In the above example, 'SelectableControl' contains all of the members of 'Control', including the private 'state' property. Since 'state' is a private member it is only possible for descendants of 'Control' to implement 'SelectableControl'. This is because only descendants of 'Control' will have a 'state' private member that originates in the same declaration, which is a requirement for private members to be compatible (section 3.8).

Within the 'Control' class it is possible to access the 'state' private member through an instance of 'SelectableControl'. Effectively, a 'SelectableControl' acts like a 'Control' that is known to have a 'select' method. The 'Button' and 'TextBox' classes are subtypes of 'SelectableControl' (because they both inherit from 'Control' and have a 'select' method), but the 'Image' and 'Location' classes are not.

7.4 Dynamic Type Checks

TypeScript does not provide a direct mechanism for dynamically testing whether an object implements a particular interface. Instead, TypeScript code can use the JavaScript technique of checking whether an appropriate set of members are present on the object. For example, given the declarations in section 7.1, the following is a dynamic check for the 'MoverShaker' interface:

```
var obj: any = getSomeObject();
if (obj && obj.move && obj.shake && obj.getStatus) {
    var moverShaker = <MoverShaker> obj;
    ...
}
```

If such a check is used often it can be abstracted into a function:

```
function asMoverShaker(obj: any): MoverShaker {
    return obj && obj.move && obj.shake && obj.getStatus ? obj : null;
}
```