```
module outer {
    var local = 2;           // Non-exported local variable
    export var b = local;    // outer.b
    export module inner {
        export var y = 20;   // outer.inner.y
    }
}
```

Assuming the two source files are part of the same program, the two declarations will have the global module as their common root and will therefore contribute to the same module instance, the instance type of which will be:

```
{
    a: number;
    b: number;
    inner: {
        x: number;
        y: number;
    };
}
```

Declaration merging does not apply to local aliases created by import declarations. In other words, it is not possible have an import declaration and a module declaration for the same name within the same module body.

Declaration merging also extends to internal module declarations with the same qualified name relative to a common root as a function, class, or enum declaration:

- When merging a function and an internal module, the type of the function object is merged with the instance type of the module. In effect, the overloads or implementation of the function provide the call signatures and the exported members of the module provide the properties of the combined type.
- When merging a class and an internal module, the type of the constructor function object is merged with the instance type of the module. In effect, the overloads or implementation of the class constructor provide the construct signatures, and the static members of the class and exported members of the module provide the properties of the combined type. It is an error to have static class members and exported module members with the same name.
- When merging an enum and an internal module, the type of the enum object is merged with the instance type of the module. In effect, the members of the enum and the exported members of the module provide the properties of the combined type. It is an error to have enum members and exported module members with the same name.

When merging a non-ambient function or class declaration and a non-ambient internal module declaration, the function or class declaration must be located prior to the internal module declaration in the same source file. This ensures that the shared object instance is created as a function object. (While it