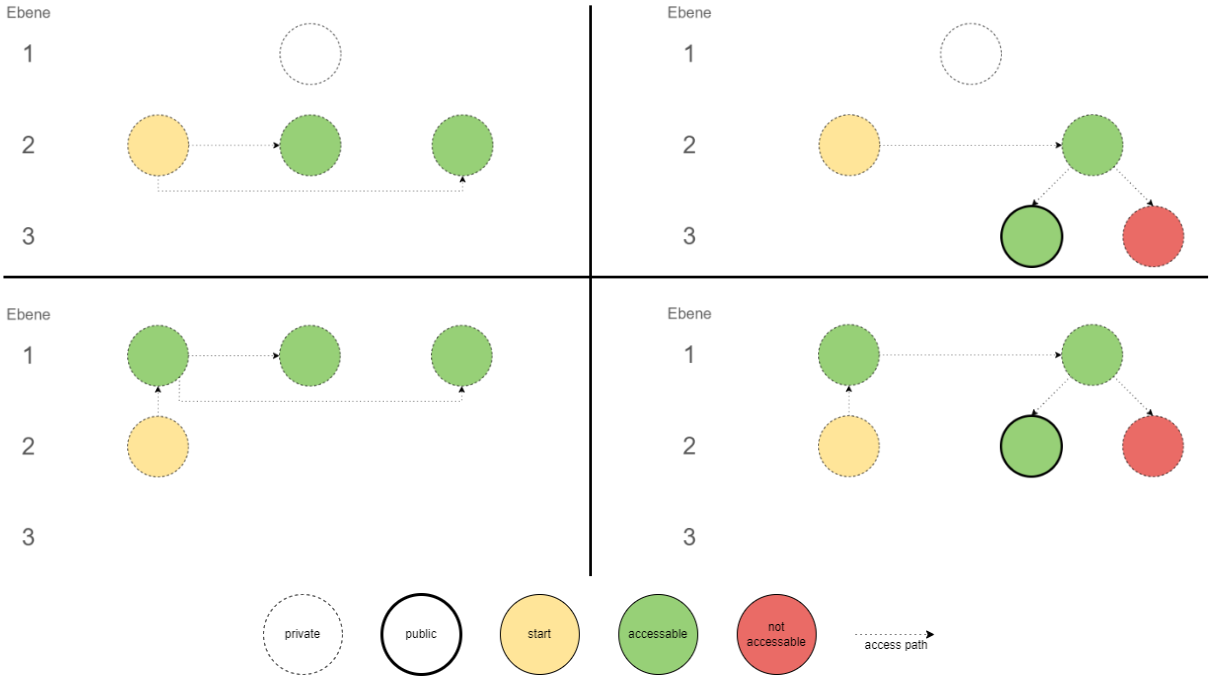
Cheat Sheet

Projekt Organisation

- Path: Ist der Pfad zu einem Item (z.B. Modul, Funktion usw.)
- Module: Enthält Items und dient der Codeorganisation (Analogie: Ordner)
- Crate: Ist entweder eine Bibliothek oder eine Executable
- Package: Enthält mindestens eine Crate und einen Bauplan für diese

Zugriffsrechte



Ownership Regeln

1. Jeder Wert hat einen Owner
2. Es gibt zu einem Zeitpunkt nur einen Owner
3. Wenn der Owner den Scope verlässt, wird der Wert fallen gelassen

Move

Der Wert von Variablen, die auf dem Heap liegen, werden bei Zuweisung zu einer neuen Variablen nicht kopiert, sondern gemoved:

```
let s1 = String::from("hello");
let s2 = s1;
println!("Value of s1 moved to s2: {}", s2);
```

➔ s1 nicht mehr verwendbar

Scope (Gültigkeit)

```
fn main() {                         // s noch nicht deklariert
        let s = String::from("hello");  // s gültig
}                                   // s nicht mehr im Scope
```

Borrow

```
fn main() {
        let s1 = String::from("hello");
        let len = calculate_length(&s1);        // borrows s1
        println!("{}", s1);
}

fn calculate_length(s: &String) -> usize {      // s ist eine Referenz
        s.len()
}                                               // s verlässt Scope, ist aber Referenz
                                                // -> nichts passiert
```

Geliehener Wert kann ohne das Ownership zu wechseln verwendet werden

Veränderbare Referenz

```
fn main() {
        let mut s = String::from("hello");
        change(&mut s);
}

fn change(some_str: &mut String) {
        some_str.push_str(", world!");
}
```

Referenz, Übergabeparameter und referenzierte Variable müssen veränderbar sein!

Regel: Maximal eine veränderbare **oder** eine beliebige Anzahl unveränderbare Referenzen