Primitives

`Byte`, `Int`, `Long`, `Float`, `Double`, `Boolean`, etc are classes, but:

- the compiler may optimize and use primitives instead

- for performance critical code:

- avoid boxing/unboxing - e.g. working with lists

- avoid nullable types (primitives cannot be null)

- use `IntArray`, `DoubleArray`, etc.

- **reusability, readability, maintainability** are *usually* more important than performance

Primitives

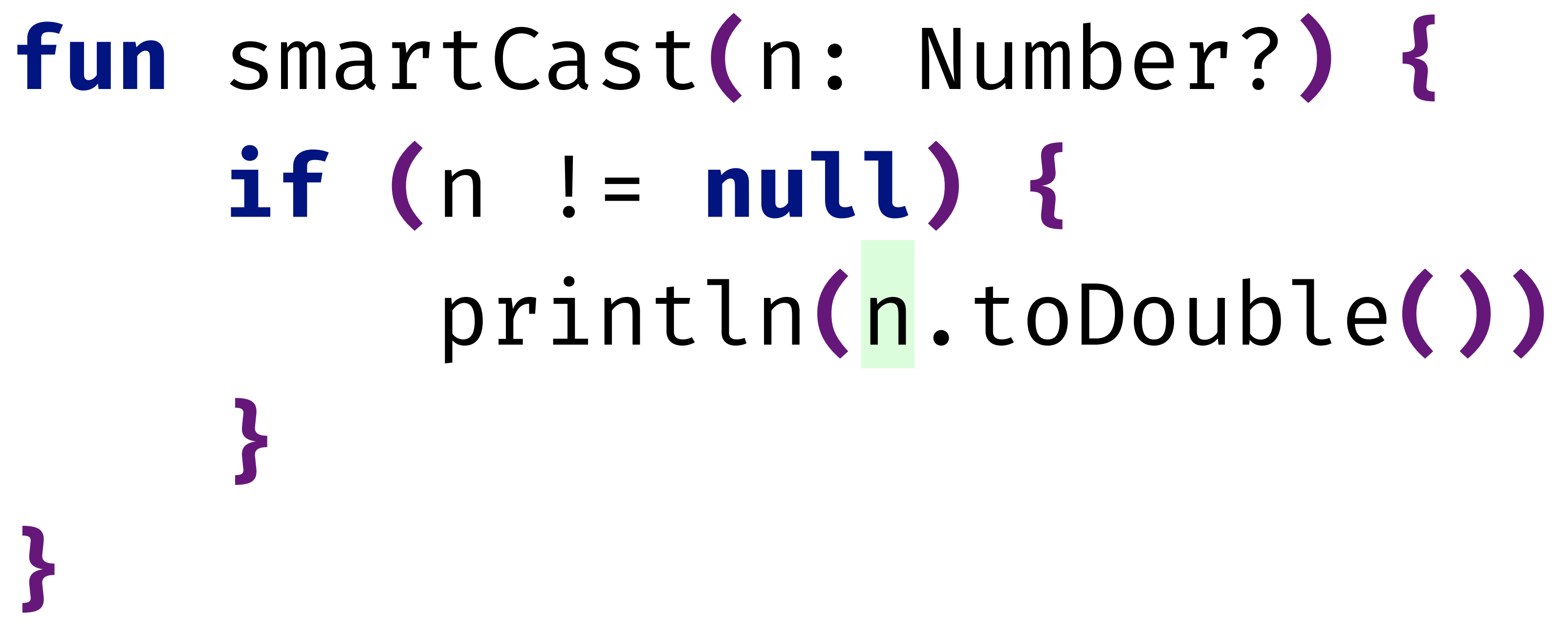
`Byte`, `Int`, `Long`, `Float`, `Double`, `Boolean`, etc are classes, but:

- the compiler may optimize and use primitives instead
- for performance critical code:
  - avoid boxing/unboxing - e.g. working with lists
  - avoid nullable types (primitives cannot be null)
  - use `IntArray`, `DoubleArray`, etc.

❗ **reusability, readability, maintainability** are *usually* more important than performance

Smart Cast

nullable to non nullable



```
fun smartCast(n: Number?) {
    if (n != null) {
        println(n.toDouble())
    }
}
```