CS1083
Assignment #5

Daniyal Khan
3765942

Driver.java:

```
import java.util.Scanner;
import java.text.NumberFormat;
import java.util.ArrayList;

public class Driver {

    public static void main(String[] args) {
        NumberFormat formatter =
NumberFormat.getCurrencyInstance();
        Scanner scan = new Scanner(System.in);

        ArrayList<Account> accounts = new ArrayList<>();
        boolean accountExists = false;
        int numberOfAccounts = -1;
        int userInput = -1;

        do {
            if(numberOfAccounts >= 0) {
                System.out.println("\nCurrently working with
account: " +  numberOfAccounts + " Balance of " +
formatter.format(accounts.get(numberOfAccounts).getBalance()));
            }
            System.out.println(userInstructions(accountExists));
            try {
                userInput = Integer.parseInt(scan.nextLine());
                if (userInput == 1) {
                    Account chequingAccount = new
ChequingAccount(0);
                    accounts.add(chequingAccount);
                    accountExists = true;
                    numberOfAccounts++;
                } else if (userInput == 2) {
                    try{
                        System.out.println("Input a non-negative
interest amount:");
                        double interestRate =
Double.parseDouble(scan.nextLine());
                        Account savingsAccount = new
SavingsAccount(0, interestRate);
                        accounts.add(savingsAccount);
                        numberOfAccounts++;
                    } catch (NegativeException ne) {
                        System.out.println(ne.getMessage());
                    }
                    accountExists = true;
                } else if (userInput == 3 && accountExists) {
                    try {
                        System.out.println("Input an amount to 
deposit: ");
                        double amount = 
Double.parseDouble(scan.nextLine());

accounts.get(numberOfAccounts).depositMoney(amount);
                    } catch (NegativeException ne) {
                        System.out.println(ne.getMessage());
                    }
                } else if (userInput == 4 && accountExists) {
                    try {
                        System.out.println("Input an amount to 
withdraw");
                        double amount = 
Double.parseDouble(scan.nextLine());

accounts.get(numberOfAccounts).withdrawMoney(amount);
                    } catch (InsufficientFundsException | 
NegativeException e) {
                        System.out.println(e.getMessage());
                    }
                } else if (userInput == 5 && accountExists) {
                    try {

accounts.get(numberOfAccounts).applyInterest();
                    } catch (NegativeException ne) {
                        System.out.println(ne.getMessage());
                    }
                } else if (userInput == 6 && accountExists) {
                    System.out.println("Which account would you 
like to switch to?");
                    int switchToAccNum = 
Integer.parseInt(scan.nextLine());
                    if (switchToAccNum >= 0 && switchToAccNum <= 
accounts.size() - 1) {
                        numberOfAccounts = switchToAccNum;
                    } else {
                        System.out.print("Account does not
exist, please try again");
                    }
                }
            } catch (NumberFormatException nfe) { // if user
doesnt enter the correct data type
                System.out.println();
            }
            System.out.println();
        } while(userInput != 0);
        scan.close();
    }

    public static String userInstructions(boolean accountMade) {
// user prompts
        String toReturn = "";
        if (!accountMade) {
            toReturn += "NO ACCOUNTS MADE. PLEASE CREATE AN
ACCOUNT.\n";
        }
        toReturn += "Please input a command:\n" +
                    "1: Create a new Chequing Account\n" +
                    "2: Create a new Savings Account\n" +
                    "3: Deposit Funds\n" +
                    "4: Withdraw Funds\n" +
                    "5: Apply Interest\n" +
                    "6: Switch to a Different Account\n" +
                    "0: To Exit\n";
        return toReturn;
    }
}
```

Bank.java:

```
public interface Bank {
    public void applyInterest() throws NegativeException;
    public void withdrawMoney(double amount) throws
InsufficientFundsException, NegativeException;
    public void depositMoney(double amount) throws
NegativeException;
}
```

Account.java:

```
import java.text.NumberFormat;

public abstract class Account implements Bank {
    private double balance;
    private NumberFormat formatter;

    public Account (double startingBalance) {
        this.balance = startingBalance;
        formatter = NumberFormat.getCurrencyInstance();
    }

    public double getBalance() {
        return balance;
    }

    public void setBalance(double balanceAmount) {
        this.balance = balanceAmount;
    }



    public void withdrawMoney(double amount) throws
InsufficientFundsException, NegativeException {
        if (amount < 0) {
            throw new NegativeException("Withdraw amounts must
be positive.");
        }
        if (balance >= amount) {
            balance -= amount;
        } else {
            throw new InsufficientFundsException("You are trying
to withdraw " + formatter.format(amount-balance) + " more than
you have in your account");
        }
    }

    public void depositMoney(double amount) throws
NegativeException {
        if (amount < 0) {
            throw new NegativeException("Deposit amounts must be 
positive.");
        }
        balance += amount;
    }
}
```

SavingsAccount.java:

```
public class SavingsAccount extends Account{
    private double interestRate;

    public SavingsAccount(double startingBalance, double 
interestRate) throws NegativeException {
        super(startingBalance);
        if (interestRate < 0) {
            throw new NegativeException("Interest rates may not 
be negative.");
        }
        this.interestRate = interestRate;
    }

    public void applyInterest() throws NegativeException {
        setBalance(super.getBalance()*(interestRate+1));
    }
}
```

ChequingAccount.java:

```
public class ChequingAccount extends Account{
    private final double INTEREST_RATE = 0.005;

    public ChequingAccount(double startingBalance) {
        super(startingBalance);
    }

    public void applyInterest() throws NegativeException {
        setBalance(super.getBalance()*(INTEREST_RATE+1));
    }
}
```

NegativeException.java:

```
public class NegativeException extends Exception{

    public NegativeException(String message) {
        super(message);
    }
}
```

InsufficientFundsException.java:

```
public class InsufficientFundsException extends Exception {

    public InsufficientFundsException(String message) {
        super(message);
    }
}
```