**Kivételkezelés**

Kivétel: a programvégrehajtás során felmerülő esemény, amely megzavarja a normál programműködést.

A hibakezelés egy formája, amelyben elválasztható a hibalekezelő kódrészlet a normál működés kódjától, valamint jelezhetjük a kódunkban előforduló hibákat és előírhatjuk kezelésüket a kódunkat használó kód számára. A hagyományos hibakezelés hiányosságait küszöböli ki.

Hagyományos hibakezelés: Keverednek a funkcionális és a hibakezelő kódrészek.

Ha a metódus eredménye hiba, ezt speciális visszatérési értékkel vagy hiba paraméterrel jelezzük. Nem egységes. Nehéz a módosítás, rontja az olvashatóságot.

Kivételek: Kivétel (exception) létrejöhet egy végrehajtás során előforduló hiba hatására (I/O hiba, tömbindex túllépés, stb.).

Saját magunk is létrehozhatunk kivételeket (pl. nullával osztás, negatív életkor, intervallum felsőhatára kisebb mint az alsóhatár, stb.

) Amikor egy kivétel létrejön (kiváltódik), azt mondjuk, hogy megdobódott a kivétel (throwing an exception). A kivétel lekezelését a kivétel elkapásának nevezzük (catching an exception). Java-ban a kivételek objektumok, amik a felmerült kivétel eseménnyel kapcsolatban az alábbi információkat tartalmazzák:   
 – hiba típusa

– a program állapota a kivétel bekövetkezésekor.

Létezik számos már definiált kivétel osztály. Lehet saját kivételosztályokat létrehozni. A kivétel típusa nagy szerepet játszik az elkapás mechanizmusában. Egy lekezelő kódrészletnél meg kell adni, hogy az milyen típusú kivételt tud elkapni. A kivétel osztályok a Throwable osztály leszármazottai.

Kötelezően kezelendő kivételek (checked exception): olyan kivételek, amelyeket helyes programozással sem lehet elkerülni, mert a környezeten múlik (létezik-e a file) vagy felhasználói inputon (érvényes értéket adott-e meg az „életkornak”). Ilyenek pl. az I/O műveletek. Ilyen kivételekkel olyankor találkozunk, amikor olyan metódust hívunk, amelynél jelezve van, hogy megdobhat kivételt. Ilyenkor a fordító ellenőrzi, hogy ezeket lekezeljük-e. Pl. a java.io.FileInputStream osztály read metódusa public int read() throws IOException Ha használom a metódust, köteles vagyok kezelni a kivételt. Egyébként nem fordul le a program. Saját metódusainknál is megadhatjuk a kötelezően lekezelendő kivételeket. Nem kötelezően lekezelendő kivételek (unchecked exceptions):

– Futásidejű kivétel: olyan kivételek tartoznak ebbe a csoportba, amelyeket helyes programozással el lehet kerülni. Pl. a null referencia hiba, tömb index túllépés, stb. Az ilyen kivételek a RuntimeException leszármazottai, ami az Exception leszármazottja.

– Rendszerhiba: az alkalmazástól függetlenül (kívülről) felmerülő HW vagy rendszer hiba. Az ilyen kivételek az Error leszármazottai, ami a Throwable leszármazottja.

A Java kivételosztályok a Throwable osztály leszármazottai. A Java platform számtalan Exception-ből származtatott kivételosztályt definiál (pl. NegativeArraySizeException, IOException, FileNotFoundException, stb.)

A Java API hibás használatából eredő kivételek a RuntimeException osztály leszármazottai (pl. NullPointerException, IndexOutOfBoundsException, stb.). Akkor helyes a Java kód, ha kielégíti az „Elkapni vagy előírni” (Catch or Specify) követelményt. Ez azt jelenti, hogy azt a kódrészt, amelyik dobhat kivételt vagy

– try – catch blokkba kell zárni (azaz a kivételt el kell kapni és le kell kezelni), vagy

– olyan metódusban kell megadni, amelyik előirja, hogy milyen kivétel(eke)t dobhat (throws).

Az a kód, amelyik nem teljesíti ezt a követelményt, nem fordul le. A követelmény teljesítése figyelmen kívül hagyható, ha nem használunk kötelezően lekezelendő kivételeket. DE ez nem ajánlott!!!

Java-ban a try-catch-finally háromrészből álló vezérlő szerkezet szolgál a kivételkezelést is tartalmazó kódrészlet írására. Ez a szerkezet lehetővé teszi, hogy a normál működés kódját elválasszuk a hibakezelő kódtól (ezáltal olvashatóbb kódot kapjunk) és hiba esetén a megfelelő hibakezelő kód fusson le. A try részbe írjuk a normál működéshez tartozó olyan kódot, ami kivételt dobhat. Ebbe nem írunk hibakezelő kódot. Ha a programvégrehajtás során a try blokkban kivétel dobódik, a felmerült kivétel eseményt a hozzá társított kivételkezelő rész kezeli le. catch rész: Ebből több is lehet. Minden catch résznél meg kell adni, hogy milyen típusú kivételt tud elkapni, valamint a lekezelő kódot. Ha egy catch blokk egyszerre több kivételt is lekezel, a paramétere implicite final (azaz a blokkon belül nem változtatható meg az értéke).

Működése: A futtató rendszer az alapján dönti el, hogy egy catch blokk eltud-e kapni egy kivételt, hogy a catch fejében megadott kivétel referencia képes-e felvenni értékül a keletkezett kivételt. Az első olyan, amelyik képes, az elkapja a kivételt. Pl. mivel IOException az Exception osztályból származik, ezért csak a catch (IOException e) {...} catch (Exception e) {...} sorrend logikus, hiszen fordított esetben az Exception elkapná az IOException kivételeket is.

Működése: – Ha a try blokk utasításai hiba nélkül lefutnak, akkor a végrehajtás a finally blokkon folytatódik (ha van), majd a try szerkezet utáni utasításon. A catch blokkok nem hajtódnak végre. – Ha a try blokk végrehajtása során kivétel dobódik, akkor a try blokk további utasításai nem hajtódnak végre, hanem létrejön egy kivétel objektum és a futtató rendszer elkezdi megkeresni a megfelelő catch blokkot, amely képes lekezelni a kivételt. Ha talál ilyet, akkor lefut a kód blokkja, majd lefut a finally blokk (ha van) majd a try szerkezet utáni utasításon folytatódik a végrehajtás. Ha nem talál, akkor lefut a finally blokk, majd a try-t úgy tekinti, hogy megdobta a keletkezett kivételt. Ha egy catch vagy a finally blokk végrehajtása során kivétel keletkezik, akkor a futtató rendszer úgy tekinti a try-t, hogy megdobta az újonnan keletkezett kivételt. Ha a try kivételt dob és nincs megfelelő catch ág, vagy lekezelés közben (a catch blokkban) vagy finally közben kivétel keletkezik, akkor mivel a try szerkezetek egymásba ágyazhatók, a try-t tartalmazó try-catch-finally hibakezelő blokkjaiban kezdi keresni a megfelelő kivételkezelőt. Ha eljutva a legkülső try-catch-finally szerkezetig nem talál megfelelő catch ágat, akkor a program hibaüzenettel leáll. Egy kivétel lekezelése után mindig az után a try után folytatódik a végrehajtás, amelynek a catch ága lekezelte a kivételt. Normál működés során, mivel a finally blokk mindig lefut a try blokk után (akkor is, ha a try során kivétel keletkezett), ide tesszük azokat az utasításokat, amelyeknek feltétlenül le kell futniuk (cleanup code, erőforrások felszabadítása, pl. fájl lezárás). A try-with-resources utasítás (Java 7-től): A try paraméterében létrehozom az erőforrás(oka)t, amit használni akarok. A try blokk lefutása után ezek automatikusan megszűnnek.

Több erőforrás is megadható ;-el elválasztva. Erőforrás lehet bármilyen objektum, amelyik implementálja a java.lang.AutoCloseable interfészt vagy annak leszármazottját a java.io.Closable interfészt. A try-with-resources utasításhoz tartozó blokkban is dobódhat kivétel. A readFirstLineFromFile metódus try blokkjában a readLine metódushívás IOException kivételt dobhat, míg a try-with-resources utasítás akkor dobhat IOException kivételt, amikor lefut a BufferedReader objektum close metódusa. Ilyen esetekben a try blokk által dobott kivétel elnyomja a try-with-resources utasítás által dobott kivételt (suppressed exception). Az elnyomott kivétel előhívható a try blokk kivételkezelőjében a Throwable.getSuppressed metódushívással. Metódus határokon nem tud automatikusan tovább adódni a kivétel. Tehát ha egy kivétel egy metóduson belül keletkezett, akkor azt a metóduson belül kell lekezelni vagy jelezni kell, hogy a metódus megdobhatja a kivételt és akkor a hívónak kell lekezelnie. A metódus a fejlécében tudja jelezni a megdobódható kivételeket: modifiers type methodname(paramlist) throws exceptiontypelist { … } exceptiontypelist: kivételosztályok vesszővel elválasztva (ebben a listában a nem kötelezően lekezelendő kivételeket nem kötelező szerepeltetni) Kivételt bármilyen kód blokk dobhat: throw someThrowableObject;

Példa: verem objektum pop() metódusa. Ha a verem üres, nem tudjuk kivenni belőle a következő elemet. public Object pop() { Object obj; if (size == 0) { throw new EmptyStackException(); } … } Az EmptyStackException a java.util csomagban definiált kivételosztály (nem kötelezően lekezelendő kivétel). A dobott kivételt vagy a hívó metóduson belül kell lekezelni (catch blokkban), vagy tovább dobhatjuk a catch blokkban, vagy jelezzük h. a hívó metódus kezelje.

A saját kivételosztályokat vagy az Exception osztályból származtatjuk (közvetve, közvetlenül), ekkor kötelezően lekezelendő lesz vagy a RuntimeException-ból, ekkor nem lesz kötelezően lekezelendő. Ügyeljünk, hogy a nem kötelezően lekezelendő kivételekkel figyelmen kívül hagyjuk az „Elkapni vagy előírni” (Catch or Specify) követelményt. A saját kivételosztály neve utaljon a kivétel jellegére. Szokásos megadása: xxxException. Ne definiáljunk már létező kivételosztályt! Az Exception osztály fontosabb tagjai:

– Adattag: String message

– Konstruktorok: Exception(), Exception(String message)

– Metódusok: getMessage(), printStackTrace()