KeyTraining

Curs java pentru testare automata

JAVA

Cuprins curs:

- Exception handling
- Try/catch/finally
- Abstract classes

Exception handling

- Tratarea excepțiilor în java este un mecanism sau o tehnică care ne permite să gestionăm erorile de rulare într-un program, astfel încât fluxul normal al programului să poată fi menținut.
- O excepție în java este un obiect care reprezintă o eroare sau o condiție anormală care apare la runtime și întrerupe fluxul normal de execuție al programului.
- Cu alte cuvinte, comportamentul / evenimentul nedorit și neașteptat care întrerupe fluxul normal de execuție al programului se numește excepție în java.
- Este declansat dintr-o metodă.
- La apelul metodei se poate prinde și gestiona excepția.

Exception handling

- De exemplu, dacă accesăm un array utilizând un index care este în afara limitelor, vom primi o eroare de rulare numită ArrayIndexOutOfBoundsException.

- Ierarhia ArrayIndexOutOfBoundsException din care face parte :
- https://docs.oracle.com/javase/7/docs/api/java/lang/ArrayIndexOutOfBoundsException.html

Exception handling

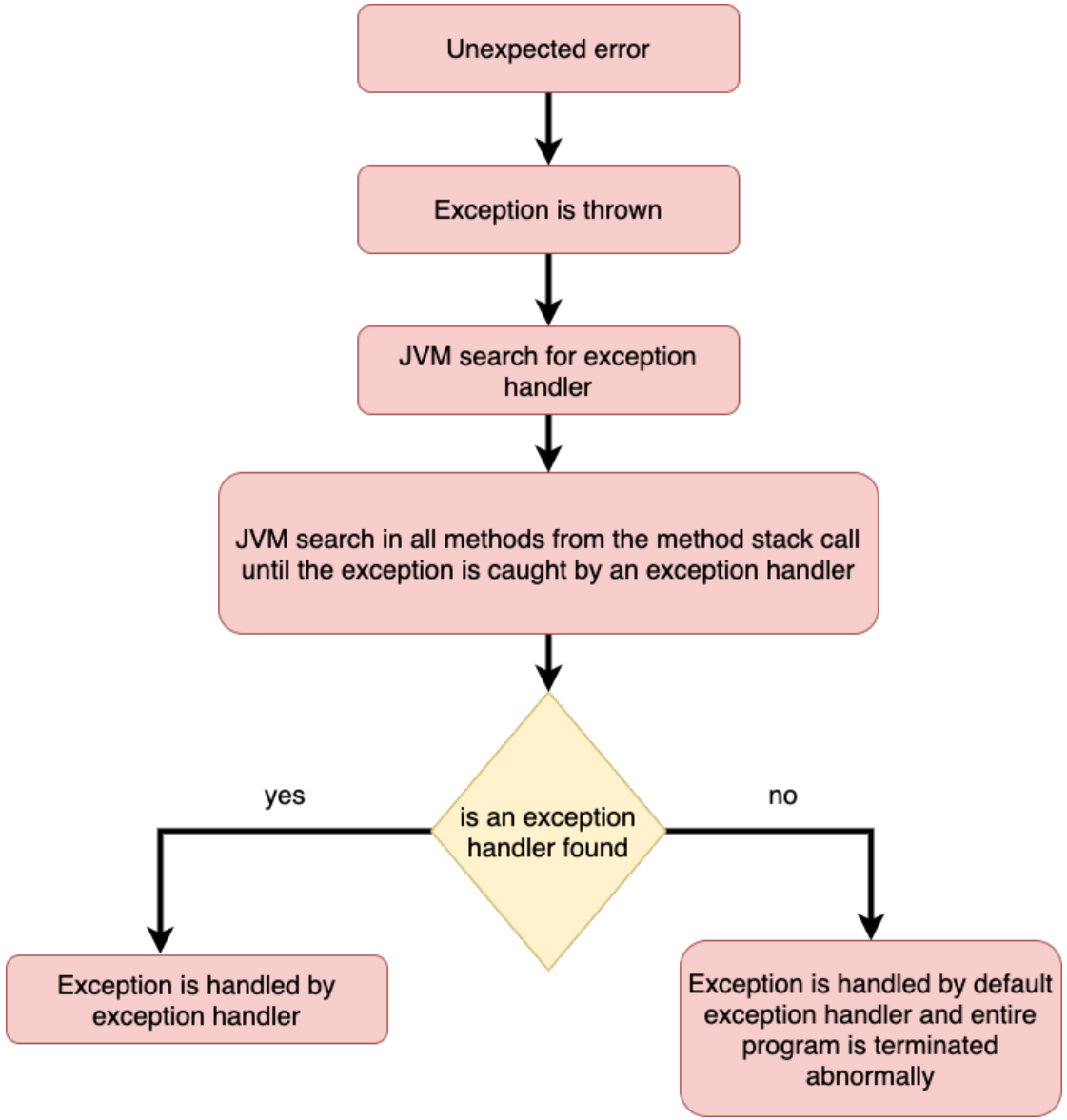
- Codul care prinde excepția aruncată de JVM se numește handler de excepție în Java. Este responsabil pentru primirea informațiilor despre excepție / eroare.
- Când apare o excepție, gestionarea excepțiilor transferă controlul execuției programului către un handler de excepție adecvat.
- De exemplu, să presupunem că apelăm la o metodă care deschide un fișier, dar fișierul nu se deschide. În acest caz, executarea acelei metode se va opri și codul pe care l-am scris pentru a face față acestei situații, va fi rulat.

Exception handling

- Prin urmare, avem nevoie de o modalitate alternativă de a spune JVM-ului ce cod trebuie executat pentru a menține fluxul normal al programului atunci când apare o anumită excepție.
- Astfel, gestionarea excepțiilor (handling exceptions) este o modalitate alternativă de a continua execuția restului programului în mod normal.

Exception handling

- Get the exception (detectează problema)
- Throw exception (informați că a apărut o eroare)
- Catch the exception (Primiți informații despre eroare)
- Handle exception (Luați măsuri adecvate și corective)


Unexpected error
Exception is thrown
JVM search for exception handler
JVM search in all methods from the method stack call until the exception is caught by an exception handler
is an exception handler found
yes
no
Exception is handled by exception handler
Exception is handled by default exception handler and entire program is terminated abnormally

De ce apar exceptiile ?

- Pot exista multe motive care ar putea genera o excepție într-un program Java.
- 1. Deschiderea unui fișier inexistent în programul dvs.
- 2. Citirea unui fișier de pe un disc, dar fișierul nu există acolo.
- 3. Scrierea datelor pe un disc, dar discul este plin sau neformatat.
- 4. Când programul solicită introducerea utilizatorului și acesta introduce date nevalide.
- 5. Când un utilizator încearcă să împartă o valoare întreagă la zero, apare o excepție.
- 6. Când un flux de date este într-un format nevalid etc.

Types of exceptions

- Practic, există două tipuri de excepții în java API.
- 1. Excepții predefinite (Excepții încorporate)
- 2. Excepții custom

Excepții predefinite: Excepțiile predefinite sunt acele excepții care sunt deja definite de sistemul Java.

Aceste excepții sunt numite și excepții încorporate. Java API acceptă gestionarea excepțiilor oferind numărul de excepții predefinite.

Aceste excepții predefinite sunt reprezentate de clase în java.

Când apare o excepție predefinită, JVM (Java runtime system) creează un obiect din clasa de excepție predefinită.

Types of exceptions

- Toate excepțiile sunt derivate din clasa java.lang.Throwable, dar nu toate clasele de excepție sunt definite în același pachet.

Toate excepțiile predefinite acceptate de java sunt organizate ca subclase într-o ierarhie din clasa Throwable.

Clasa Throwable este rădăcina ierarhiei excepțiilor și este o subclasă imediată a clasei Object.

Toate excepțiile predefinite sunt împărțite în două grupuri:

1. Checked Exceptions
2. Unchecked Exceptions

Checked Exceptions

- Checked Exceptions : sunt acele excepții care sunt verificate de compilatorul Java însuși la momentul compilării și care nu se află în ierarhia claselor de excepții de execuție.
- Dacă o metodă aruncă o excepție checked într-un program, metoda trebuie fie să gestioneze excepția, fie să o transmită unei metode de apelare.
- Excepțiile checked trebuie tratate fie folosind blocul try and catch, fie folosind clauza throws în declarația metodei.
- Dacă nu se manipulează corect, va da o eroare în timpul compilării.
- Toate excepțiile apar întotdeauna numai la runtime, dar unele excepții sunt detectate în timpul compilării și altele la runtime. Excepțiile care sunt verificate de compilatorul Java în momentul compilării se numește excepție verificată (checked exceptions) în Java. Toate excepțiile, cu excepția RuntimeException, Error și subclasele lor, sunt excepții verificate.

Checked Exceptions

- O listă cu câteva excepții importante din categoria checked este prezentată mai jos:
- ClassNotFoundException
- InterruptedException
- InstantiationException
- IOException
- SQLException
- IllegalAccessException
- FileNotFoundException, etc

Unchecked Exceptions

- Excepțiile unchecked în Java sunt acele excepții care sunt verificate de JVM, nu de compilatorul Java.
- Acestea apar în timpul rulării unui program. Toate excepțiile din clasa de excepții runtime sunt numite excepții unchecked sau excepții runtime în Java.
- Putem scrie un program Java și îl putem compila. Dar nu putem vedea efectul excepțiilor și erorilor nebifate până nu executăm programul. Acest lucru se datorează faptului că compilatorul Java ne permite să scriem un program Java fără a gestiona excepțiile și erorile unchecked.
- Compilatorul Java nu verifică excepția runtime la compilare, indiferent dacă programatorul le gestionează sau nu. Dacă apare o excepție de runtime într-o metodă și programatorul nu o gestionează, JVM încetează programul fără executarea restului codului.

Unchecked Exceptions

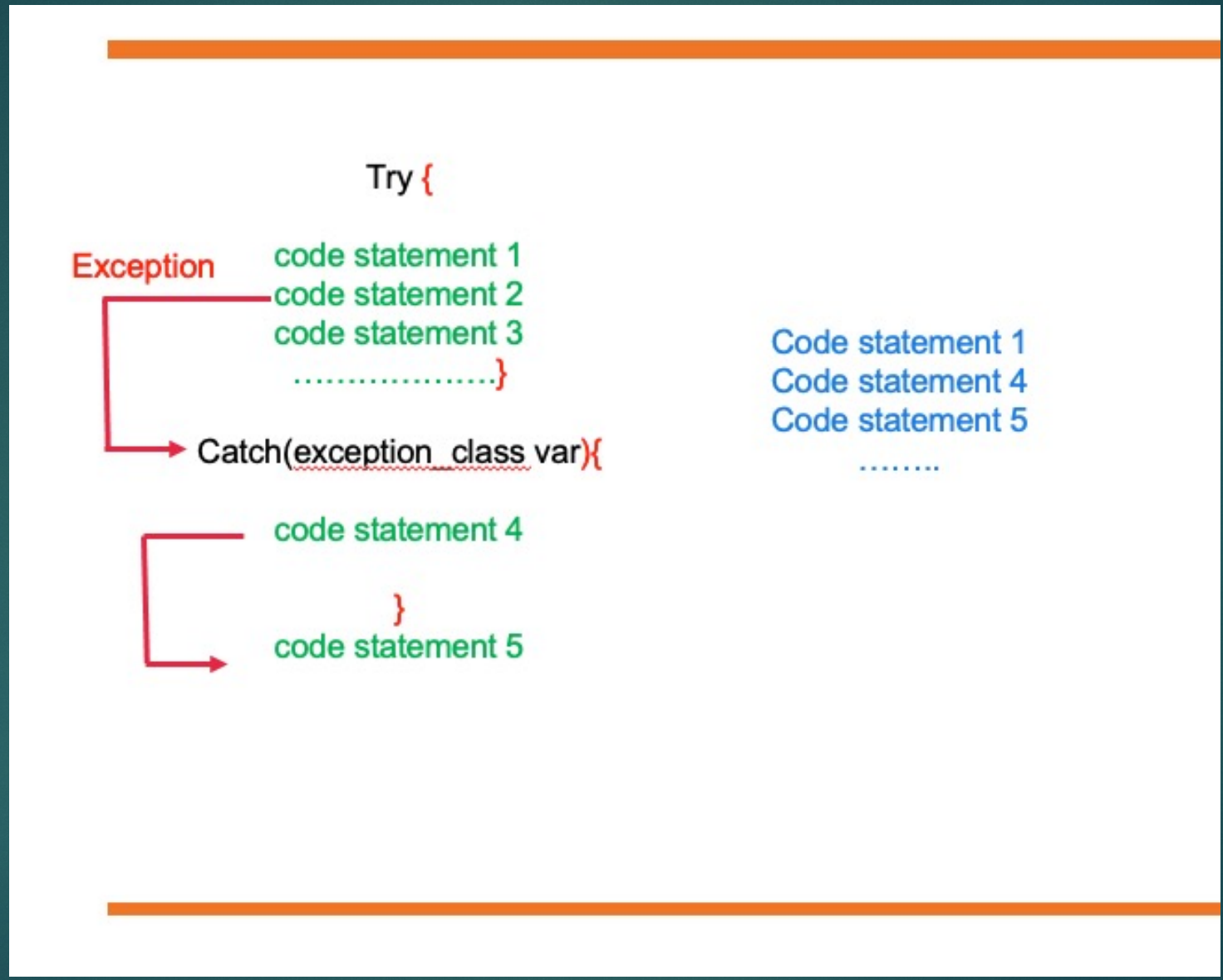
- Câteva exemple importante de excepții în timpul rulării sunt prezentate mai jos:
- ArithmeticException
- ClassCastException
- NullPointerException
- ArrayIndexOutOfBoundsException
- NegativeArraySizeException
- ArrayStoreException
- IllegalThreadStateException
- SecurityException, etc.

Checked vs Unchecked Exceptions

- 1. Clasele care moștenesc direct clasa Throwable, cu excepția RuntimeException și Error, se numesc excepții checked, în timp ce clasele care moștenesc direct RuntimeException sunt numite excepții unchecked.
- 2. Excepțiile checked sunt verificate și tratate în timpul compilării, în timp ce excepțiile necontrolate nu sunt verificate și tratate la compilare. Sunt verificate în timpul rulării.
- 3. Când apare o excepție checked într-o metodă, metoda trebuie fie să prindă excepția, fie să transmită excepția la metoda apelantului său. Dar, în cazul excepției unchecked, compilatorul Java nu forțează să prindă excepția sau să o declare într-o clauză throws.
- 4. Excepțiile checked în java extind clasa Exception, în timp ce excepțiile unchecked extind clasa RuntimeException.

Try -Catch

- Blocul try-catch este o tehnică utilizată pentru a prinde și gestiona excepțiile in Java.
- Dacă apare o excepție în blocul try, restul de cod din blocul try nu este executat și controlul executării este transferat din blocul try in blocul catch care gestionează excepția aruncată de blocul try.
- Catch acționează ca un handler de excepție care ia un singur argument. Acest argument este o referință a unui obiect de excepție care poate sa fie excepție de clasă, fie excepție de superclasă.


Try {
Exception
code statement 1
code statement 2
code statement 3
.....}
Catch(exception_class var){
code statement 4
}
code statement 5
Code statement 1
Code statement 4
Code statement 5
.....

Try –Catch

- O excepție poate fi gestionată folosind try, catch și finally
- Putem avea nested try catch
- Un try poate avea mai multe statementuri catch

Try –Catch-Finally

- „finally” este un cuvânt cheie folosit pentru a crea un bloc de cod care urmează unui bloc try sau catch.
- Un bloc finally conține toate codurile cruciale, cum ar fi închiderea conexiunilor, fluxului etc., care se execută întotdeauna indiferent dacă apare sau nu o excepție în cadrul unui try-catch.
- Când finally este atașat cu un bloc try-catch, se execută întotdeauna dacă blocul catch a tratat excepția aruncată de blocul try sau nu.

Try –Catch-Finally

- Un bloc finally este opțional, dar cel puțin unul dintre catch sau finally trebuie să existe cu un try.
- Trebuie definit la sfârșitul ultimului bloc catch. Dacă finally este definit înainte de catch, programul nu se va compila cu succes.
- Spre deosebire de catch, mai multe finally nu pot fi declarate cu un singur bloc de try. Adică poate exista o singură clauză finală cu un singur bloc de try.

Throw keyword

- Throw in Java este un cuvânt cheie care este utilizat pentru a arunca în mod explicit sau manual o excepție.
- Folosind cuvântul cheie throw, putem arunca excepții checked sau unchecked în programarea java. Când apare o excepție în blocul try, cuvântul cheie throw transferă controlul execuției către apelant aruncând un obiect de excepție.
- Numai un obiect de tip excepție poate fi aruncat folosind cuvântul cheie throw la un moment dat.
- Cuvântul cheie Throw poate fi utilizat în interiorul unei metode sau a unui bloc static, cu condiția să fie prezentă gestionarea excepțiilor.

Throws keyword

- Throws este utilizat în declarația metodei. Oferă informații metodei apelantului despre excepțiile care sunt aruncate, iar metoda apelantului trebuie să își asume responsabilitatea de a gestiona excepția.
- Cuvântul cheie Throws este utilizat numai în caz de excepție verificată pentru că, dacă nu gestionăm excepții de execuție (excepții unchecked), compilatorul Java nu dă nicio eroare legată de excepțiile de execuție.
- Dacă apare o eroare, nu putem face nimic. Când codul generează o excepție checked în interiorul unei metode, dar metoda nu o gestionează, compilatorul Java o detectează și ne informează despre aceasta pentru a gestiona această excepție.
- În acest caz, în mod obligatoriu, trebuie să gestionăm excepția checked, altfel vom primi o eroare marcată de compilatorul Java.

OOP : Abstraction

- Abstracția este o tehnică prin care putem ascunde datele inutile care nu sunt necesare unui utilizator și expunem doar că datele sunt de interes pentru utilizator.
- Ascundem toate datele nedorite, astfel încât utilizatorii să poată lucra numai cu datele necesare.
- Abstracția este unul dintre principiile fundamentale ale programării orientate pe obiecte, care permite utilizatorului să utilizeze un obiect fără să cunoască detaliile sale interne.
- Ajută la reducerea complexității prin faptul că nu include detalii de fundal.

OOP : Abstraction

- Un exemplu în timp real de abstractizare este „trimiterea de Email-uri”.
- Când trebuie să trimiteți un email, tastați doar textul și trimiteți mesajul. Dar nu știți procesarea internă despre livrarea mesajului.
- Există două moduri de a realiza sau implementa abstractizarea în programul Java.
- Acestea sunt după cum urmează:
  - Clasa abstractă
  - Interfață

Clasa abstracta

- O clasă abstractă în Java este o clasă, care este declarată cu un cuvânt cheie „abstract”.
- Diferențele fata de o clasa normala sunt :
  - Nu putem crea un obiect al acestei clase. Se pot crea numai obiecte din subclasele sale non-abstracte (sau concrete).
  - Poate avea zero sau mai multe metode abstracte care nu sunt permise într-o clasă non-abstractă (clasă concretă)

Clasa abstracta

- „Abstract” este un modificator fără acces(non –acces modifier) în java care se aplică pentru clase, interfețe, metode și clase interioare.
- Reprezintă o clasă incompletă care depinde de subclase pentru implementarea sa.
- Crearea subclasei este obligatorie pentru clasa abstractă.
- O clasă non-abstractă este uneori numită clasă concretă.
- Un concept abstract nu se aplică variabilelor.