```
fun main() {

    while (true) {
        print("What's yout level of pain between 1 and 3 ? :")

        //val number: Int = readLine()!!.toInt()//!! nem lehet null
        val number: Int? = readlnOrNull()?.toIntOrNull()
        if (number == null) {
            println("Hiba kérlek egy számot adj meg")
            continue // újra kérdezi
        } else {

            when (number) {
                1 -> println("You will be okay.")
                2 -> println("Are you okay?")
                3 -> println("You are cool.")
                else -> {
                    println("Please choose a number between 1 and 3.")
                    continue // újra kérdezi
                }

            }
        }
        break//ha 1-3 között van kilép a ciklusból
    }
}
```

Természetesen, Ati! Lássuk lépésről lépésre, mit csinál a kódod:

---

1. Végtelen ciklus `while (true)`

```
while (true) {
```

- Ez azt jelenti, hogy a blokkod végtelenül ismétlődik, amíg be nem lép a `break` utasítás.
- Ez a ciklus addig fut, amíg a felhasználó nem ad meg érvényes bemenetet.

---

2. Kiírás a konzolra

```
print("What's yout level of pain between 1 and 3 ? :")
```

- Kiírja a kérdést, hogy a felhasználó adjon meg egy számot 1 és 3 között.

---

3. Bemenet beolvasása és átalakítása

```
val number: Int? = readlnOrNull()?.toIntOrNull()
```

- `readlnOrNull()` — beolvassa a konzolról a sort, de ha nincs bemenet (pl. Ctrl+D vagy üres sor), akkor `null`-t ad.
- `?.` — biztonságos hívás: ha az előző lépés eredménye nem `null`, akkor végrehajtja a `toIntOrNull()` hívást.
- `toIntOrNull()` — megpróbálja átalakítani a beolvasott stringet `Int`-té; ha ez nem sikerül (például nem számot írt be), akkor `null`-t ad vissza.
- Így a `number` változó lehet:
  - egy egész szám,
  - vagy `null` (ha a beolvasás vagy a konvertálás nem sikerült).

---

4. Null érték ellenőrzése

```
if (number == null) {
    println("Hiba kérlek egy számot adj meg")
    continue // újra kérdezi
}
```

- Ha nem kaptunk érvényes számot (vagy nem érkezett semmi bemenet), kiírja a hibát.
- A `continue` miatt a ciklus az elejére ugrik, azaz újra megkérdezi a számot.

---

5. Érték ellenőrzése `when` szerkezettel

```
when (number) {
    1 -> println("You will be okay.")
    2 -> println("Are you okay?")
    3 -> println("You are cool.")
    else -> {
        println("Please choose a number between 1 and 3.")
        continue // újra kérdezi
    }
}
```

- A `when` a Kotlin egyik vezérlési szerkezete, ami egyfajta "switch"-ként működik.
- Ha a `number` értéke 1, 2 vagy 3, akkor kiírja a hozzá tartozó üzenetet.
- Ha más számot adott meg a felhasználó, kiírja, hogy csak 1 és 3 közötti számot válasszon, és `continue`-val újra kéri a bemenetet.

---

6. Kilépés a ciklusból

```
break
```

- Ha a bemenet érvényes volt (1, 2 vagy 3), a program itt kilép a végtelen ciklusból.
- Ezzel a program befejeződik.

Összegzés:

- A program beolvassa a felhasználó bemenetét.
- Ellenőrzi, hogy a bemenet nem üres, és egész szám-e.
- Ha nem, újra kéri a bemenetet.
- Ha számot adott meg, megnézi, hogy az 1, 2 vagy 3-e.
- Ha nem az, akkor figyelmeztet és újra kéri a bemenetet.
- Ha igen, kiírja a megfelelő üzenetet, majd befejezi a futást.

Lentebb a Kód: az if-es verzió.....

`readlnOrNull()`

- Ez a Kotlin **beépített függvénye**, ami beolvas egy sort a szabványos bemenetről (például a billentyűzetről).
- Ha **sikerül beolvasni egy sort**, akkor azt a String-et adja vissza.
- Ha **nem sikerül (például EOF vagy nincs bemenet)**, akkor `null`-t ad vissza.
- Ez az előnye a `readLine()`-hoz képest, ami nem nullable, de kicsit régebbi, míg a `readlnOrNull()` újabb és biztonságosabb.

`toIntOrNull()`

- Ez a String osztály egy függvénye, ami megpróbálja a szöveget egész számmá alakítani.
- Ha **a String tényleg egy érvényes egész szám**, akkor visszaadja azt az Int értéket.
- Ha **nem lehet átalakítani számra** (például a szövegben betűk vannak vagy üres a bemenet), akkor `null`-t ad vissza.
- Így biztonságosan tudsz számot olvasni anélkül, hogy kivétel keletkezne.

Miért használod együtt?

```
val number: Int? = readlnOrNull()?.toIntOrNull()
```

- Először megpróbálja beolvasni a felhasználó sorát (String vagy null).
- Ha nem null, akkor megpróbálja egész számmá konvertálni.
- Ha az is sikerül, akkor egy `Int` értéket kapsz.
- Ha bármi hiba van (nem olvas be semmit vagy nem szám), akkor `number` értéke `null` lesz.

Ezért tudod utána leellenőrizni, hogy `number == null` és kezelni a hibát biztonságosan.

Összefoglalva

- `readlnOrNull()` = biztonságos, nullable szövegbeolvasás
- `toIntOrNull()` = biztonságos, nullable egész szám konvertálás szövegből

Ez a kettő együtt lehetővé teszi, hogy ne ess hibába, ha a felhasználó nem számot ír be.

```
fun main() {

    while (true) {
        print("What's yout level of pain between 1 and 3 ? :")

        //val number: Int = readLine()!!.toInt()//!! nem lehet null
        val number: Int? = readlnOrNull()?.toIntOrNull()
        if (number == null) {
            println("Hiba kérlek egy számot adj meg")
            continue // újra kérdezi
        } else {

            if (number == 1) {
                println("you will be okay")
            } else if (number == 2) {
                println("Are you okay")
            } else if (number == 3) {
                println("You are so cool")
            } else {
                println("Please you choose to 1-3")
                continue // újra kérdezi
            }
        }
        break//ha 1-3 között van kilép a ciklusból
    }
}
```

Mit csinál ez a program?

Ez egy egyszerű konzolos program, ami megkérdezi a felhasználótól, hogy mennyire fájdalmas neki egy dolog (1 és 3 között kell választani), és a megadott érték alapján különböző válaszokat ad vissza.

Részletes elemzés

```
while (true) {
```

- Ez egy végtelen ciklus, ami addig fut, amíg be nem lép a `break` utasítás.
- Azért használod, hogy újra és újra kérje a felhasználótól a bemenetet, amíg érvényes választ nem ad meg.

```
print("What's yout level of pain between 1 and 3 ? :")
```

- Kiír egy kérdést a konzolra, hogy a felhasználó adjon meg egy számot.

```
val number: Int? = readlnOrNull()?.toIntOrNull()
```

- `readlnOrNull()` beolvassa a sor bemenetét, vagy `null`-t ad vissza, ha nincs semmi.
- A `?.` operátorral csak akkor hívja meg a `toIntOrNull()` függvényt, ha nem `null` a beolvasott string.
- `toIntOrNull()` megpróbálja a szöveget egész számmá konvertálni, ha ez nem lehetséges, `null`-t ad vissza.
- Így a `number` változó típusa `Int?`, azaz lehet egész szám vagy `null`.

```
if (number == null) {
    println("Hiba kérlek egy számot adj meg")
    continue // újra kérdezi
}
```

- Ha a beolvasott érték nem szám (vagy nincs semmi), kiír egy hibaüzenetet.
- A `continue` miatt a ciklus újraindul, azaz újra kérdezi a számot.

```
else {
    if (number == 1) {
        println("you will be okay")
    } else if (number == 2) {
        println("Are you okay")
    } else if (number == 3) {
        println("You are so cool")
    } else {
        println("Please you choose to 1-3")
        continue // újra kérdezi
    }
}
```

- Ha a `number` nem `null`, megnézi, hogy pontosan 1, 2 vagy 3-e.
- Minden értékhez tartozik egy válasz (kiírás).
- Ha a szám nem 1, 2 vagy 3, akkor figyelmeztet, hogy csak 1 és 3 közötti számot válasszon, majd a `continue` újraindítja a ciklust.

```
break
```

- Ha a szám érvényes (1, 2 vagy 3), akkor kilép a ciklusból, véget ér a program.

Összefoglalva:

1. A program bekér egy számot 1 és 3 között.
2. Ha nem számot ad meg, vagy nem az elvárt tartományban van, újra kérdez.
3. Ha megfelelő számot ad meg, kiír egy személyre szabott választ és befejezi a futást.

Ha szeretnéd, segítek tovább fejleszteni vagy optimalizálni is!