Finally, by defaulting the copy-assignment operator in `Person5c` (shown in the example above), we regain all of the regular functionality of the original **aggregate class**, but because of the user-provided copy constructor, `Person5c` is no longer amenable to aggregate initialization. Note that the destructor is *never* suppressed by the explicit declaration of any other function.

6. User-provided copy-assignment operator. Once we understand the consequences of user-providing a copy constructor (e.g., `Person5c` in the example in item 4), there are no surprises here. Again, we'll provide, for reference, only the final, transitive result, `Person6b`:

```
struct Person6b  // adding a user-provided copy-assignment operator
{
    String firstName;
    String lastName;

    Person6b& operator=(const Person6b& rhs)    // copy assignment
    {
        firstName = rhs.firstName;
        lastName  = rhs.lastName;
        return *this;
    }

    // already added to Person6a (not shown)
    Person6b(Person6b&&) = default;             // move constructor
    Person6b& operator=(Person6b&&) = default;  // move assignment

    // new here in Person6b
    Person6b() = default;                       // default constructor
    Person6b(const Person6b&) = default;        // copy constructor
};
```

Again, we've omitted `Person6` and `Person6a` from the example above, but we'll walk through those revisions. Providing a user-defined copy-assignment operator in `Person6`, unlike providing a copy constructor, leaves the class of aggregate type but similarly suppresses the declaration of both the move constructor and move-assignment operator. Restoring move assignment in `Person6a` has no further suppressive effects, but restoring move construction in turn suppresses both default construction and copy construction. Class `Person6b` above provides the same functionality as the original aggregate `Person` class, i.e., including aggregate initialization, along with the ability to add a benign implementation, affecting no other special-member implementations, to the user-provided copy-assignment operator.