```
fun main() {
    val bestFriends: ArrayList<String?> = arrayListOf("Ryan", "Emma")
    while (true) {
        println("What is your BestFriends?")

        val bestfriendInput: String? = readlnOrNull()
        if (bestfriendInput == null) {
            //println(bestFriends.joinToString("\n"))
        } else if (bestfriendInput.lowercase() == "exit") {
            break
        } else if (bestfriendInput.isBlank()){//Enter probléma uj sor amit az ide
add hozzá megoldva
            //isBlank() azt ellenőrzi, hogy a felhasználó csak üresen vagy
szóközökkel nyomott-e
            println(bestFriends.joinToString("\n"))
            continue
        }
        else {
            bestFriends.add(bestfriendInput)
            println(bestFriends.joinToString("\n"))
        }

        /*for (friend: String? in bestFriends) { //ezzel is soronként irjuk ki
            println(friend)
        }*/
    }
}
```

```
fun main() {
    val bestFriends: ArrayList<String?> = arrayListOf("Ryan", "Emma")
    while (true) {
        println("What is your BestFriends?")

        val bestfriendInput: String? = readlnOrNull()
        if (bestfriendInput == null) {

        } else if (bestfriendInput.lowercase() == "exit") {
            break
        }  else if (bestfriendInput.isBlank()){ //isNullorBlank
        } else {
            bestFriends.add(bestfriendInput)
        }
     for (friend: String? in bestFriends) { //ezzel is soronként irjuk ki
            println(friend)
        }
    }
}
```

```
fun main() {
    val s: String? = null
    println(s!!.length)  // Itt fog hibázni futás közben!
}
//szándékor nullhiba
```

Összefoglalás:

---

1. Első kód – Felhasználói bevitel kezelése, üres sorokkal (isBlank)

```
fun main() {
    val bestFriends: ArrayList<String?> = arrayListOf("Ryan", "Emma")
    while (true) {
        println("What is your BestFriends?")

        val bestfriendInput: String? = readlnOrNull()
        if (bestfriendInput == null) {
            // semmit nem csinál, ha EOF vagy hiba (readlnOrNull null-t ad)
        } else if (bestfriendInput.lowercase() == "exit") {
            break  // Kilép a ciklusból "exit" beírására
        } else if (bestfriendInput.isBlank()) {
            // Ha csak üres vagy szóközöket ír be a felhasználó,
            // kiírja a barátok listáját új sorokkal elválasztva, majd folytatja
a következő körrel
            println(bestFriends.joinToString("\n"))
            continue
        } else {
            bestFriends.add(bestfriendInput) // Új barát hozzáadása a listához
            println(bestFriends.joinToString("\n")) // Kiírja az egész listát
soronként
        }
    }
}
```

Mit csinál ez?

- A `readlnOrNull()` beolvassa a sort. Ha `null` (például EOF), nem csinál semmit.
- Ha "exit" a bevitel, kilép.
- Ha az input üres vagy csak szóköz, akkor kiírja a barátok listáját (soronként), de nem ad hozzá semmit.
- Egyébként hozzáadja a bevitt nevet és kiírja az egész listát.
- Az `isBlank()` segít kezelni az "üres" vagy "csak szóköz" eseteket, hogy ne legyen fölösleges vagy zavaró viselkedés.

---

2. Második kód – Barátok listájának folyamatos kiírása

```
fun main() {
    val bestFriends: ArrayList<String?> = arrayListOf("Ryan", "Emma")
    while (true) {
        println("What is your BestFriends?")

        val bestfriendInput: String? = readlnOrNull()
        if (bestfriendInput == null) {
            // semmi, ha nincs input
        } else if (bestfriendInput.lowercase() == "exit") {
            break
        }  else if (bestfriendInput.isBlank()){
            // üres sor esetén semmi nem történik
        } else {
            bestFriends.add(bestfriendInput)
        }
        for (friend: String? in bestFriends) {
            println(friend)  // Soronként írja ki az összes barátot
        }
    }
}
```

Mit csinál ez?

- Ez is hasonló a fentihez, csak itt nincs kiírás a beírás után, csak a `for` ciklus soronként írja ki az egész listát minden input után.
- Ha a felhasználó üres sort üt be (csak enter vagy szóköz+enter), akkor nem történik semmi új hozzáadás.
- Ha `null` jön, nem csinál semmit.
- Ha "exit", kilép.
- Ez a megoldás kicsit “folyamatosan frissíti” a teljes barátlistát, minden input után újra kiírva.

---

3. Harmadik kód – Szándékos null pointer hiba

```
fun main() {
    val s: String? = null
    println(s!!.length)  // Itt fog hibázni futás közben!
}
```

Mit csinál ez?

- A `s` nullable String, ami null értéket kap.
- A `!!` operátor azt jelenti: “Nem lehet null, ha mégis null, akkor dobj NullPointerException-t.”
- Mivel `s` valóban null, ezért futás közben **NullPointerException** lesz, és a program hibaüzenettel leáll.
- Ez egy szándékos példa arra, hogyan lehet nullra való hivatkozásnál hibát okozni Kotlinban.

Összefoglalás

- Az első két kódban a **felhasználói input kezelés** a cél, különösen az üres vagy csak szóközös sorok figyelembevétele, hogy ne kerüljön fölösleges adat a listába, és szép legyen a kiírás.
- Az `isBlank()` megakadályozza, hogy csak "üres sor" bekerüljön a listába.
- A második kódban a lista minden lépés után soronként kiírásra kerül a `for` ciklussal.
- A harmadik kód pedig egy egyszerű példa a **null érték miatti futási hibára** Kotlinban, amikor `!!`-t használunk egy null értéken.