1) (a) There is no difference between the two tags. HTML is not case sensitive.  
(b) `<A HREF = http://www.ziporyn.com/audio/Amok6exc.mp3> MP3 Link </A>`
(c)  
   Hey diddle diddle  
   the cat and the fiddle  
   the cow jumped over the moon!!!
(d) The checkbox type creates a checkbox on the screen. The programmer can group the checkboxes together through the `name` attribute and can assign values to the checkboxes through the `value` attribute. Each of these attributes is required. The reset type allows the user to reset--erase or set to some default value--all elements in the form.

2) The following code should have been added to the page on which the submission was to be done. The input types can be altered.

```html
<FORM METHOD=POST ACTION="http://www.usc.edu/cgi-bin/form_handler">
  <INPUT TYPE="hidden" NAME="FH_Recipients" VALUE="youraccount@usc.edu">
  <INPUT TYPE="hidden" NAME="FH_Topic" VALUE="Form Handler Test">
  <INPUT TYPE="hidden" NAME="FH_OK_URL"
        VALUE="http://www.usc.edu/Help/Authors/OK.html">
  <INPUT TYPE="hidden" NAME="FH_ERROR_URL"
        VALUE="http://www.usc.edu/Help/Authors/ERROR.html">
  <INPUT TYPE="hidden" NAME="FH_Display_Order" VALUE="number, color, FH_Topic">
  <INPUT TYPE="hidden" NAME="FH_Required_Fields" VALUE="color, number">
  <B>Name:</B> <INPUT NAME="FH_Name"><P>
  <B>Email:</B> <INPUT NAME="FH_Email"><P>
  <B>Subject:</B> <INPUT NAME="FH_Subject"><P>
  Choose some colors:<P>
      <UL>
        <LI><INPUT TYPE="checkbox" NAME="color" VALUE="red"> Red<BR>
        <LI><INPUT TYPE="checkbox" NAME="color" VALUE="green"> Green<BR>
        <LI><INPUT TYPE="checkbox" NAME="color" VALUE="blue"> Blue<BR>
      </UL>
  Choose a number:<P>
      <UL>
        <LI><INPUT TYPE="radio" NAME="number" VALUE="1"> 1<BR>
        <LI><INPUT TYPE="radio" NAME="number" VALUE="2"> 2<BR>
        <LI><INPUT TYPE="radio" NAME="number" VALUE="3"> 3<BR>
      </UL>
  </P>
  <INPUT TYPE="submit" VALUE="Submit">
</FORM>
```
This answer is very similar to that of 1.1 on Chapter 9, pg. 269 of the book. Movie[*] indicates a variable-length collection of Movies. For both classes, the get operations simply return the relevant variables. The set operations set the appropriate variables. The first addMovie operation takes a Movie object as input and adds it to the movies collection. The second addMovie method creates a new Movie object with the input movieName and returns the object. The removeMovie method removes a Movie object from the movies collection and returns the object. The removeAllMovies method removes all Movie objects from Movies.

Because the multiplicity on the Person side is one, the operations in the Movie class for the person attribute is very simple. However, the zero or more multiplicity (*) on the Person side maps to a set of editing functions that are standard for collections. These are the design patterns related to multiplicity.
public class Wizard
{
    public static void main (String args[])
    {
        int year, ageHarry, ageAlbus;

        year = 2001;
        ageHarry = 21;
        ageAlbus = 150;

        System.out.println("Harry Potter was born in " +
                          calculateDOB(ageHarry, year) + ").");
        System.out.println("Albus Dumbledore was born in " +
                          calculateDOB(ageAlbus, year) + ").");
    }

    public static int calculateDOB(int a, int y)
    {
        int date;
        date = y - a;
        return date;
    }
}