Programming Coursework

PART II: Bank Account Management

Joe Halloran

Sample of test output

```
20230715    Donald Trump    £450.00
31558040    Bill Gates    £100.00
20230715    Donald Trump    £525.00

20230715    Donald Trump    £525.00
31558040    Bill Gates    £100.00
44003050    Tom Cruise    £600.00

Total deposits:        £1,225.00

Tax paid by Donald Trump:        £78.75
Tax paid by Bill Gates:        £15.00
Tax paid by Tom Cruise:        £90.00

Interest paid to Donald Trump:        £6.69
Interest paid to Bill Gates:        £1.27
Interest paid to Tom Cruise:        £7.65
Interest paid to Inland Revenue:        £2.76

Trump creation date:        Tue Jun 06 16:10:50 BST 2017
Gate creation date:        Tue Jun 06 16:10:50 BST 2017
Cruise creation date:        Tue Jun 06 16:10:50 BST 2017

Process finished with exit code 0
```

Appendix: Source code listing

ManageAccount.java

```java
import java.text.DecimalFormat;
import java.text.NumberFormat;

/**
 * ManageAccount class manages 4 accounts (trump, gate, cruise, inlandRevenue)
 * based on instructions from Task 2: Bank Account Management
 */
public class ManageAccount {
    // Initialise formators used for cash values
    DecimalFormat decimalFormat = new DecimalFormat("#.##");
    NumberFormat stringFormat = NumberFormat.getCurrencyInstance();

    // Initialise accouts                                       // Task 1.i
    Account trump = new Account("Donald Trump", 20230715, 400);
    Account gates = new Account("Bill Gates", 31558040, 500);
    Account cruise = new Account("Tom Cruise", 44003050, 600);
    Account inlandRevenue = new Account("Inland Revenue", 11223344); // Task 6 - no initial deposit


    public static void main(String[] args) {
        // create accounts
        ManageAccount accounts = new ManageAccount();
    }

    /**
     * Executes of tasks
     */
    public ManageAccount() {
        trump.deposit(50);                      // Task 1.ii
        System.out.println (trump.toString());

        gates.withdraw(400,0);              // Task 1.iii
        System.out.println (gates.toString());

        trump.deposit(75);                      // Task 1.iv
        System.out.println (trump.toString());

        System.out.println();                      // Task 1.v
        System.out.println (trump.toString());
        System.out.println (gates.toString());
        System.out.println (cruise.toString());

        System.out.println(" ");                    // Task 2 - total deposits
        System.out.println (getTotalDeposits());

        System.out.println();                       // Task 7 - pay taxes
        deductTax(trump, inlandRevenue);
        deductTax(gates, inlandRevenue);
        deductTax(cruise, inlandRevenue);

        System.out.println();                       // Task 8 - add interest
        double interestRate = 0.015;
        addInterest(trump, interestRate);
        addInterest(gates, interestRate);
        addInterest(cruise, interestRate);
        addInterest(inlandRevenue, interestRate);

        System.out.println();                       // Task 9 - creation date
        System.out.println("Trump creation date:\t\t" + trump.getCreationDate());
        System.out.println("Gate creation date:\t\t" + gates.getCreationDate());
        System.out.println("Cruise creation date:\t\t" + cruise.getCreationDate());

    }
    //-----------------------------------------------------------------
    //  gets total deposits of trump, gates, and cruise accounts combined
    //  Task 2
    //-----------------------------------------------------------------
    public String getTotalDeposits() {
        double total = trump.getBalance() +  gates.getBalance() + cruise.getBalance();
        return "Total deposits:\t\t" + toString(total);
    }

    //-----------------------------------------------------------------
    //  Caculates the tax due for a given account
    //  Task 5
    //-----------------------------------------------------------------
    public double calculateTax(Account account) {
        double tax = calculatePercentage(account.getBalance(), 0.15);
        return tax;
    }

    //-----------------------------------------------------------------
    //  Withdraws tax from citizen and pays to taxMan.
    //  Uses a try - catch statement (with resets) to eliminate the possibilty
    //  that the withdrawal from citizen may complete, but deposit in taxMan fails.
    //  Task 5
    //-----------------------------------------------------------------
    public void deductTax(Account citizen, Account taxMan) {
        double citizenReset = citizen.getBalance();
        double taxManReset = taxMan.getBalance();
        try {
            double tax = calculateTax(citizen);
            citizen.withdraw(tax);
            taxMan.deposit(tax);
            System.out.println("Tax paid by " + citizen.name + ":\t\t" + toString(tax));
        } catch (Exception e) {
            System.out.println("Tax payment could not be processed");
            citizen.setBalance(citizenReset);
            taxMan.setBalance(taxManReset);
        }


    }

    //-----------------------------------------------------------------
    //  Adds interest to account
    //  Task 8
    //-----------------------------------------------------------------
    public void addInterest(Account account, double interestRate) {
        double interest = calculatePercentage(account.getBalance(), interestRate);
        account.deposit(interest);
        System.out.println("Interest paid to " + account.name + ":\t\t" + toString(interest));
    }

    //-----------------------------------------------------------------
    //  Utility function to handle correct rounding to 2 decimal places
    //  when calculating the percentage of a cash value
    //  Used in Task 5 (tax) and Task 8 (interest)
    //-----------------------------------------------------------------
    private double calculatePercentage(double value, double percentage) {
        double output = Double.valueOf(decimalFormat.format(value * percentage));
        return output;
    }

    //-----------------------------------------------------------------
    //  Utility function to correctly format money for console printing
    //-----------------------------------------------------------------
    private String toString (double value) {
        return stringFormat.format(value);
    }

}
```

Account.java

```java
//********************************************************************
//  Account.java       Author: Lewis/Loftus
//
//
//  Represents a bank account with methods deposit and withdraw.
//********************************************************************


import java.text.NumberFormat;
import java.util.Date;

public class Account
{
    int acctNumber;
    double balance;
    String name;
    Date creationDate;
    double overdraftLimit;      // Task 10

    //-----------------------------------------------------------------
    //  Sets up the account by defining its owner's name and account
    //  number only.
    //  Task 6
    //-----------------------------------------------------------------
    public Account (String x, int y)
    {
        name = x;
        acctNumber = y;
        balance = 0;
        creationDate = new Date();      // Task 9
        overdraftLimit = 100;           // Task 10
    }


    //-----------------------------------------------------------------
    //  Sets up the account by defining its owner's name, account
    //  number, and initial balance.
    //-----------------------------------------------------------------
    public Account (String x, int y, double z)
    {
        name = x;
        acctNumber = y;
        balance = z;
        creationDate = new Date();      // Task 9
        overdraftLimit = 0;             // Task 10
    }

    //-----------------------------------------------------------------
    //  Deposits the specified amount x into the account.
    //-----------------------------------------------------------------
    public void deposit (double x)
    {
        balance = balance + x;
    }

    //-----------------------------------------------------------------
    //  Withdraws the specified amount from the account for no fee.
    //
    //-----------------------------------------------------------------
    public void withdraw (double x)
    {
        withdraw(x, 0);
    }           // Task 4

    //-----------------------------------------------------------------
    //  Withdraws the specified amount from the account and applies
    //  the fee.
    //-----------------------------------------------------------------
    public void withdraw (double x, double fee)
    {
        if (balance + overdraftLimit > (x + fee) ){                  // Task 10
            balance = balance - x - fee;
        } else {
            System.out.println("You have insufficient funds to make this withdrawal"); // Task 3
        }
    }


    //-----------------------------------------------------------------
    //  Returns the current balance of the account.
    //-----------------------------------------------------------------
    public double getBalance ()
    {
        return balance;
    }

    //-----------------------------------------------------------------
    //  Set balance to a specified value.
    //  An additional method to restore balance to a cached value,
    //  in case of incomplete transaction.
    //-----------------------------------------------------------------
    public void setBalance (double value) {
        balance = value;
    }

    //-----------------------------------------------------------------
    //  Returns the creation date of the account                        // Task 9
    //-----------------------------------------------------------------
    public Date getCreationDate ()
    {
        return creationDate;
    }

    //-----------------------------------------------------------------
    //  Returns a one-line description of the account as a string.
    //-----------------------------------------------------------------
    public String toString ()
    {
        NumberFormat fmt = NumberFormat.getCurrencyInstance();
        return (acctNumber + "\t" + name + "\t" + fmt.format(balance));
    }
}
```