```
package largerNumber

fun max2(a: Int?, b: Int?): Int? { //típusdekralációk Int? iNT LEHET NULL!
    if (a != null && b != null) {
        return if (a > b) a else b
    }
    return null
}
/*if ha a nem gyenlő null és b nem egyenlő null akkor
a nagyobb b nél vagy mégse azaz b is lehet nagyobb a nál
különbön érték null
(null jelentése semmi nincs)
*/
fun main() {
    print("Add meg az első számot: ")
    val num1 = readlnOrNull()?.toIntOrNull() //?:0

    print("Add meg a második számot: ")
    val num2 = readlnOrNull()?.toIntOrNull() //?:0

    val result = max2(num1, num2)


    if (num1 != null && num2 != null) {
        println("A nagyobb szám: $result")
    }else {
        println("Kérlek adj meg egy értéket!")
    }
}
```

vagy ez jobb

```
package largerNumber

fun max(a: Int?, b: Int?): Int? { //típusdekralációk Int? lehet null!
    return if (a == null || b ==null) null else if (a > b) a else b
    /*return visszadja az értéket ha az a vagy a || b null akkor az érték
    null különbön ha a nagyobb b nél különben b nagyobb a-nál
    */

}

fun main() {
    print("Add meg az első számot: ")
    val num1 :Int? = readlnOrNull()?.toIntOrNull()

    print("Add meg a második számot: ")
    val num2 :Int? = readlnOrNull()?.toIntOrNull()

    val result :Int? = max(num1, num2)
```

```
    if (result != null) {
    println("A nagyobb szám: $result")
}else {
    println("Kérlek adj meg egy értéket!")
}
}
//ilyenkor lehet null is azaz ha nem adsz meg értéket nem ad hibát
//de önmagában az elvis operátor nem véd meg a nulltól?
```

és

a fun max függvény magyarázata

```
fun max(a :Int, b: Int) :Int//típusdekralációk
{return if(a > b) a else b}//a nagyobb b-nél külömben b
//return visszaadja az értéket után if() else
//egy egysoros if else

fun main() {
    val num1 = 47//első szám
    val num2 = 17//második szám

    val result :Int = max(num1, num2)
    println("The maximum (larger)of the numbers is: $result")
    println("A nagyobb szám: $result")

}
```

elemzés:

Szuper, Ati! Lássuk részletesen, mit csinál a három kódod, és hogy miért jók vagy esetleg mit lehet még javítani rajtuk.

---

1. `max2` függvény és hozzá tartozó `main`

```
fun max2(a: Int?, b: Int?): Int? {
    if (a != null && b != null) {
        return if (a > b) a else b
    }
    return null
}

fun main() {
    print("Add meg az első számot: ")
    val num1 = readlnOrNull()?.toIntOrNull()

    print("Add meg a második számot: ")
    val num2 = readlnOrNull()?.toIntOrNull()

    val result = max2(num1, num2)

    if (num1 != null && num2 != null) {
        println("A nagyobb szám: $result")
```

```
    } else {
        println("Kérlek adj meg egy értéket!")
    }
}
```

Elemzés:

- A `max2` függvény paraméterei nullable-k, azaz lehetnek `null` értékek.
- Ha **mindkét szám nem null**, akkor összehasonlítja őket, és visszaadja a nagyobbat.
- Ha **bármelyik null**, akkor `null`-t ad vissza.
- A `main`-ben szintén ellenőrzöd, hogy a két bemenet nem `null` mielőtt kiírod az eredményt.

Ez a kód jó, működőképes és jól kezeli a null értékeket.

---

2. `max` függvény és hozzá tartozó `main`

```
fun max(a: Int?, b: Int?): Int? {
    return if (a == null || b == null) null else if (a > b) a else b
}

fun main() {
    print("Add meg az első számot: ")
    val num1: Int? = readlnOrNull()?.toIntOrNull()

    print("Add meg a második számot: ")
    val num2: Int? = readlnOrNull()?.toIntOrNull()

    val result: Int? = max(num1, num2)
    if (result != null) {
        println("A nagyobb szám: $result")
    } else {
        println("Kérlek adj meg egy értéket!")
    }
}
```

Elemzés:

- Ez nagyon hasonló az elsőhöz, csak másképp van megfogalmazva a `max` függvényben a logika.
- Ha **bármelyik paraméter** `null`, akkor azonnal `null`-t ad vissza.
- Ellenkező esetben összehasonlítja a két számot, és a nagyobbat adja vissza.
- A `main`-ben már az eredményt (`result`) ellenőrzöd, hogy nem `null`-e, és annak megfelelően írsz ki üzenetet.

Ez is teljesen helyes, működik és a null kezelése is megfelelő.

---

3. Nem-nullable `max` és fix számok

```
fun max(a: Int, b: Int): Int {
    return if (a > b) a else b
}

fun main() {
    val num1 = 47
    val num2 = 17

    val result: Int = max(num1, num2)
    println("The maximum (larger) of the numbers is: $result")
    println("A nagyobb szám: $result")
}
```

Elemzés:

- Egyszerű, tiszta példa, ahol nem kell nullával foglalkozni, mert a számok fix értékűek, sosem lesznek `null`-ok.
- A `max` függvény egy egysoros `if-else`-sel megadja a nagyobb számot.
- Ez egy klasszikus, hibamentes példa, amit akár oktatásban is használnak.

Összefoglaló, ajánlások:

- Az **1. és 2. kód is helyes, csak más a megfogalmazásuk**. Mindkettő jól kezeli, ha a felhasználó nem ad meg számot (null érték).
- A **3. példa tiszta és egyszerű, de nem foglalkozik felhasználói inputtal, csak fix számokkal dolgozik**.
- Ha felhasználói bemenetet vársz, az első két verzió a megfelelő.
- Az `?:` (elvis operátor) nem védi meg automatikusan a `null`-tól, mert ha a változó `null`, akkor a hívás (pl. `max(num1, num2)`) nem fog működni, ha nem nullable a paraméter. Ezért kell a paramétereket nullable-ként definiálni vagy null ellenőrzést végezni.

Ha szeretnéd, átfésülhetem az egyik verziódat úgy, hogy még tisztább legyen, vagy hozzáadhatunk alapértelmezett értéket, pl.:

```
val num1 = readlnOrNull()?.toIntOrNull() ?: 0
```

így, ha nem ad meg értéket, 0 lesz az alapértelmezett.
