4.0 Anatomy of a class (how to write a class)

```
1  //*******************************************************
2  // Account.java
3  //
4  // A bank account class with methods to deposit to, withdraw from,
5  // change the name on, charge a fee to, and print a summary of the account.
6  //*******************************************************
7
8  public class Account {
9      private double balance;
10     private String name;
11     private long acctNum;
12
13     //-----------------------------------------------
14     //Constructor -- initializes balance, owner, and account number
15     //-----------------------------------------------
16     public Account(double initBal, String owner, long number) {
17         balance = initBal;
18         name = owner;
19         acctNum = number;
20     }
21
22     //-----------------------------------------------
23     // Checks to see if balance is sufficient for withdrawal.
24     // If so, decrements balance by amount; if not, prints message.
25     //-----------------------------------------------
26     public void withdraw(double amount) {
27         if (balance >= amount)
28             balance -= amount;
29         else
30             System.out.println("Insufficient funds");
31     }
32
33     //-----------------------------------------------
34     // Adds deposit amount to balance.
35     //-----------------------------------------------
36     public void deposit(double amount) {
37         balance += amount;
38     }
```

1. The class header:
   - States that this code is the definition for an **Account** class.
   - The class name must match the file name, except the file name has .java on the end.
   - As per convention, use an upper-case letter to start a class name

2. Instance Variables
   - The **attributes** that objects of this class will have.
   - Each Account object instantiated (by a client) will have its own values for these variables. Don't set them here, just declare them.
   - According to the encapsulation principle, these should **always** be declared as private.
   - The constructor and all of the methods in this class can manipulate these variables.

3. The constructor
   - This block executes when a client instantiates an object of this class using the **new** statement.
   - Its purpose is to assign *initial* values to the instance variables. It can receive parameters from the client for this purpose.
   - It looks like a method but it's not, the header is slightly different:
     - There is no **void** or **return type** like for a method.
     - The name matches the class name.

4. The methods
   - Everything else in the class is *methods*.
   - The **behaviors** that objects of this class will have.
   - Methods are responsible for accessing and manipulating the instance variables declared up top.
   - **public** methods can be called by a client using the dot (.) operator. This is how you give clients access to functionality in your class.
   - **private** methods can only be called from within this class.
   - The methods can optionally receive parameters and/or return a value.

```
40      //-----------------------------------------------
41      // Returns balance.
42      //-----------------------------------------------
43      public double getBalance() {
44          return balance;
45      }
46
47
48      //-----------------------------------------------
49      // Prints name, account number, and balance.
50      //-----------------------------------------------
51      public void printSummary() {
52          //print name
53
54          //print acct number
55
56          //print balance
57
58      }
59
60      //-----------------------------------------------
61      // Deducts $10 service fee if balance is under $1000
62      //-----------------------------------------------
63      public void chargeFee() {
64
65      }
66
67      //-----------------------------------------------
68      // Changes the name on the account
69      //-----------------------------------------------
70      public void changeName(String newName) {
71
72      }
73
74  }
75
```