to log your times working on this project with the template provided on Canvas. The time log

is an individual assignment. You will lose 5 points if the log is not submitted. If the times and comments are

not properly logged, you will only get partial credits. You must type, handwriting is not acceptable.

4. You MUST create a Class Diagram for this project to document the software structure. The diagram is worth 5

points. Hand-drawing diagram is NOT acceptable!

5. Each Java class must go in a separate file. -2 points if you put more than one Java class into a file.

6. Your program MUST handle bad commands! Commands are case-sensitive. -2 points for each violation.

7. You must include the classes listed below. -5 points for each class missing. Override the toString() and equals()

method where appropriate. The toString() methods in the subclasses must reuse the code in the superclass by

calling the toString() method in the superclass. -1 point for each violation. All the overriding methods must

include the annotation @Override on top of the methods. -1 points for each annotation missing. You CANNOT

add additional instance variables. -2 points for each violation. You can add constants and other methods if

necessary. You CANNOT change the signatures of the methods listed, -2 points for each violation. You

CANNOT do I/O in all classes, EXCEPT the TransactionManager class, and the print methods in the

AccountDatabase class. -2 points for each violation. The floating-point numbers must be displayed with 2

decimal places. -1 point for each violation.

(a) Account class. This is an abstract class that defines the common features of all account types.

```
public abstract class Account {
 private Profile holder;
 private double balance;
private Date dateOpen;
 public void debit(double amount) { } //decrease the balance by amount
 public void credit(double amount) { } //increase the balance by amount
 public String toString() { }
public abstract double monthlyInterest { }
public abstract double monthlyFee() { }
}
```

(b) Profile class. This class defines the profile of an account holder.

```
public class Profile {
 private String fname;
 private String lname;
}
```

(c) Date class. You CANNOT use the Date class in the Java API library, or -10 points. You must implement

the Date class yourself. This class implements the Java Interface Comparable.

```
public class Date implements Comparable<Date> {
 private int year;
 private int month;
private int day;
 public int compareTo(Date date) { } //return 0, 1, or -1
 public String toString() { } //in the format mm/dd/yyyy
public boolean isValid() { }
}
```

(d) AccountDatabase class. This is an array-based container class with an initial capacity of 5. It will

automatically grow the capacity by 5 if the database is full. The array shall hold different account instances

in Checking, Savings or MoneyMarket.

```
public class AccountDatabase {
 private Account[] accounts;
 private int size;
 private int find(Account account) { }
 private void grow() { }
 public boolean add(Account account) { } //return false if account exists
```

public boolean remove(Account account) { } //return false if account doesn’t exist

public boolean deposit(Account account, double amount) { }

//return 0: withdrawal successful, 1: insufficient funds, -1 account doesn’t exist

public int withdrawal(Account account, double amount) { }

private void sortByDateOpen() { } //sort in ascending order

private void sortByLastName() { } //sort in ascending order

public void printByDateOpen() { }

public void printByLastName() { }

public void printAccounts() { }

}

(e) Checking class.

public class Checking extends Account {

private boolean directDeposit;

}

(f) Savings class.

public class Savings extends Account {

private boolean isLoyal;

}

(g) MoneyMarket class.

public class MoneyMarket extends Account {

private int withdrawals;

}

(h) TransactionManager class. This is the user interface class that handles the transactions and displays the

results on the console. You can define the data fields and private methods you need.

Each transaction begins with a two-letter command identifying the transaction type and account type, followed by the data tokens and ends with a new line character (\n). The two-letter commands are casesensitive, which means the commands with lowercase letters are invalid! In addition, you are required to

handle the bad commands that are not supported. -2 points for each invalid command not handled. You

MUST try-catch all the exceptions from invalid data tokens or handle the mismatch data types, such as

InputMismatchException or NumberFormatException. -2 points for each exception not caught. Below

are examples of valid commands.

• O commands – open a new checking (C), savings (S) or money market (M) account; for example,

OC John Doe 300 false //open a checking account with $300, non-direct deposit

OS John Doe 500.5 true //open a savings account with $500.50, loyal customer

OM John Doe 1234.567 //open a money market account with $1,234.57

• C commands – close an existing account; for example,

CC John Doe //close a checking account associated with the name

CS John Doe //close a savings account associated with the name

CM John Doe //close a money market account associated with the name

• D commands – deposit funds to an existing account; for example,

DC John Doe 500 //deposit $500 to a checking account associated with the name

DS John Doe 500 //deposit $500 to a savings account associated with the name

DM John Doe 500 //deposit $500 to a money market account associated with the name

• W commands – withdraw funds from an existing account; for example,

WC John Doe 500 //withdraw $500 from a checking account associated with the name

WS John Doe 500 //withdraw $500 from a savings account associated with the name

WM John Doe 500 //withdraw $500 from a money market account associated with the name

• P commands – print the list of accounts or print account statements; for example,

PA //print the list of accounts in the database

PD //calculate the monthly interests and fees, and print the account statements,

//sorted by the dates opened in ascending order

PN //same with PD, but sorted by the last names in ascending order

• Q command – display “Transaction processing completed.” and stop the program execution.

(i) RunProject2 class. This is the driver class to run Project 2

```
public class RunProject2 {

 public static void main(String[] args) {

 new TransactionManager().run();

 }

}
```

8. You must follow the instructions in the Software Development Ground Rules and comment your code. You are

required to generate the Javadoc after you properly commented your code. Your Javadoc must include the

documentations for the constructors, private methods and public methods of all Java classes. Generate the

Javadoc in a single folder and include it in your project folder to be submitted to Canvas. You will lose 5 points

for not including the Javadoc.

9. You must create JUnit test classes for Checking and MoneyMarket classes to test the monthlyInterest() and

monthlyFee() method. You must create a JUnit test class for the Date class to test the isValid() method. Each

JUnit test class is worth 5 points.

Sample Input

```
o

p

c

OA John Doe 500 1/1/2010 true
```

```
cc John Doe 500 1/1/2010 true
cs John Doe 500 1/1/2010 true
cm John Doe 500 1/1/2010 true
oc John Doe 500 1/1/2010 true
os John Doe 500 1/1/2010 true
om John Doe 500 1/1/2010 true
CA John Doe 500 1/1/2010 true
cc John Doe 500 1/1/2010 true
cs John Doe 500 1/1/2010 true
cm John Doe 500 1/1/2010 true
pn
pd
pa
PA
PN
PD
CC John Doe
CS John Doe
CM John Doe
OC John Doe 500 1/1/2010 true
OS John Doe 500 1/1/2010 true
OM John Doe 500 1/1/2010
OC John Doe 500 1/1/2010 true
OS John Doe 500 1/1/2010 true
OM John Doe 500 1/1/2010
PA
WC John Doe 100
```

WS John Doe 100

WM John Doe 100

PA

DC John Doe 100

DS John Doe 100

DM John Doe 100

PA

WC JOHN DOE 100

DS JANE DOE 100

OC Jane Doe 500 13/1/2010 false

OC Jane Doe 500 12/32/2010 false

OC Jane Doe 500 2/29/2010 false

OC Jane Doe 500 2/29/2008 false

OS Jane Doe 500 11/31/2019 false

OS Jane Doe 500 11/1/2019 false

PA

CC John Doe

CM John Doe

PA

OC Richard Scanlan 1500 7/31/2014 true

5 | Page

OC Jason Brown 300.123 6/28/2015 false

OS Mary Johnson 2000.87654 7/31/2014 true

OS Charlie Brown 800 6/28/2015 false

OM Mary Johnson 456.78 7/10/2014

OM Charlie Brown 600.33 6/2/2015

OM Forever Young 3634.45 2/21/2020

OC Jerry Anderson 1001.4 9/23/2020 false

OS Tom Moore xxx 9/23/2020 false

OS Tom Moore 1001.4 9/23/2020 FALS

E

OC Tom Moore 1001.4 9/23/2020 flash

PA

WC Jerry Anderson xxx

DC Jerry Anderson xxx

WM Forever Young 100

WM Forever Young 100

WM Forever Young 100

WM Forever Young 100

WM Forever Young 100

WM Forever Young 100

WM Forever Young 100

WM Forever Young 10000

PA

PN

WS Charlie Brown 600

PDQ

Sample Output

Transaction processing starts.....

Command 'o' not supported!

Command 'p' not supported!

Command 'c' not supported!

Command 'OA' not supported!

Command 'cc' not supported!

```
Command 'cs' not supported!

Command 'cm' not supported!

Command 'oc' not supported!

Command 'os' not supported!

Command 'om' not supported!

Command 'CA' not supported!

Command 'cc' not supported!

Command 'cs' not supported!

Command 'cm' not supported!

Command 'pn' not supported!

Command 'pd' not supported!

Command 'pa' not supported!

Database is empty.

Database is empty.

Database is empty.

Account does not exist.

Account does not exist.

Account does not exist.

Account opened and added to the database.

Account opened and added to the database.

Account opened and added to the database.

Account is already in the database.

Account is already in the database.

Account is already in the database.

--Listing accounts in the database--

*Checking*John Doe* $500.00*1/1/2010*direct deposit account*
```

```
*Savings*John Doe* $500.00*1/1/2010*special Savings account*
*Money Market*John Doe* $500.00*1/1/2010*0 withdrawals*
--end of listing--
100.00 withdrawn from account.
100.00 withdrawn from account.
100.00 withdrawn from account.
--Listing accounts in the database--
*Checking*John Doe* $400.00*1/1/2010*direct deposit account*
*Savings*John Doe* $400.00*1/1/2010*special Savings account*
*Money Market*John Doe* $400.00*1/1/2010*1 withdrawal*
--end of listing--
100.0 deposited to account.
100.0 deposited to account.
100.0 deposited to account.
--Listing accounts in the database--
*Checking*John Doe* $500.00*1/1/2010*direct deposit account*
*Savings*John Doe* $500.00*1/1/2010*special Savings account*
*Money Market*John Doe* $500.00*1/1/2010*1 withdrawal*
--end of listing--
Account does not exist
Account does not exist.
13/1/2010 is not a valid date!
12/32/2010 is not a valid date!
2/29/2010 is not a valid date!
Account opened and added to the database.
11/31/2019 is not a valid date!
Account opened and added to the database.
```

```
--Listing accounts in the database--
*Checking*John Doe* $500.00*1/1/2010*direct deposit account*
*Savings*John Doe* $500.00*1/1/2010*special Savings account*
*Money Market*John Doe* $500.00*1/1/2010*1 withdrawal*
*Checking*Jane Doe* $500.00*2/29/2008
*Savings*Jane Doe* $500.00*11/1/2019
--end of listing--
Account closed and removed from the database.
Account closed and removed from the database.
--Listing accounts in the database--
*Savings*Jane Doe* $500.00*11/1/2019
*Savings*John Doe* $500.00*1/1/2010*special Savings account*
*Checking*Jane Doe* $500.00*2/29/2008
--end of listing--
Account opened and added to the database.
Account opened and added to the database.
Account opened and added to the database.
Account opened and added to the database.
Account opened and added to the database.
Account opened and added to the database.
Account opened and added to the database.
Account opened and added to the database.
Input data type mismatch.
Account opened and added to the database.
Input data type mismatch.
--Listing accounts in the database--
*Savings*Jane Doe* $500.00*11/1/2019
```

```
*Savings*John Doe* $500.00*1/1/2010*special Savings account*
*Checking*Jane Doe* $500.00*2/29/2008
*Checking*Richard Scanlan* $1,500.00*7/31/2014*direct deposit account*
7 | Page
*Checking*Jason Brown* $300.12*6/28/2015
*Savings*Mary Johnson* $2,000.88*7/31/2014*special Savings account*
*Savings*Charlie Brown* $800.00*6/28/2015
*Money Market*Mary Johnson* $456.78*7/10/2014*0 withdrawals*
*Money Market*Charlie Brown* $600.33*6/2/2015*0 withdrawals*
*Money Market*Forever Young* $3,634.45*2/21/2020*0 withdrawals*
*Checking*Jerry Anderson* $1,001.40*9/23/2020
*Savings*Tom Moore* $1,001.40*9/23/2020
--end of listing--
Input data type mismatch.
Input data type mismatch.
100.00 withdrawn from account.
100.00 withdrawn from account.
100.00 withdrawn from account.
100.00 withdrawn from account.
100.00 withdrawn from account.
100.00 withdrawn from account.
100.00 withdrawn from account.
Insufficient funds.
--Listing accounts in the database--
*Savings*Jane Doe* $500.00*11/1/2019
*Savings*John Doe* $500.00*1/1/2010*special Savings account*
*Checking*Jane Doe* $500.00*2/29/2008
```

```
*Checking*Richard Scanlan* $1,500.00*7/31/2014*direct deposit account*
*Checking*Jason Brown* $300.12*6/28/2015
*Savings*Mary Johnson* $2,000.88*7/31/2014*special Savings account*
*Savings*Charlie Brown* $800.00*6/28/2015
*Money Market*Mary Johnson* $456.78*7/10/2014*0 withdrawals*
*Money Market*Charlie Brown* $600.33*6/2/2015*0 withdrawals*
*Money Market*Forever Young* $2,934.45*2/21/2020*7 withdrawals*
*Checking*Jerry Anderson* $1,001.40*9/23/2020
*Savings*Tom Moore* $1,001.40*9/23/2020
--end of listing--
--Printing statements by last name--
*Checking*Jerry Anderson* $1,001.40*9/23/2020
-interest: $ 0.04
-fee: $ 25.00
-new balance: $ 976.44
*Checking*Jason Brown* $300.12*6/28/2015
-interest: $ 0.01
-fee: $ 25.00
-new balance: $ 275.14
*Savings*Charlie Brown* $800.00*6/28/2015
-interest: $ 0.17
-fee: $ 0.00
-new balance: $ 800.17
*Money Market*Charlie Brown* $600.33*6/2/2015*0 withdrawals*
-interest: $ 0.33
-fee: $ 12.00
-new balance: $ 588.66
```

```
*Savings*John Doe* $500.00*1/1/2010*special Savings account*
```

```
-interest: $ 0.15
-fee: $ 0.00
-new balance: $ 500.15
*Checking*Jane Doe* $500.00*2/29/2008
-interest: $ 0.02
-fee: $ 25.00
-new balance: $ 475.02
*Savings*Jane Doe* $500.00*11/1/2019
-interest: $ 0.10
-fee: $ 0.00
-new balance: $ 500.10
*Money Market*Mary Johnson* $456.78*7/10/2014*0 withdrawals*
-interest: $ 0.25
-fee: $ 12.00
-new balance: $ 445.03
*Savings*Mary Johnson* $2,000.88*7/31/2014*special Savings account*
-interest: $ 0.58
-fee: $ 0.00
-new balance: $ 2,001.46
*Savings*Tom Moore* $1,001.40*9/23/2020
-interest: $ 0.21
-fee: $ 0.00
-new balance: $ 1,001.61
*Checking*Richard Scanlan* $1,500.00*7/31/2014*direct deposit account*
-interest: $ 0.06
```

```
-fee: $ 0.00
-new balance: $ 1,500.06
*Money Market*Forever Young* $2,934.45*2/21/2020*7 withdrawals*
-interest: $ 1.59
-fee: $ 12.00
-new balance: $ 2,924.04
--end of printing--
600.00 withdrawn from account.
--Printing statements by date opened--
*Checking*Jane Doe* $475.02*2/29/2008
-interest: $ 0.02
-fee: $ 25.00
-new balance: $ 450.04
*Savings*John Doe* $500.15*1/1/2010*special Savings account*
-interest: $ 0.15
-fee: $ 0.00
-new balance: $ 500.29
*Money Market*Mary Johnson* $445.03*7/10/2014*0 withdrawals*
-interest: $ 0.24
-fee: $ 12.00
-new balance: $ 433.27
```

```
*Savings*Mary Johnson* $2,001.46*7/31/2014*special Savings account*
-interest: $ 0.58
-fee: $ 0.00
-new balance: $ 2,002.04
*Checking*Richard Scanlan* $1,500.06*7/31/2014*direct deposit account*
```

```
-interest: $ 0.06
-fee: $ 0.00
-new balance: $ 1,500.13
*Money Market*Charlie Brown* $588.66*6/2/2015*0 withdrawals*
-interest: $ 0.32
-fee: $ 12.00
-new balance: $ 576.97
*Savings*Charlie Brown* $200.17*6/28/2015
-interest: $ 0.04
-fee: $ 5.00
-new balance: $ 195.21
*Checking*Jason Brown* $275.14*6/28/2015
-interest: $ 0.01
-fee: $ 25.00
-new balance: $ 250.15
*Savings*Jane Doe* $500.10*11/1/2019
-interest: $ 0.10
-fee: $ 0.00
-new balance: $ 500.21
*Money Market*Forever Young* $2,924.04*2/21/2020*7 withdrawals*
-interest: $ 1.58
-fee: $ 12.00
-new balance: $ 2,913.62
*Checking*Jerry Anderson* $976.44*9/23/2020
-interest: $ 0.04
-fee: $ 25.00
-new balance: $ 951.48
```

```
*Savings*Tom Moore* $1,001.61*9/23/2020

-interest: $ 0.21

-fee: $ 0.00

-new balance: $ 1,001.82

--end of printing--

Transaction processing completed.
```