/\*\*

This program demonstrates the BankAccount and

derived classes.

\*/

public class AccountDriver

{

public static void main(String[] args)

{

double put\_in = 500;

double take\_out = 1000;

String money;

String money\_in;

String money\_out;

boolean completed;

// Test the CheckingAccount class.

CheckingAccount myCheckingAccount =

new CheckingAccount("Benjamin Franklin", 1000);

System.out.println("Account Number " +

myCheckingAccount.

getAccountNumber() +

" belonging to " +

myCheckingAccount.getOwner());

money = String.format("%.2f",

myCheckingAccount.

getBalance());

System.out.println("Initial balance = $" + money);

myCheckingAccount.deposit(put\_in);

money\_in = String.format("%.2f", put\_in);

money = String.format("%.2f",

myCheckingAccount.

getBalance());

System.out.println("After deposit of $" +

money\_in + ", balance = $" +

money);

completed = myCheckingAccount.withdraw(take\_out);

money\_out = String.format("%.2f", take\_out);

money = String.format("%.2f",

myCheckingAccount.

getBalance());

if (completed)

{

System.out.println("After withdrawal of $" +

money\_out + ", balance = $" +

money);

}

else

{

System.out.println("Insuffient funds to " +

"withdraw $" + money\_out +

", balance = $" + money);

}

System.out.println();

// Test the SavingsAccount class.

SavingsAccount yourAccount =

new SavingsAccount("William Shakespeare", 400);

System.out.println("Account Number " +

yourAccount.getAccountNumber() +

" belonging to " +

yourAccount.getOwner());

money = String.format("%.2f",

yourAccount.getBalance());

System.out.println("Initial balance = $" + money);

yourAccount.deposit(put\_in);

money\_in = String.format("%.2f", put\_in);

money = String.format("%.2f",

yourAccount.getBalance());

System.out.println("After deposit of $" +

money\_in + ", balance = $" +

money);

completed = yourAccount.withdraw(take\_out);

money\_out = String.format("%.2f", take\_out);

money = String.format("%.2f",

yourAccount.getBalance());

if (completed)

{

System.out.println("After withdrawal of $" +

money\_out + ", balance = $" +

money);

}

else

{

System.out.println("Insuffient funds " +

"to withdraw $" + money\_out +

", balance = $" + money);

}

yourAccount.postInterest();

money = String.format("%.2f",

yourAccount.getBalance());

System.out.println("After monthly interest " +

"has been posted," +

"balance = $" + money);

System.out.println();

// Test the copy constructor of the

// SavingsAccount class.

SavingsAccount secondAccount =

new SavingsAccount(yourAccount, 5);

System.out.println("Account Number " +

secondAccount.

getAccountNumber() +

" belonging to " +

secondAccount.getOwner());

money = String.format("%.2f",

secondAccount.getBalance());

System.out.println("Initial balance = $" + money);

secondAccount.deposit(put\_in);

money\_in = String.format("%.2f", put\_in);

money = String.format("%.2f",

secondAccount.getBalance());

System.out.println("After deposit of $" + money\_in +

", balance = $" + money);

secondAccount.withdraw(take\_out);

money\_out = String.format("%.2f", take\_out);

money = String.format("%.2f",

secondAccount.getBalance());

if (completed)

{

System.out.println("After withdrawal of $" +

money\_out + ", balance = $" +

money);

}

else

{

System.out.println("Insuffient funds " +

"to withdraw $" + money\_out +

", balance = $" + money);

}

System.out.println();

// Test to make sure new accounts are

// numbered correctly.

CheckingAccount yourCheckingAccount =

new CheckingAccount("Issac Newton", 5000);

System.out.println("Account Number " +

yourCheckingAccount.

getAccountNumber() +

" belonging to " +

yourCheckingAccount.getOwner());

}

}

/\*\*

\*

\* Bank Account Class

\*/

/\*\*

The BankAccount class is an abstract class that holds

general data about a bank account. Classes representing

specific types of bank accounts should inherit from

this class.

\*/

public abstract class BankAccount

{

// Class variable so that each account

// has a unique number

protected static int numberOfAccounts = 100001;

// Current balance in the account

private double balance;

// Name on the account

private String owner;

// Number bank uses to identify account

private String accountNumber;

/\*\*

Default constructor

\*/

public BankAccount()

{

balance = 0;

accountNumber = numberOfAccounts + "";

numberOfAccounts++;

}

/\*\*

Standard constructor

@param name The owner of the account.

@param amount The beginning balance.

\*/

public BankAccount(String name, double amount)

{

owner = name;

balance = amount;

accountNumber = numberOfAccounts + "";

numberOfAccounts++;

}

/\*\*

Copy constructor creates another account

for the same owner.

@param oldAccount The account with information

to copy.

@param amount The beginning balance of the

new account.

\*/

public BankAccount(BankAccount oldAccount,

double amount)

{

owner = oldAccount.owner;

balance = amount;

accountNumber = oldAccount.accountNumber;

}

/\*\*

Allows you to add money to the account.

@param amount The amount to deposit in the account.

\*/

public void deposit(double amount)

{

balance = balance + amount;

}

/\*\*

Allows you to remove money from the account if

enough money is available, returns true if the

transaction was completed, returns false if

there was not enough money.

@param amount The amount to withdraw from

the account.

@return True if there was sufficient funds to

complete the transaction, false otherwise.

\*/

public boolean withdraw(double amount)

{

boolean completed = true;

if (amount <= balance)

{

balance = balance - amount;

}

else

{

completed = false;

}

return completed;

}

/\*\*

Accessor method to balance

@return The balance of the account.

\*/

public double getBalance()

{

return balance;

}

/\*\*

accessor method to owner

@return The owner of the account.

\*/

public String getOwner()

{

return owner;

}

/\*\*

Accessor method to account number

@return The account number.

\*/

public String getAccountNumber()

{

return accountNumber;

}

/\*\*

Mutator method to change the balance

@param newBalance The new balance for the account.

\*/

public void setBalance(double newBalance)

{

balance = newBalance;

}

/\*\*

Mutator method to change the account number

@param newAccountNumber The new account number.

\*/

public void setAccountNumber(String newAccountNumber)

{

accountNumber = newAccountNumber;

}

}

/\*

\*

\* CheckingAccount Class

\*/

public class CheckingAccount extends BankAccount {

private static double Fee = 0.15;

private String AccountNumber;

//Initializing CheckingAccount Constructor taken from Super Class

public CheckingAccount (String fullName, double initialAmmount){

super(fullName, initialAmmount);

AccountNumber = super.getAccountNumber() + "-" + "10";

}

//Returns accountNumber for the Checking Account class

public String getAccountNumber() {

return AccountNumber;

}

@Override

//OverRide withdraw method to add fee to amount

public boolean withdraw (double ammount) {

ammount += Fee;

return super.withdraw(ammount);

}

}

/\*

\* SavingAccount Class

\*/

public class SavingsAccount extends BankAccount {

private double rate = 2.5;

private int savingsNumber = 0;

private String AccountNumber;

//SavingsAccount Constructor, Inherited from Super class

public SavingsAccount(String name, double ammount) {

super (name , ammount);

AccountNumber = super.getAccountNumber() + "-" + savingsNumber;

savingsNumber++;

}

//Method to calculate interest and add it to the balance

public void postInterest () {

double interest;

double amount;

interest = (rate/12)/100;

amount = getBalance()\*interest;

deposit(amount);

}

/\*\*

\* A copy constructor

\* @param savings

\* @param initialAmmount

\*/

public SavingsAccount (SavingsAccount savings, double initialAmmount){

super(savings, initialAmmount);

AccountNumber = super.getAccountNumber() + "-" + savingsNumber;

savingsNumber++;

}

//Getter account Number method

public String getAccountNumber(){

return AccountNumber;

}

}

// ---------------------------------------------------Output

Account Number 100001-10 belonging to Benjamin Franklin

Initial balance = $1000.00

After deposit of $500.00, balance = $1500.00

After withdrawal of $1000.00, balance = $499.85

Account Number 100002-0 belonging to William Shakespeare

Initial balance = $400.00

After deposit of $500.00, balance = $900.00

Insuffient funds to withdraw $1000.00, balance = $900.00

After monthly interest has been posted,balance = $901.88

Account Number 100002-0 belonging to William Shakespeare

Initial balance = $5.00

After deposit of $500.00, balance = $505.00

Insuffient funds to withdraw $1000.00, balance = $505.00

Account Number 100003-10 belonging to Isaac Newton

Process finished with exit code 0