

```

/*
 * Account.java
 *
 * Created on July 21, 2004, 12:16 AM
 */
import java.util.*;
abstract class Account {
    protected Customer customer;
    protected double balance;
    protected String accountNumber;
    protected Transaction[] transaction;
    protected static int accNumber = 0;

    protected static final int INIT_CAP = 20;
    protected int capacity;
    protected int tranIndex;

    /** Creates a new instance of Account */
    public Account() {
        capacity = INIT_CAP;
    }

    /**
     * Returns account balance
     * @return double Account balance
     */
    public double getBalance(){
        return this.balance;
    }

    /**
     * Returns account owner
     * @return Customer Account owner
     */
    public Customer getCustomer(){
        return this.customer;
    }

    /**
     * Returns account number
     * @return String Account number
     */
    public String getNumber(){
        return this.accountNumber;
    }

    /**
     * Changes account balance
     * pre: balance must be a positive value
     * post: account balance changes to <balance>
     * @param double balance New balance
     */
    public void setBalance(double balance){
        this.balance = balance;
    }

    /**
     * Changes account customer
     * pre: customer must be valid
     * post: account owner changes to <customer>
     * @param Customer customer New customer
     */
    public void setCustomer(Customer customer){
        this.customer = customer;
    }

```

```

    }

    /**
     * Converts Account object to a String object
     * @return String account information as String object
     */
    public String toString(){
        String message;
        String type;
        if (customer instanceof Senior){
            type = "Senior";
        }else if (customer instanceof Adult){
            type = "Adult";
        }else{
            type = "Student";
        }
        message = "Account: " + accountNumber + "\nOwner: " +
            customer.getName() + "\nType of customer: " +
            type + "\nBalance: $" + balance;
        return message;
    }

    /** Allocate a new array to hold the transactions. */
    public void reallocate() {
        capacity *= 2;
        Transaction[] newTransaction = new Transaction[capacity];
        System.arraycopy(transaction, 0, newTransaction, 0, transaction.length);
        transaction = newTransaction;
    }

    abstract double deposit(double amount);
    abstract double withdrawal(double amount);
}

```

Customer.java

```

/*
 * Customer.java
 *
 * Created on July 21, 2004, 12:21 AM
 */
abstract class Customer {
    private String name;
    private String address;
    private int age;
    private String telephoneNumber;
    private String customerNumber;
    protected static int custNumber = 0;

    /** Creates a new instance of Customer */
    public Customer() {
    }

    /**
     * Returns customer's name
     * @return String customer's name
     */
    public String getName(){
        return this.name;
    }
}

```

```

/*****
 * Returns customer's address
 * @return String customer's address
 */
public String getAddress(){
    return this.address;
}

/*****
 * Returns customer's age
 * @return int customer's age
 */
public int getAge(){
    return this.age;
}

/*****
 * Returns customer's phone number
 * @return String customer's phone number
 */
public String getTelephoneNumber(){
    return this.telephoneNumber;
}

/*****
 * Returns customer's number
 * @return String customer's number
 */
public String getCustomerNumber(){
    return this.customerNumber;
}

/*****
 * Changes customer's name
 * pre: name must be a not null string
 * post: customer's name changes to <name>
 * @param String name New name
 */
public void setCustomerName(String name){
    this.name = name;
}

/*****
 * Changes customer's address
 * pre: address must be a not null string
 * post: customer's address changes to <address>
 * @param String address New address
 */
public void setCustomerAddress(String address){
    this.address = address;
}

/*****
 * Changes customer's age
 * pre: age must be a positive value
 * post: customer's age changes to <age>
 * @param String age New age
 */
public void setCustomerAge(int age){
    this.age = age;
}

/*****

```

```

    * Changes customer's phone number
    * pre: phone number must be a not null string
    * post: customer's phone number changes to <phoneNumber>
    * @param String phoneNumber New number
    */
    public void setCustomerTelephoneNumber(String phoneNumber){
        this.telephoneNumber = phoneNumber;
    }

    /*****
    * Changes customer's number
    * pre: customerNumber must be a not null string
    * post: customer's number changes to <customerNumber>
    * @param String customerNumber New customer number
    */
    public void setCustomerNumber(String customerNumber){
        this.customerNumber = customerNumber;
    }

    abstract double getSavingsInterest();
    abstract double getCheckInterest();
    abstract double getCheckCharge();
    abstract double getOverdraftPenalty();
}

```

SavingsAccount.java

```

/*
 * SavingsAccount.java
 *
 * Created on July 21, 2004, 12:50 AM
 */
public class SavingsAccount extends Account{

    /** Creates a new instance of SavingsAccount */
    public SavingsAccount(Customer customer) {
        this.customer = customer;
        balance = 0;
        accountNumber = Integer.toString(accNumber);
        accNumber++;
        transaction = new Transaction[INIT_CAP];
        tranIndex = 0;
    }

    /*****
    * Adds amount to balance
    * pre: amount must be a positive value
    * post: balance increases in amount value
    * @param amount double Deposit amount
    * @return double New account balance
    */
    public double deposit(double amount){
        if (tranIndex == capacity)
            reallocate();
        transaction[tranIndex] = new Transaction(
            customer.getCustomerNumber(), 0, amount, "DEP");
        tranIndex++;
        balance += amount;
        addInterest();
    }
}

```

```

        return balance;
    }

    /**
     * Subtracts amount from balance
     * pre: amount must be a positive value
     * post: balance decreases in amount value
     * @param amount double Withdrawal amount
     * @return double New account balance
     */
    public double withdrawal(double amount){
        if (tranIndex == capacity)
            reallocate();
        transaction[tranIndex] = new Transaction(
            customer.getCustomerNumber(), 0, amount, "CR");
        tranIndex++;
        balance -= amount;
        return balance;
    }

    /**
     * Adds amount to balance
     * pre: amount must be a positive value
     * post: balance increases in amount value
     * @param amount double Interest amount
     * @return double New account balance
     */
    public double addInterest(){
        double amount;

        amount = balance * customer.getSavingsInterest();
        if (tranIndex == capacity)
            reallocate();
        transaction[tranIndex] = new Transaction(
            customer.getCustomerNumber(), 0, amount, "INT");
        tranIndex++;
        balance += amount;
        return balance;
    }
}

```

CheckingAccount.java

```

/*
 * CheckingAccount.java
 *
 * Created on July 21, 2004, 12:57 AM
 */

public class CheckingAccount extends Account{

    /** Creates a new instance of CheckingAccount */
    public CheckingAccount(Customer customer) {
        this.customer = customer;
        this.balance = 0;
        this.accountNumber = Integer.toString(accNumber);
        accNumber++;
        this.transaction = new Transaction[INIT_CAP];
        tranIndex = 0;
    }

    /**

```

```

    * Adds amount to balance
    * pre: amount must be a positive value
    * post: balance increases in amount value
    * @param amount double Deposit amount
    * @return double New account balance
    */
public double deposit(double amount){
    if (tranIndex == capacity)
        reallocate();
    transaction[tranIndex] = new Transaction(
        customer.getCustomerNumber(), 0, amount, "DEP");
    tranIndex++;
    balance += amount;
    return balance;
}

/*****
 * Subtracts amount from balance
 * pre: amount must be a positive value
 * post: balance decreases in amount value
 * @param amount double Withdrawal amount
 * @return double New account balance
 */
public double withdrawal(double amount){
    if (tranIndex == capacity)
        reallocate();

    transaction[tranIndex] = new Transaction(
        customer.getCustomerNumber(), 0, amount, "CR");
    tranIndex++;

    //add charge for using checking account
    amount += customer.getCheckCharge();
    if (amount > balance){
        //add overdraft penalty fee
        amount += customer.getOverdraftPenalty();
    }
    balance -= amount; //amount can exceed balance because of overdraft
    return balance;
}

/*****
 * Adds amount to balance
 * pre: amount must be a positive value
 * post: balance increases in amount value
 * @param amount double Interest amount
 * @return double New account balance
 */
public double addInterest(){
    double amount;

    amount = balance * customer.getCheckInterest();
    if (tranIndex == capacity)
        reallocate();
    transaction[tranIndex] = new Transaction(
        customer.getCustomerNumber(), 0, amount, "INT");
    tranIndex++;
    balance += amount;
    return balance;
}
}

```

Senior.java

```

/*
 * Senior.java
 *
 * Created on July 21, 2004, 1:13 AM
 */
public class Senior extends Customer{
    public static final double SAVINGS_INTEREST = 0.04; //4%
    public static final double CHECK_INTEREST = 0.01; //1%
    public static final double CHECK_CHARGE = 0.01; //1 cent
    public static final double OVERDRAFT_PENALTY = 25; //$25

    /** Creates a new instance of Senior */
    public Senior(String cName, String cAddress, int cAge,
        String cPhoneNumber) {
        this.setCustomerName(cName);
        this.setCustomerAddress(cAddress);
        this.setCustomerAge(cAge);
        this.setCustomerTelephoneNumber(cPhoneNumber);
        this.setCustomerNumber(Integer.toString(custNumber));
        this.custNumber++;
    }

    double getSavingsInterest(){
        return this.SAVINGS_INTEREST;
    }

    double getCheckInterest(){
        return this.CHECK_INTEREST;
    }

    double getCheckCharge(){
        return this.CHECK_CHARGE;
    }

    double getOverdraftPenalty(){
        return this.OVERDRAFT_PENALTY;
    }
}

```

Adult.java

```

/*
 * Adult.java
 *
 * Created on July 21, 2004, 1:17 AM
 */
public class Adult extends Customer{

    public static final double SAVINGS_INTEREST = 0.03; //3%
    public static final double CHECK_INTEREST = 0.01; //1%
    public static final double CHECK_CHARGE = 0.03; //3 cents
    public static final double OVERDRAFT_PENALTY = 25; //$25

    /** Creates a new instance of Adult */
    public Adult(String cName, String cAddress, int cAge,
        String cPhoneNumber) {
        this.setCustomerName(cName);
        this.setCustomerAddress(cAddress);
        this.setCustomerAge(cAge);
        this.setCustomerTelephoneNumber(cPhoneNumber);
        this.setCustomerNumber(Integer.toString(custNumber));
    }
}

```

```

        this.custNumber++;
    }

    double getSavingsInterest() {
        return this.SAVINGS_INTEREST;
    }

    double getCheckInterest() {
        return this.CHECK_INTEREST;
    }

    double getCheckCharge() {
        return this.CHECK_CHARGE;
    }

    double getOverdraftPenalty() {
        return this.OVERDRAFT_PENALTY;
    }
}

```

Student.java

```

/*
 * Student.java
 *
 * Created on July 21, 2004, 1:18 AM
 */

public class Student extends Customer{

    public static final double SAVINGS_INTEREST = 0.04; //4%
    public static final double CHECK_INTEREST = 0.01; //1%
    public static final double CHECK_CHARGE = 0.02; //2 cents
    public static final double OVERDRAFT_PENALTY = 25; //$25

    /** Creates a new instance of Student */
    public Student(String cName, String cAddress, int cAge,
        String cPhoneNumber) {
        this.setCustomerName(cName);
        this.setCustomerAddress(cAddress);
        this.setCustomerAge(cAge);
        this.setCustomerTelephoneNumber(cPhoneNumber);
        this.setCustomerNumber(Integer.toString(custNumber));
        this.custNumber++;
    }

    double getSavingsInterest() {
        return this.SAVINGS_INTEREST;
    }

    double getCheckInterest() {
        return this.CHECK_INTEREST;
    }

    double getCheckCharge() {
        return this.CHECK_CHARGE;
    }

    double getOverdraftPenalty() {
        return this.OVERDRAFT_PENALTY;
    }
}

```

Transaction.java


```

/*
 * Transaction.java
 *
 * Created on July 21, 2004, 1:18 AM
 */

public class Transaction {
    private String customerNumber;
    private int transactionType;
    private double amount;
    private String date;
    private String fees;

    /** Creates a new instance of Transaction */
    public Transaction(String customerNumber, int tranType,
        double amount, String fees) {
        this.customerNumber = customerNumber;
        this.transactionType = tranType;
        this.amount = amount;
        this.fees = fees;
    }

    public void processTran(){
        //Insert processing functionality here (e.g., save to a file)
    }
}

```

Bank.java

```

/*
 * Bank.java
 *
 * Created on July 21, 2004, 1:23 AM
 */

public class Bank {
    private Account[] accounts;
    private int size;
    private int capacity;

    private static final int SAVINGS = 0;
    private static final int CHECKING = 1;
    private static final int SENIOR = 0;
    private static final int ADULT = 1;
    private static final int STUDENT = 2;
    private static final int INIT_CAPACITY = 100;

    /** Creates a new instance of Bank */
    public Bank() {
        accounts = new Account[INIT_CAPACITY];
        capacity = INIT_CAPACITY;
    }

    /**
     * Creates a new account.
     * pre: customer information must be not null and types must be valid
     * post: New account added to bank
     * @param customerName String Account owner's name
     * @param customerAddress String Owner's address
     * @param customerAge int Owner's age (in years)
     * @param customerPhoneNumber String Owner's phone number
     * @param customerType int Owner's type of customer within bank
     */
}

```

```

* @param typeAccount int Account type (savings or checking)
* @return String New account number
*/
public String addAccount(String customerName, String customerAddress,
                        int customerAge, String customerPhoneNumber,
                        int customerType, int typeAccount){
    String accountNumber;
    Customer customer;

    if (customerType == SENIOR){
        customer = new Senior(
            customerName, customerAddress, customerAge, customerPhoneNumber);
    }else if (customerType == ADULT){
        customer = new Adult(
            customerName, customerAddress, customerAge, customerPhoneNumber);
    }else{
        customer = new Student(
            customerName, customerAddress, customerAge, customerPhoneNumber);
    }

    if (size == capacity)
        reallocate();

    if (typeAccount == SAVINGS){
        accounts[size] = new SavingsAccount(customer);
    }else{
        accounts[size] = new CheckingAccount(customer);
    }
    accountNumber = accounts[size].getNumber();
    size++;
    return accountNumber;
}

/*****
* Make a deposit into account.
* pre: amount must be a positive integer
* post: Account's balance increases
* @param accountNumber String Account's number
* @param amount double Amount to deposit
* @return double New balance
*/
public String makeDeposit(String accountNumber, double amount){
    int index = find(accountNumber);
    accounts[index].deposit(amount);
    return Double.toString(accounts[index].getBalance());
}

/*****
* Make a withdrawal from account.
* pre: amount must be a positive integer
* post: Account's balance decreases
* @param accountNumber String Account's number
* @param amount double Amount to withdraw
* @return double New balance
*/
public String makeWithdrawal(String accountNumber, double amount){
    int index = find(accountNumber);
    accounts[index].withdrawal(amount);
    return Double.toString(accounts[index].getBalance());
}

/*****

```

```

    * Returns account information as a string so it can be displayed
    * @param accountNumber String Account's number
    * @return String Account information as a String object
    */
    public String getAccount(String accountNumber){
        int index = find(accountNumber);
        return accounts[index].toString();
    }

    /**
     * Given an account number tells if the account exists or not
     * @param accountNumber String Account's number
     * @return int TRUE if accountNumber exists in accounts[]. Otherwise, -1.
     */
    private int find(String accountNumber){
        for (int i = 0; i < accounts.length; i++){
            if (accounts[i].getNumber().equals(accountNumber)){
                return i;
            }
        }
        return (-1);
    }

    /** Allocate a new array to hold the transactions. */
    private void reallocate() {
        capacity *= 2;
        Account[] newAccounts = new Account[capacity];
        System.arraycopy(accounts, 0, newAccounts, 0, accounts.length);
        accounts = newAccounts;
    }
}

```

BankApp.java

```

/*
 * BankApp.java
 *
 * Created on July 21, 2004, 1:44 AM
 */

public class BankApp {
    public Bank bank;

    /** Creates a new instance of BankApp */
    public BankApp() {
        bank = new Bank();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        BankApp bankApp = new BankApp();
        BankGUI gui = new BankGUI();
        gui.processCommands(bankApp.bank);
    }
}

```

BankGUI.java

```

/*
 * BankGUI.java
 *
 * Created on July 21, 2004, 10:21 AM
 */
import javax.swing.*;

/*****
 * This class is an implementation of PDUserInterface
 * that uses JOptionPane to display the menu of command choices.
 * @author Rafael
 */
public class BankGUI {

    /** A reference to the Bank object to be processed.
     Globally available to the command-processing methods.
     */
    private Bank theBank = null;

    // Methods
    /** Method to display the command choices and process user
     commands.
     pre: The bank exists and has accounts.
     post: Accounts are updated based on user commands.
     @param bank A reference to the Bank
     to be processed.
     */
    public void processCommands(Bank bank) {
        String[] commands = {"Add Account",
            "Deposit",
            "Withdrawal",
            "Check Account",
            "Exit"};

        theBank = bank;

        int choice;
        do {
            choice = JOptionPane.showOptionDialog(
                null, // No parent
                "Select action", // Prompt message
                "Bank System", // Window title
                JOptionPane.YES_NO_CANCEL_OPTION, // Option type
                JOptionPane.QUESTION_MESSAGE, // Message type
                null, // Icon
                commands, // List of commands
                commands[commands.length - 1]); // Default choice
            switch (choice) {
                case 0: doAddAccount(); break;
                case 1: doDeposit(); break;
                case 2: doWithdrawal(); break;
                case 3: doCheckAccount(); break;
                case 4: System.exit(0);
            }
        } while (choice < commands.length - 1);
        System.exit(0);
    }

    /** Method to add an account.
     pre: The bank exists and has accounts and customers.
     post: A new account is created
     */

```

```

private void doAddAccount() {
    // Request the name
    String customerName = JOptionPane.showInputDialog(
        "Enter Customer Name");
    if (customerName == null) {
        return; // Dialog was cancelled.
    }

    // Request the address
    String customerAddress = JOptionPane.showInputDialog(
        "Enter Customer Address");
    if (customerAddress == null) {
        return; // Dialog was cancelled.
    }

    // Request the age
    String age = JOptionPane.showInputDialog("Enter Customer Age");
    if (age == null) {
        return; // Dialog was cancelled.
    }
    int customerAge = Integer.parseInt(age);

    // Request the phone number
    String customerPhoneNumber = JOptionPane.showInputDialog(
        "Enter Customer Phone Number");
    if (customerPhoneNumber == null) {
        return; // Dialog was cancelled.
    }

    //Request type of customer
    String[] custType = {"Senior", "Adult", "Student", "Cancel"};

    int choice;
    choice = JOptionPane.showOptionDialog(
        null, // No parent
        "Select customer type", // Prompt message
        "Bank System", // Window title
        JOptionPane.YES_NO_CANCEL_OPTION, // Option type
        JOptionPane.QUESTION_MESSAGE, // Message type
        null, // Icon
        custType, // List of commands
        custType[custType.length - 1]); // Default choice

    if (choice == custType.length - 1){
        return; //Dialog was cancelled.
    }

    int customerType = choice;

    //Request type of account
    String[] commands = {"Savings", "Checking", "Cancel"};

    choice = JOptionPane.showOptionDialog(
        null, // No parent
        "Select account type", // Prompt message
        "Bank System", // Window title
        JOptionPane.YES_NO_CANCEL_OPTION, // Option type
        JOptionPane.QUESTION_MESSAGE, // Message type
        null, // Icon
        commands, // List of commands
        commands[commands.length - 1]); // Default choice

    if (choice == commands.length - 1){

```

```

        return; //Dialog was cancelled.
    }

    String theNumber = theBank.addAccount(customerName,
                                           customerAddress, customerAge,
                                           customerPhoneNumber, customerType, choice);

    String message = null;
    message = "Account " + theNumber + " created.";
    // Display confirmation message.
    JOptionPane.showMessageDialog(null, message);
}

/** Method to deposit.
pre: The bank exists and has accounts.
post: Balance in accounts increases.
*/
private void doDeposit() {
    // Request the account number.
    String accountNumber = JOptionPane.showInputDialog(
        "Enter Account Number");
    if (accountNumber == null) {
        return; // Dialog was cancelled.
    }

    String theAmount = JOptionPane.showInputDialog("Enter Amount");
    if (theAmount == null) {
        return; // Dialog was cancelled.
    }

    double amount = Double.parseDouble(theAmount);

    // Look up the name.
    String theBalance = theBank.makeDeposit(accountNumber, amount);
    String message = null;
    if (theBalance != null) { // Name was found.
        message = "Account " + accountNumber + " new balance $" +
            theBalance;
    } else { // Name was not found.
        message = accountNumber + " does not exist";
    }
    // Display the result.
    JOptionPane.showMessageDialog(null, message);
}

/** Method to withdrawal.
pre: The bank exists and has accounts.
post: Balance in accounts decreases.
*/
private void doWithdrawal() {
    // Request the account number.
    String accountNumber = JOptionPane.showInputDialog(
        "Enter Account Number");
    if (accountNumber == null) {
        return; // Dialog was cancelled.
    }

    String theAmount = JOptionPane.showInputDialog("Enter Amount");
    if (theAmount == null) {
        return; // Dialog was cancelled.
    }

    double amount = Double.parseDouble(theAmount);

```

```

        // Look up the name.
        String theBalance = theBank.makeWithdrawal(accountNumber, amount);
        String message = null;
        if (theBalance != null) { // Name was found.
            message = "Account " + accountNumber + " new balance $" +
                theBalance;
        } else { // Name was not found.
            message = accountNumber + " does not exist";
        }
        // Display the result.
        JOptionPane.showMessageDialog(null, message);
    }

    /** Method to deposit.
    pre: The bank exists and has accounts.

    */
    private void doCheckAccount() {
        // Request the account number.
        String accountNumber = JOptionPane.showInputDialog(
            "Enter Account Number");
        if (accountNumber == null) {
            return; // Dialog was cancelled.
        }

        // Look up the number.
        String theAccount = theBank.getAccount(accountNumber);
        String message = null;
        if (theAccount != null) { // Name was found.
            message = theAccount;
        } else { // Name was not found.
            message = accountNumber + " does not exist";
        }
        // Display the result.
        JOptionPane.showMessageDialog(null, message);
    }
}

```