NULL

Nullable<Thing> x;

x = null

x = something

if(x is null)

```
if(x is null) {
    // x is NULL only
} else {
    // x is a Thing
}
```

Uninitialized Memory

1. always initialize — array values / field values — default value

2. force programmer to do it — compiler ↓ our way

Correct ⊃ provably correct

out-of-bounds memory access (buffer overflow)

arrays have size
Check the size

a[i] compiles as → if (i ≥ 0 && i < a.length) a[i] else error

(if (a.length < n+1) rerm
n = a.length - 1
for (i = 0; i < n; i++) ... a[i+1] ...

malloc / free

Ownership

- Transfer
- borrow

```
int baz() {
    int *a = malloc(...)
    :
    bar(a)
    :
    free(a)
}
```

automatically add free()