Figure 9.2 The *Vector* class.

Here's an example of how to instantiate a **Vector**:

```
Vector list = new Vector();
```

here's a way to add things to it:

```
list.addElement("sock");
list.addElement("hat");
```

and here's a way to get things back out of it:

```
String item = (String) list.elementAt(0);
System.out.println("First item in list is " + item);
```

The "**(String)**" in the first statement is a *cast*. The compiler needs to know that the thing on the right-hand side is of the right type to be stored in the variable on the left, and the *cast* tells the compiler you're certain it's a **String**.

Question: Why can't the compiler figure this out for itself?

elementAt() doesn't remove items from the **Vector**; it just gives back a reference to the *indexed* item. Here's how we retrieve the second item in our list: ____________ .

We can also "jump around" when retrieving items. Assuming that our list has 16 items, here's how we could retrieve and print the 4-*th*, 8-*th*, and 12-*th* item:

```
for (                    ) {
    String item = (String) list.elementAt(i);
    System.out.println("(4-th) next item in list is " + item);
}
```

This example illustrates one of the more useful features of **Vectors**; namely that each element can be accessed *directly* without going through all previous elements.