Recursion on the dimensions of an array

In the box to the right is a single integer, 5, a one-dimensional array containing the two integers 4 and 3, and a two-dimensional array, where the three rows contain 4, 2, and 3 integers. The latter is a ragged array. Java allows ragged arrays. If you are not familiar with this term, look it up in JavaHyperText.

We want to write a function that will calculate the sum of all the integers in such an array, and it should work even for an array of 2, 3, 4, 5 — any number — of dimensions. We can do that using recursion! This is neat!

Below is the specification of function arraySum. Its parameter obj has type Object, so it can be any object whatsoever. But look carefully at the Precondition, which limits what the parameter can be. It must be an Integer, which we consider an array of 0 dimensions, or an Integer array of any number of dimensions.

```java
/** Return the sum of all integer values in obj.
 * Precondition: obj is one of the classes: Integer, Integer [], Integer [][][], etc.
 * None of the elements in obj is null. */
public static int arraySum(Object obj)
```

Consider writing function arraySum recursively. The base case is obviously a parameter whose type is Integer, and we can use operation instanceof to test for the base case.

In the recursive case, we know that parameter obj is an array, so we can cast obj down to type Object[]. We then have the parameter as an array, and we can call function arraySum recursively on each of its elements and add their returned values together. So, we write function arraySum like this:

```java
/** Return the sum of all integers in obj.
 * Precondition: obj is one of the classes: Integer, Integer [], Integer [][][], etc.
 * Precondition. None of the elements in obj is null. */
public static int arraySum(Object obj)
{
    if (obj instanceof Integer) return (Integer) obj;
    // obj is an Object array
    Object [] oba = (Object []) obj;
    int sum = 0;
    for(Object ob: oba)
        sum += arraySum(ob);
    return sum;
}
```

What a beautiful little recursive function that is!

Plop this function into some Java class and try it out on the examples we give below. Note that an integer like 5 represents an Integer object that contains that integer.

- For the argument 5, the value 5 is returned.
- For the argument array {1, 2, 3}, 6 is returned because 1+2+3 = 6.
- For the argument array {{1, 2, 5}, {3, 4}}, 15 is returned because 1+2+5+3+4 = 15.
- For the argument array {{1}, {0, 3}, {}}, {{1,2,3}, {3}}, 13 is returned because 1+0+3+0+1+2+3+3 = 13.