Jak pobierać dane z konsoli - `readLine()`

TIP

- `readLine()` czyta tekst z konsoli, czyli zwraca `String?`
- konwersja: `.toInt()`, `.toDouble()`, itp.
- bezpieczna konwersja: `.toIntOrNull()`, `.toDoubleOrNull()`, itp.

Podaj swoje imię - pobieramy `String?`

```
fun main() {
    println("Podaj swoje imię:")
    val name: String? = readLine()  // odczytuje cały wiersz jako
String?
    println("Cześć, $name!")
}
```

- zmienna `name` jest typu `String?`, bo taki typ zwraca metoda `readLine()`
- zmienna `name` może przyjąć dowolny ciąg tekstu (type `String`) lub `null` (taki jest sens znaku `?`)

Podaj swój wiek - konwersja do `Int`

Niebezpieczna konwersja

- `readLine` zawsze zwraca typ `String?`
- my chcemy informację o wieku przechowywać jako liczbę całkowitą `Int`

```
fun main() {
```

```
    println("Podaj swój wiek:")
    val age = readLine()!!.toInt()  // zamiana na Int
    println("Za sto lat będziesz miał  ${age+100} lat.")
}
```

`!!` ***NOT-NULL ASSERTION OPERATOR*** **(OPERATOR WYMUSZENIA NIE-NULL)**

- Kotlin rozróżnia typy
  - *nullable* (`String?`, `Int?`) i
  - *non-nullable* (`String`, `Int`).
- Jeśli masz zmienną typu *nullable*, nie możesz jej bezpośrednio używać tam, gdzie oczekiwany jest typ *non-null*.
- Operator `!!` mówi do kompilatora:

> **NOTE**
>
> „Jestem absolutnie pewien, że ta zmienna nie jest `null`. Traktuj ją jak zwykły (*non-null*) typ”

- Jak działa?
  - Jeśli wartość nie jest `null`, to działa poprawnie.
  - Jeśli wartość jest null, to wyrzuca wyjątek `NullPointerException` (*NPE*).

:::

Bezpieczna konwersja

safe call operator

```
fun main() {
    println("Podaj liczbę:")
    val number = readLine()?.toIntOrNull()

    if (number != null) {
        println("Podałeś liczbę: $number, a ja ją powiększę o 99
${number+99}")
```

```
    } else {
        println("To nie jest poprawna liczba!")
    }
}
```

`?.` *SAFE CALL OPERATOR* (OPERATOR BEZPIECZNEGO WYWOŁANIA)

Co robi?

- Pozwala bezpiecznie odwołać się do właściwości lub metody na obiekcie, który może być `null`.
- Jeśli obiekt nie jest `null`, to wywołuje właściwość/metodę.
- Jeśli obiekt jest `null`, to zwraca `null`, zamiast wyrzucać wyjątek.
- prosty przykład

```
val text: String? = null

val length = text?.length   // zamiast NullPointerException ->
wynik = null

println(length) // wypisze: null
```

Elvis operator `?:`

Co robi?

- Elvis operator mówi: „Jeśli coś jest `null`, to użyj wartości domyślnej”.

```
val text: String? = null

val length = text?.length ?: 0
println(length) // 0
```

- przykład z pobieraniem danych z konsoli

```
fun main() {
    println("Podaj swoje imię:")
```

```
    val name = readLine() ?: "Anonim"
    println("Witaj, $name!")
}
```

- wyjaśnienie:
  - jeżeli wpiszesz imie, to zostanie ono wyświetlone w konsoli
  - jeżeli wciśniesz `Enter`, to w konsoli pojawi się napis `Witaj, !`, ponieważ `Enter` wprowadza pusty string
  - jeżeli wciśniesz `Ctr+d`, to funkcja `readLine()` zwróci `null`, więc pojawi się napis `Witaj, Anonim!`

Pobieranie liczb zmiennoprzecinowych

```
val pi = readLine()!!.toDouble()                          //
niebezpieczna konwersja
val safePi = readLine()?.toDoubleOrNull()                 // bezpieczna
konwersja
 val superSafePi = readLine()?.toDoubleOrNull() ?: 0      // super
bezpieczna konwersja
```

Pobieranie pojedynczego znaku

```
val letter = readLine()!!.first()           // pierwszy znak
wprowadzonego tekstu // niebezpieczna konwersja
val safeLetter = readLine()?.firstOrNull()  // null jeśli puste
// bezpieczna konwersja
```

Edit this page