Aufgabe 4: Rekursiver Abstieg (3)

Schreiben Sie Parsermethoden für folgende Produktionen:

```
MethodCall  = ident "(" [ Parameters ] ")" ";".
Parameters  = Par {"," Par}.
Par         = ["out" | "ref"] ident.
```

Lösung

Man muss hier berücksichtigen, dass Par mit einem löschbaren Konstrukt (["out" | "ref"]) beginnt. Die terminalen Anfänge von Par (und somit auch von Parameters) sind daher nicht nur out und ref, sondern auch ident.

```
static void MethodCall() {
    check(ident);
    check(lpar);
    if (sym == out || sym == ref || sym == ident) Parameters();
    check(rpar);
    check(semicolon);
}

static void Parameters() {
    Par();
    while (sym == comma) {
        scan();
        Par();
    }
}

static void Par() {
    if (sym == out) scan();
    else if (sym == ref) scan();
    // no error branch
    check(ident);
}
```