Exercise 11: Syntax Error Handling with Specific Anchors

Write parser methods for the following grammar and implement error handling with specific anchors (Section 3.4.3).

```
Program      = "program" ident {Declaration} "{" {Statement} "}".
Declaration  = ("public" | "private") Type ident ";".
Type         = ident ["[" "]"].
```

Since public and private are keywords that only occur at the start of a declaration, the start of the declaration is a good synchronization point. You do not need to implement the parser method for Statement.

Solution

Since the synchronization point is in Declaration, it is important to ensure that the iteration {Declaration} is not left prematurely (e.g., in case public or private are mistyped at the start of a declaration) so that the synchronization point is reached. The iteration should only be left with a valid successor of the iteration or eof. We assume that there is also a synchronization point in Statement. Thus, the statement sequence is treated in the same way as the declaration sequence.

```
static void Program() {
    check(program);
    check(ident);
    while (sym != lbrace && sym != eof) {
        Declaration();
    }
    check(lbrace);
    while (sym != rbrace && sym != eof) {
        Statement();
    }
    check(rbrace);
}

static void Declaration() {
    if (sym != public_ && sym != private_ && sym != lbrace) {  // synchronization point
        error("Declaration expected");
        do scan(); while (sym != public_ && sym != private_ && sym != lbrace && sym != eof);
        errDist = 0;
    }
    if (sym == lbrace || sym == eof) break;
    if (sym == public_) scan();
    else if (sym == private_) scan();
    // no error branch, because synchronization ended with public, private, lbrace or eof
    Type();
    check(ident);
    check(semicolon);
}

static void Type() {
    check(ident);
    if (sym == lbrack) {
        scan();
        check(rbrack);
    }
}
```