Exception hibakezelés Kotlinban Példák:

```
package exception

import java.io.File
import java.io.FileReader
import java.io.IOException

fun main() {
    try {
        val file = File("src/exception/example.txt")
        // val file = File("src/example.txt") //hibat dodna a rendszer nem
találja a megdot fájlt
       val reader = FileReader(file)
        println("File read sucessfully.")
        reader.close()
    }catch (e: IOException){
        println("Error reading the file: ${e.message}")//e.message automatikusan
kiirja a hibát
    }
}
```

HandlingExceptions.kt

```
package exception

fun main(){

    try {
        val result = 10/0 //Hibát dob mert nem lehet nulla
        //a hiba kimenete Error : / by zero
        println("Result : $result")
    }catch (e: Exception){
        println("Error : ${e.message}")
    }
}
```

HandlingUncheckedException.kt

FinallyBlock.kt (with Finally block for Try&Catch)

```
package exception

import java.io.File
import java.io.FileReader
import java.io.IOException

fun main() {
    var fileReader : FileReader? = null
    try {
```

```
        val file = File("src/exception/example.txt")
        fileReader = FileReader(file)
        println("File read is sucessfully.")
    }catch (e: IOException) {
        println("Error readin the file : ${e.message}")
    }finally {
        fileReader?.close()
        println("Cleanup operations in the finally block.")
    }
}
```

elemzés:

---

1. Példa: `Exception` kezelés fájlolvasásnál (`IOException` kezelése)

```
package exception

import java.io.File
import java.io.FileReader
import java.io.IOException

fun main() {
    try {
        val file = File("src/exception/example.txt")
        val reader = FileReader(file)
        println("File read sucessfully.")
        reader.close()
    } catch (e: IOException) {
        println("Error reading the file: ${e.message}")
    }
}
```

Elemzés:

- **Mi történik?**
  Egy adott elérési útvonalon próbál megnyitni egy fájlt és beolvasni (itt csak létrehozza a `FileReader` objektumot).
- **Milyen kivételt kezel??**
  Az `IOException`-t, ami általánosan fájlkezelési hibáknál fordulhat elő (például: fájl nem létezik, nincs hozzáférés stb.).
- **A `try` blokkban:**
  Minden kódot, ami potenciálisan kivételt dobhat, ide teszünk.
- **A `catch` blokk:**
  Itt kezeljük az esetleges hibát. A `e.message` kiírja a hiba rövid leírását.
- **Záró megjegyzés:**
  Itt nem biztos, hogy a `reader.close()` mindig lefut, ha kivétel történik a `FileReader` létrehozásakor (nem biztos, hogy példány jön létre, de `close()`-t hívjuk). Ezért jobb a `finally` blokk, amit a harmadik példában is látunk.

---

2. Példa: Általános kivétel (`Exception`) kezelése, osztás nullával (`ArithmeticException`)

```
package exception

fun main() {
    try {
        val result = 10 / 0
        println("Result : $result")
    } catch (e: Exception) {
        println("Error : ${e.message}")
    }
}
```

Elemzés:

- **Mi történik?**
  Nullával osztás miatt futás közben kivétel keletkezik: `ArithmeticException` (ami az `Exception` leszármazottja).
- **Milyen kivételt kezel?**
  Általánosan `Exception`-t kezel, ami lefedi az összes kivételt, ami az `Exception` osztályból származik.
- **Mit csinál a `catch`?**
  Kiírja a hibaüzenetet (`/ by zero`).
- **Megjegyzés:**
  Érdemes lehet konkrétabban kezelni, pl. `ArithmeticException`-nel, ha külön akarod kezelni az osztás nullával hibát.

---

3. Példa: `try-catch-finally` blokk fájlolvasásnál

```
package exception

import java.io.File
import java.io.FileReader
import java.io.IOException

fun main() {
    var fileReader: FileReader? = null
    try {
        val file = File("src/exception/example.txt")
        fileReader = FileReader(file)
        println("File read is sucessfully.")
    } catch (e: IOException) {
        println("Error readin the file : ${e.message}")
    } finally {
        fileReader?.close()
        println("Cleanup operations in the finally block.")
    }
}
```

Elemzés:

- `try` **blokk:**
  Fájl megnyitása és olvasása (itt csak megnyitja a `FileReader`-t).
- `catch` **blokk:**
  Kezeli a fájlolvasás során esetleg fellépő `IOException`-t.
- `finally` **blokk:**
  Fontos: Itt zárjuk le a `fileReader`-t, ha az nem `null`.
  Ez biztosítja, hogy függetlenül attól, hogy sikerült-e a fájlolvasás vagy nem, a források felszabadulnak (close).
- **Megjegyzés:**
  A `finally` blokk mindig lefut, akár a `try` sikeres volt, akár kivétel keletkezett.

Összefoglaló:

- `try` **blokkban** van a potenciálisan hibát okozó kód.
- `catch` **blokkban** kezeljük a kivételt, amit meg akarunk fogni.
- `finally` **blokkban** olyan műveleteket helyezünk, amelyeknek mindenképp le kell futniuk (pl. erőforrások lezárása), akár volt hiba, akár nem.
- Ha nem adsz meg `finally` blokkot, a hibakezelés akkor is működik, de erőforrásokat könnyebben hagyhatsz nyitva.
- Érdemes a lehető legkonkrétabb kivételt kezelni (pl. `IOException`, `ArithmeticException`), mert az általános `Exception` elfedhet más hibákat, amiket máshogy kéne kezelni.
