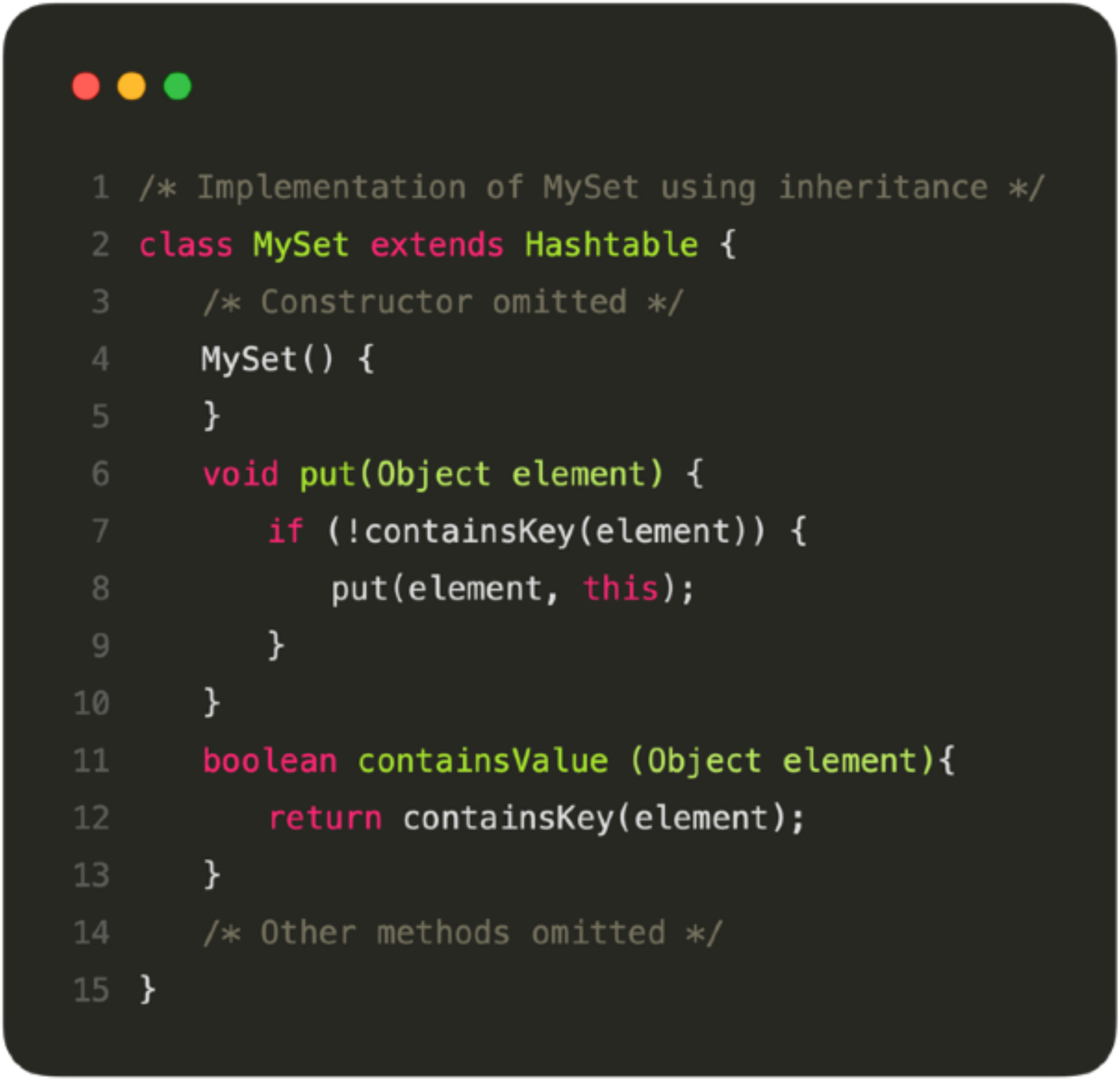
TEMPLE
UNIVERSITY®

Ian Tyler Applebaum • Instructor • Temple University CIS

Figure 8-3 An example of implementation inheritance. The left column depicts a questionable

implementation of `MySet` using implementation inheritance. The right column depicts an improved implementation using delegation (UML class diagram and Java).

Delegation and Implementation Inheritance



```java
/* Implementation of MySet using inheritance */
class MySet extends Hashtable {
    /* Constructor omitted */
    MySet() {
    }
    void put(Object element) {
        if (!containsKey(element)) {
            put(element, this);
        }
    }
    boolean containsValue (Object element){
        return containsKey(element);
    }
    /* Other methods omitted */
}
```

Delegation

constructor. This addresses both issues:

is private. Hence, we can change the internal

- The only significant change is the private field a List) without impacting any clients of `MySet`.

table and its initialization in the MySet()

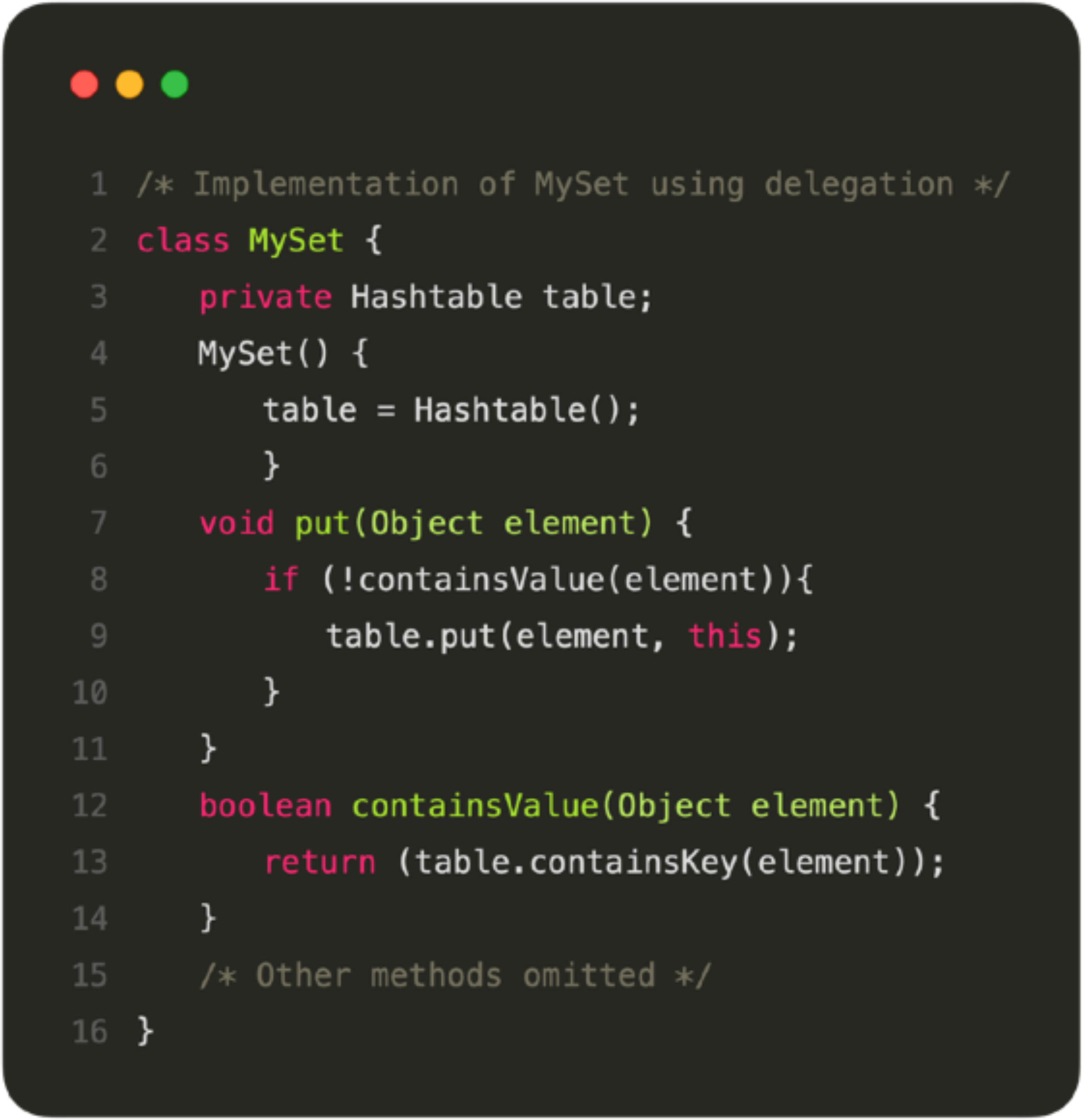
`Hashtable` in client code. Previously using

- ***Subtyping***: `MySet` doesn’t inherit from representation of `MySet` to another class (e.g.,

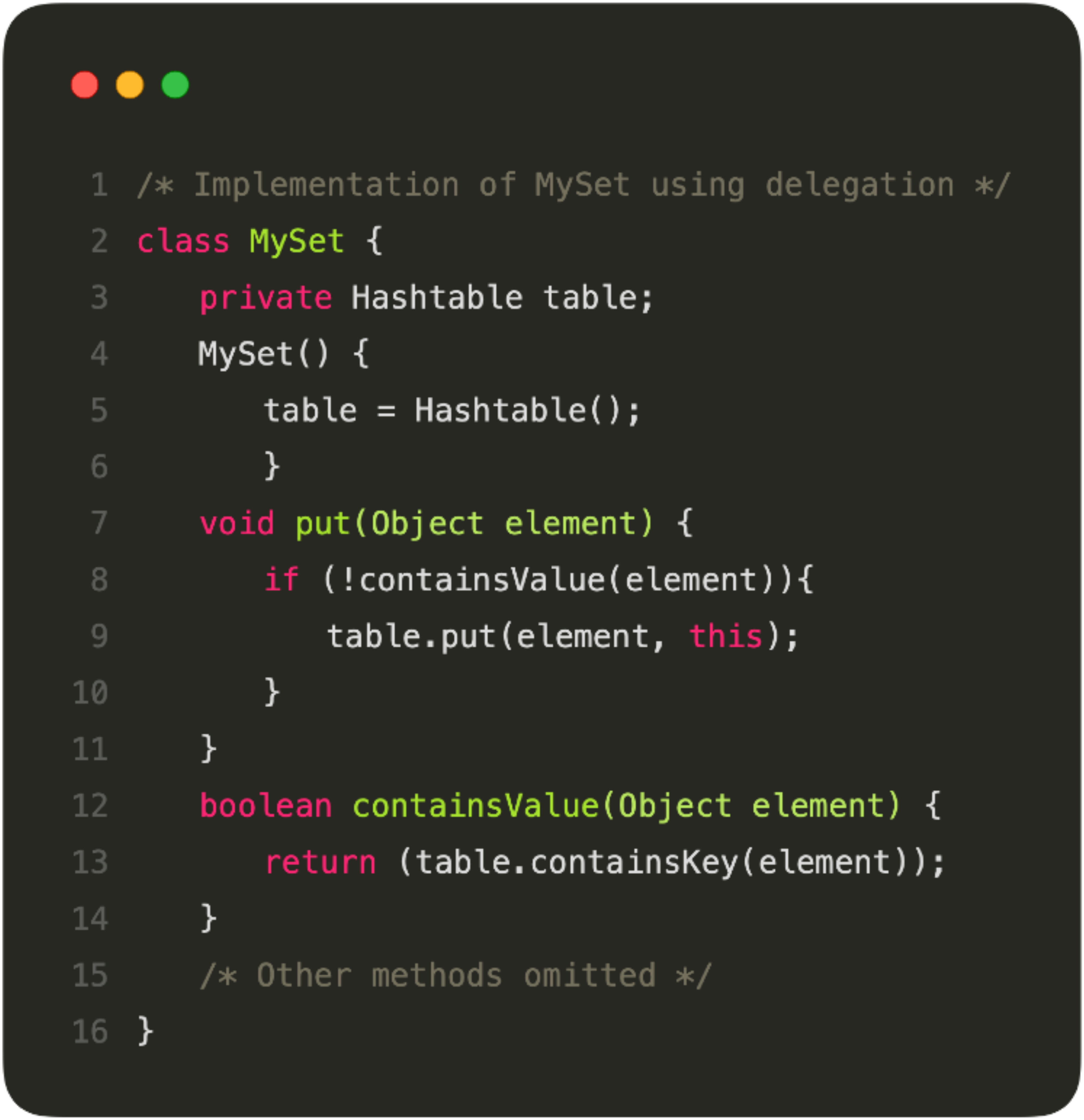
- ***Extensibility*:** The `MySet` on the right column

`Hashtable`, so it can’t be substituted for a method in its interface and the new field table does not include the `containsKey()`

Hashtables still works as expected.



```
/* Implementation of MySet using delegation */
class MySet {
    private Hashtable table;
    MySet() {
        table = Hashtable();
        }
    void put(Object element) {
        if (!containsValue(element)){
            table.put(element, this);
        }
    }
    boolean containsValue(Object element) {
        return (table.containsKey(element));
    }
    /* Other methods omitted */
}
```



```
/* Implementation of MySet using delegation */
class MySet {
    private Hashtable table;
    MySet() {
        table = Hashtable();
        }
    void put(Object element) {
        if (!containsValue(element)){
            table.put(element, this);
        }
    }
    boolean containsValue(Object element) {
        return (table.containsKey(element));
    }
    /* Other methods omitted */
}
```