Account class: keeps track of bank account details.

Instance variables:
- name, address, accountNumber, balance

Methods:
- displayDetails, inCredit (true iff balance is nonnegative),
- credit, debit (add/subtract specified amount to/from balance),
- main

Skeleton of Account class definition:

```java
public class Account {
    String name;
    String address;
    String accountNumber;
    double balance;

    ... other stuff here
}
```
public void displayDetails() {
    System.out.println("Name: " + name);
    System.out.println("Address: " + address);
    System.out.println("Account number: " + accountNumber);
    System.out.println("Balance: " + balance);
}

customor boolean inCredit() {
    return balance >= 0.0;
}

customor void credit(double amount) {
    balance += amount;
}

customor void debit(double amount) {
    balance -= amount;
}
public static void main(String args[]) {
    Account azw = new Account();
    azw.name = "Aaron Werschulz";
    azw.address = "123 Main Street, Somewhere NJ";
    azw.accountNumber = "123-45-6789";
    azw.balance = 150.23;
    azw.displayDetails();
    azw.credit(12.75);
    azw.displayDetails();
    azw.debit(14.92);
    azw.displayDetails();
    if (azw.inCredit())
        System.out.println("In credit");
    else
        System.out.println("Overdrawn");
}
```java
public Account (String theName, String theAddress, 
    String theAccountNumber, 
    double theBalance) {
    name = theName;
    address = theAddress;
    accountNumber = theAccountNumber;
    balance = theBalance;
}

public static void main(String args[]) {
    Account azw = new Account("Aaron Werschulz", 
        "123 Main Street, Somewhere NJ", 
        "123-45-6789", 150.23);
    azw.displayDetails();
    azw.credit(12.75);
    azw.displayDetails();
    azw.debit(14.92);
    azw.displayDetails();
    if (azw.inCredit())
        System.out.println("In credit");
    else
        System.out.println("Overdrawn");
}
```
Encapsulating Data

Enforce the “principle of least privilege”. So:

```java
text.*;

public class Account {
    private String name;
    private String address;
    private String accountNumber;
    private double balance;
    private DecimalFormat money = new DecimalFormat("$0.00");

    public Account (String theName, String theAddress, String theAccountNumber, double theBalance)
    {
        name = theName;
        address = theAddress;
        accountNumber = theAccountNumber;
        balance = theBalance;
    }
```
public void displayDetails() {
    System.out.println("Name: " + name);
    System.out.println("Address: " + address);
    System.out.println("Account number: " + accountNumber);
    System.out.println("Balance: " + money.format(balance));
}

public void setName(String customerName) {
    name = customerName;
}

public void setAddress(String customerAddress) {
    address = customerAddress;
}

public void setAccountNumber(String customerNumber) {
    accountNumber = customerNumber;
}

public void setBalance (double customerBalance) {
    balance = customerBalance;
}
public String getName() {
    return name;
}

public String getAddress() {
    return address;
}

public String getAccountNumber() {
    return accountNumber;
}

public double getBalance() {
    return balance;
}
public void credit(double amount) {
    balance += amount;
}

public void debit(double amount) {
    balance -= amount;
}

public boolean inCredit() {
    return balance >= 0.0;
}

public class NewAccount {
    public static void main(String args[]) {
        Account azw = new Account("Aaron Werschulz", "123 Main Street, Somewhere NJ", "123-45-6789", -3.75);
        azw.displayDetails();
        if (azw.inCredit())
            System.out.println("In credit");
        else
            System.out.println("Overdrawn");
        azw.setAddress("1 Marsh St, Cranford NJ");
        System.out.println("Confirming new address: " + azw.getAddress());
        azw.credit(42.75);
        azw.displayDetails();
        azw.debit(14.92);
        azw.displayDetails();
    }
}
$ javac NewAccount.java
$ java NewAccount
Name: Aaron Werschulz
Address: 123 Main Street, Somewhere NJ
Account number: 123-45-6789
Balance: -$3.75
Overdrawn
Confirming new address: 1 Marsh St, Cranford NJ
Name: Aaron Werschulz
Address: 1 Marsh St, Cranford NJ
Account number: 123-45-6789
Balance: $39.00
Name: Aaron Werschulz
Address: 1 Marsh St, Cranford NJ
Account number: 123-45-6789
Balance: $24.08
Calling methods

Classes: A  B

Class A has a (void) method m1 (which B can access), as well as another (void) method m2.

Class B can have code such as

```
A a = new A();
.
.
.
a.m1();
```

However, within class A we only need to reference

```
m2();
```

(means invoke method m2 upon current object). Could also write

```
this.m2();
```

(Can also use this for data members.)
For example, might add new method `creditReport()` to Account class.

```java
public void creditReport() {
    String m;
    displayDetails();
    if (inCredit())
        m = "in credit";
    else if (getBalance() > -100)
        m = "overdrawn: send notification";
    else if (getBalance() > -1000)
        m = "overdrawn: send warning letter";
    else
        m = "overdrawn: send suspend account notification";
    System.out.println("Account " + m);
}
```
Its use within the NewAccount class?

```java
public class NewAccount {

    public static void main(String[] args) {
        Account azw = new Account("Aaron Werschulz", "123 Main Street, Somewhere NJ", "123-45-6789", -3.75);
        azw.creditReport();
        azw.credit(600.00);
        azw.creditReport();
    }
}
```
Sample run?

Name: Aaron Werschulz
Address: 123 Main Street, Somewhere NJ
Account number: 123-45-6789
Balance: -$3.75
Account overdrawn: send notification
Name: Aaron Werschulz
Address: 123 Main Street, Somewhere NJ
Account number: 123-45-6789
Balance: $596.25
Account in credit
One possible improvement

To print an Account, need to do
    azw.displayDetails();
If we could do
    System.out.println(azw);
then Account would resemble other datatypes.
Can do this if we provide a toString() method.

In our case, we would replace Account's displayDetails() method with the following:

    public String toString() {
        return "Name: " + name + "\n" +
            "Address: " + address + "\n" +
            "Account number: " + accountNumber + "\n" +
            "Balance: " + money.format(balance);
    }
A note on the Java compiler

If the class Foo involves class Bar, then

javac Foo.java

will also compile Bar.java if necessary, i.e., if Bar.java is newer than Bar.class.

This means that in the previous example, you need only do

javac NewAccount.java
Command-line arguments for applications

The `String[]` parameter of a public class's `public static void main()` method contains the command-line parameters.

**Example:** The following program prints out its command-line arguments:

```java
public class PrintArgs {
    public static void main(String[] args) {
        for (int i = 0; i < args.length; i++)
            System.out.println("Arg "+i+": "+args[i]);
    }
}
```

```
$ java PrintArgs sis boom bah
Arg 0: sis
Arg 1: boom
Arg 2: bah
```
Analogous version in C++:

```cpp
#include <iostream>

int main (int argc, char *argv[]) {
    for (int i = 0; i < argc; i++)
        std::cout << "Arg " << i << " : " << argv[i] << std::endl;
}
```

If executable is `printargs`, then we have

```
$ ./printargs sis boom bah
Arg 0: ./printargs
Arg 1: sis
Arg 2: boom
Arg 3: bah
```