

# Midterm Programming Question: *ArrayList* Style

```
public class TestBank {
    public static void main(String[] args) {
        int nMaxNumberOfAccounts = Input.getInt("Number of Accounts:", 1, 1000);
        Bank badBank = new Bank(nMaxNumberOfAccounts);

        int menuChoice;
        do {
            System.out.println("\n1. Add Account");
            System.out.println("2. Perform Month End");
            System.out.println("3. Display All Accounts");
            System.out.println("0. Exit");
            menuChoice = Input.getInt("Enter Choice:", 0, 3);
            System.out.println();
            switch (menuChoice) {
                case 1: badBank.addAccount(); break;
                case 2: badBank.calculateMonthEnd(); break;
                case 3: badBank.displayAllAccounts(); break;
            } // end switch
        } while (menuChoice != 0);
    } // end void main()
} // end class TestBank
```

```
abstract public class Account {
    private String name;
    private int accountNum;
    private double balance;

    abstract public double getInterestRate();
    abstract public double getFees();

    public void set() {
        name = Input.getString("Account Holder's Name:");
        accountNum = Input.getInt("Account Number:", 10000, 99999);
        balance = Input.getDouble("Opening Balance:", 0.01, 100000.00);
    }

    public void calculateMonthEnd(){
        double interest = balance * (getInterestRate() / 12.0 / 100.0);
        balance += interest;
        balance -= getFees();
    }

    public String toString() {
        return String.format("%d: %-16s Balance: %7.2f", accountNum, name, balance);
    }
} // end class Account
```

```
public class Savings extends Account {

    @Override
    public double getInterestRate() { return 3.0; }

    @Override
    public double getFees() { return 4.50; }
} // end class Savings
```

```
public class Checking extends Account {
    private int numTransactions;

    @Override
    public double getInterestRate() { return 0.0; }

    @Override
    public double getFees() { return 6.00 + numTransactions * 0.75; }

    @Override
    public void set() {
        super.set();
        numTransactions = Input.getInt("Number of Transactions:", 0, 100);
    }

    @Override
    public void calculateMonthEnd() {
        super.calculateMonthEnd();
        numTransactions = 0; // charged customer for transaction fees, now reset to 0
    }

    @Override
    public String toString() {
        return String.format("%s Transactions:%d", super.toString(), numTransactions);
    }
} // end class Checking
```

```
import java.util.ArrayList;

public class Bank {
    private ArrayList<Account> listAccounts;

    public Bank(int initialSize) {
        listAccounts = new ArrayList<Account>(initialSize);
    }

    public void addAccount() {
        Account newAccount = null;
        switch (Input.getInt("1:Savings 2:Checking :", 1, 2)) {
            case 1: newAccount = new Savings(); break;
            case 2: newAccount = new Checking(); break;
        } // end switch
        newAccount.set();
        listAccounts.add(newAccount);
    }

    public void calculateMonthEnd() {
        for (Account acct : listAccounts)
            acct.calculateMonthEnd();
    }

    public void displayAllAccounts() {
        for (Account acct : listAccounts)
            System.out.println(acct);
    }
} // end class Bank
```

Note: This is only one possible solution.  
There are many other ways to  
implement a correct solution.

# Midterm Programming Question: *array* Style

```
public class TestBank {
    public static void main(String[] args) {
        int nMaxNumberOfAccounts = Input.getInt("Number of Accounts:", 1, 1000);
        Bank badBank = new Bank(nMaxNumberOfAccounts);

        int menuChoice;
        do {
            System.out.println("1. Add Account");
            System.out.println("2. Perform Month End");
            System.out.println("3. Display All Accounts");
            System.out.println("0. Exit");
            menuChoice = Input.getInt("Enter Choice:", 0, 3);
            System.out.println();
            switch (menuChoice) {
                case 1: badBank.addAccount(); break;
                case 2: badBank.calculateMonthEnd(); break;
                case 3: badBank.displayAllAccounts(); break;
            } // end switch
        } while (menuChoice != 0);
    } // end void main()
} // end class TestBank
```

```
abstract public class Account {
    private String name;
    private int accountNum;
    private double balance;

    abstract public double getInterestRate();
    abstract public double getFees();

    public void set() {
        name = Input.getString("Account Holder's Name:");
        accountNum = Input.getInt("Account Number:", 10000, 99999);
        balance = Input.getDouble("Opening Balance:", 0.01, 100000.00);
    }

    public void calculateMonthEnd(){
        double interest = balance * (getInterestRate() / 12.0 / 100.0);
        balance += interest;
        balance -= getFees();
    }

    public String toString() {
        return String.format("%d: %-16s Balance: %7.2f", accountNum, name, balance);
    }
} // end class Account
```

```
public class Savings extends Account {

    @Override
    public double getInterestRate() { return 3.0; }

    @Override
    public double getFees() { return 4.50; }
} // end class Savings
```

```
public class Checking extends Account {
    private int numTransactions;

    @Override
    public double getInterestRate() { return 0.0; }

    @Override
    public double getFees() { return 6.00 + numTransactions * 0.75; }

    @Override
    public void set() {
        super.set();
        numTransactions = Input.getInt("Number of Transactions:", 0, 100);
    }

    @Override
    public void calculateMonthEnd() {
        super.calculateMonthEnd();
        numTransactions = 0; // charged customer for transaction fees, now reset to 0
    }

    @Override
    public String toString() {
        return String.format("%s Transactions:%d", super.toString(), numTransactions);
    }
} // end class Checking
```

```
public class Bank {
    private Account[] listAccounts;
    private int numAccounts;

    public Bank(int capacity) {
        listAccounts = new Account[capacity];
        numAccounts = 0;
    }

    public boolean addAccount() {
        if (numAccounts >= listAccounts.length) {
            System.err.println("Array full");
            return false; // no room in array
        }

        switch (Input.getInt("1:Savings 2:Checking :", 1, 2)) {
            case 1: listAccounts[numAccounts] = new Savings(); break;
            case 2: listAccounts[numAccounts] = new Checking(); break;
        } // end switch
        listAccounts[numAccounts++].set();
        return true;
    }

    public void calculateMonthEnd() {
        for (int i=0; i<numAccounts; ++i)
            listAccounts[i].calculateMonthEnd();
    }

    public void displayAllAccounts() {
        for (int i=0; i<numAccounts; ++i)
            System.out.println(listAccounts[i]);
    }
} // end class Bank
```

Note: This is only one possible solution.  
There are many other ways to  
implement a correct solution.