Figure 8.2 Variables of class type vs. primitive type.

We've talked about this before—variables of class type and primitive type are inherently different. For example, consider the class **Ex** defined by:

```
class Ex {
    public int field;
}
```

The table below compares class variables of type **Ex** to primitive variables of type **int**:

CLASS TYPE	*PRIMITIVE TYPE*
Ex e = new Ex();	**int i;**
Ex f = new Ex();	**int j;**
e.field = 2;	**i = 2;**
f.field = 3;	**j = 3;**
System.out.println(e.field);	**System.out.println(i);**
e = f;	**i = j;**
System.out.println(e.field);	**System.out.println(i);**
f.field = 4;	**j = 4;**
System.out.println(e.field);	**System.out.println(i);**

Variables of class type contain a *reference*—a way of "finding" the object referred to. Here's what's going on above:

i = j; $\longrightarrow$ copies the *value* of **j** into **i**
e = f; $\longrightarrow$ copies the value of **f** into **e**, *but that value is a reference*.

We often say "**e** is an object of the class **Ex**", and it's usually o.k. to think of it that way. Sometimes, though, (as in the above examples) it's important to remember that **e** is a *reference* to the object, and not the object itself.