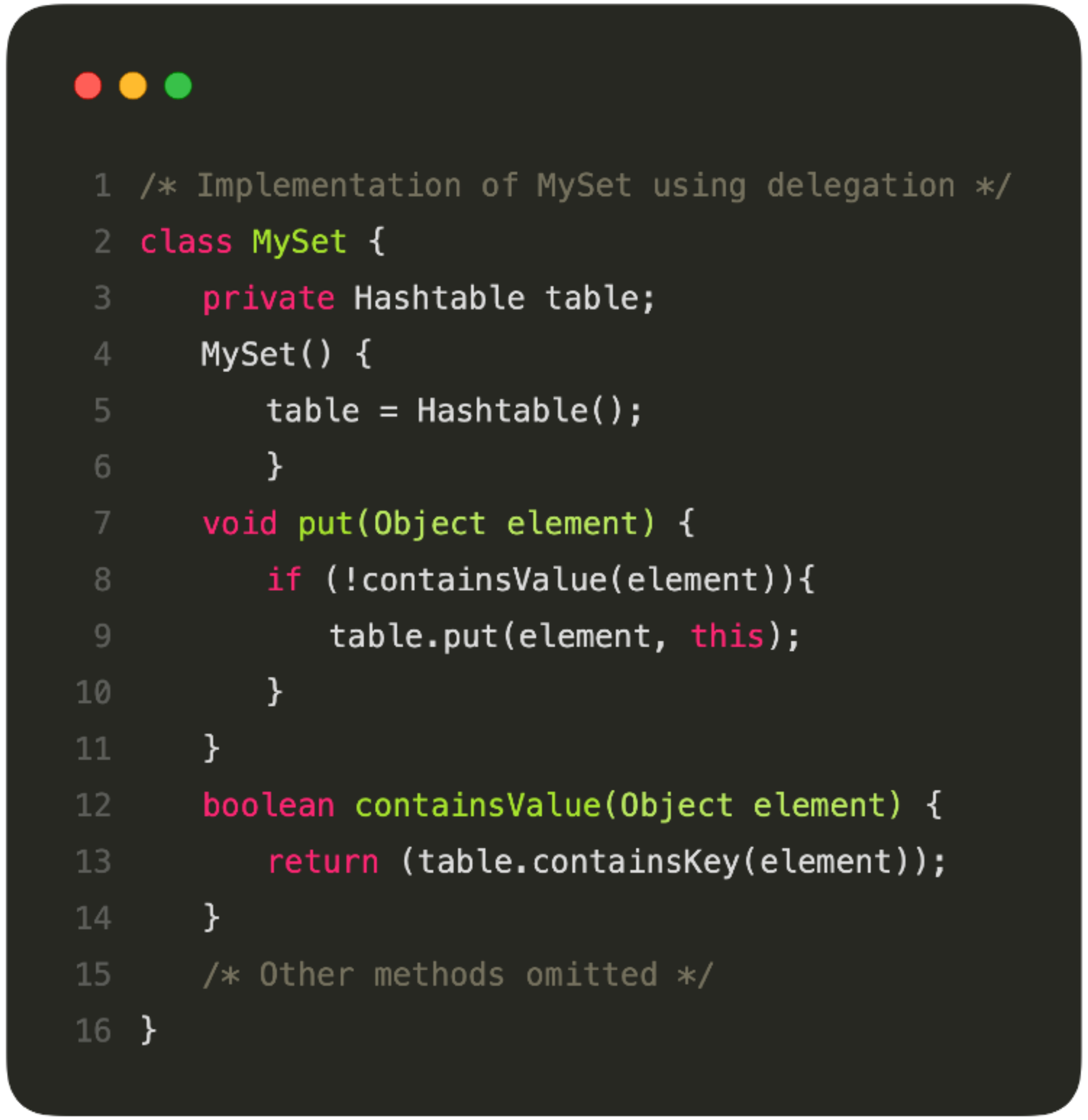
Delegation

- The only significant change is the private field table and its initialization in the MySet() constructor. This addresses both issues:

- ***Extensibility***: The `MySet` on the right column does not include the `containsKey()` method in its interface and the new field table is private. Hence, we can change the internal representation of `MySet` to another class (e.g., a List) without impacting any clients of `MySet`.

- ***Subtyping***: `MySet` doesn't inherit from `Hashtable`, so it can't be substituted for a `Hashtable` in client code. Previously using 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 */
}
```

Delegation

- **Delegation** is the alternative to implementation inheritance that should be used when reuse is desired.

- A class is said to delegate to another class if it implements an operation by resending a message to another class.

- Delegation makes explicit the dependencies between the reused class and the new class.