Writing good classes - Some general guidelines

Importance of data encapsulation : Keep data fields private unless absolutely necessary

Mutator and Accessor methods :
Be careful about providing accessor and mutator methods. Mutator methods can change states of an object, which may lead to un-expected consequences.
In accessor methods, providing a return reference value to an object could also lead to unexpected side effects, since possessing such a reference enables the possessor to change the state of an object from which the reference is returned. In cases where the state of an object is to be revealed in a method return, make a clone of the object and return a reference to it.

A class with no mutator method is called an **immutable** class. String class in Java is immutable.

The "Law" of Demeter (or Principle of Least Knowledge) is a design guideline that suggests that a method not operate on global objects or objects that are part of another object. See
http://www.ccs.neu.edu/research/demeter/demeter-method/LawOfDemeter/Smyth/priority-inversion
or
http://research.microsoft.com/en-us/um/people/mbj/mars_pathfinder/authoritative_account.html
to read about how one of the Mars Pathfinder team members, Dave Smyth, used the Law of Demeter in his work.

The Law of Demeter for functions requires that a method *M* of an object *O* may only invoke the methods of the following kinds of objects:

1. *O* itself
2. *M*'s parameters
3. any objects created/instantiated within *M*

Programming by Contract is a software design paradigm which was first introduced by Bertrand Meyer, the creator of the Eiffel programming language. Programming by contract creates a contract between the software developer and the software user. Every method starts with a pre-condition that must be satisfied by the user of the method, and ends with post-conditions which the developer guarantees to be true if and only if the preconditions were met. Also, each class has an *invariant* which must be satisfied after any changes to the object represented by the class. In the other words, the invariant guarantees the object is in a valid state before and after the method call (but not during the call).
The design by contract principle extends to the method level in each class. For each method call, the contract asks
what are the preconditions ? (what does the method expect?)
what are the postcondtions ? (what does the method guarantee?)
what are the invariants ? (what does the method maintain?)

Here are links to two good lectures on programming by contract:

http://www.cs.usfca.edu/~parrt/course/601/lectures/programming.by.contract.html

http://www.cs.unc.edu/~stotts/COMP204/contract.html