Contracts

Contracts

Define guarantees to the compiler

Contract 1 – guarantees based on the return value

```kotlin
@OptIn(ExperimentalContracts::class)
fun containsJustDigits(str: String?): Boolean {
    contract {
        // small DSL for testing things about this function
        returns(true) implies (str != null)
    }
    return str?.all { it.isDigit() } ?: false
}
```

necessary annotation

opens contract DSL

condition implies compiler guarantee

```kotlin
if (containsJustDigits(maybeString)) {
    println("I can have the length: ${maybeString.length}")
}
```

compiler has extra info

Useful for type checking/flow typing

Contracts

Define guarantees to the compiler

Contract 2 – guarantees based on how lambdas are called

```
@OptIn(ExperimentalContracts::class)
fun <R: Resource, A> R.bracket(block: (R) -> A): A {
    contract {
        callsInPlace(block, InvocationKind.EXACTLY_ONCE)
    }

    this.open()
    try { return block(this) } finally { this.close() }
}
```

how many times a lambda can be called

```
resource.bracket {
    // this code runs just once
    result = it.getMotivation() // allowed, ONE-TIME assignment
}
```

Careful: fulfill your part of the contract!

Kotlin rocks