EXCEPTIONS

1. Error

An error is an irrecoverable condition occurring at runtime. Such as OutOfMemory error. These JVM errors you cannot repair at runtime. Though error can be caught in the catch block the execution of the application will come to a halt and is not recoverable.

1. Exception

Exceptions are conditions that occur because of bad input or human error etc. e.g. FileNotFoundException will be thrown if the specified file does not exist. Or a NullPointerException will take place if you try using a null reference. In most cases, it is possible to recover from an exception (probably by giving the user feedback for entering proper values, etc.

1. How can you handle Java exceptions?

There are five keywords used to handle exceptions in Java:

* + - try
    - catch
    - finally
    - throw
    - throws

1. Checked Exception

* The classes that extend the Throwable class except RuntimeException and Error are known as checked exceptions.
* Checked exceptions are checked at compile time.
* Example: IOException, SQLException, etc.

1. Unchecked Exception

* The classes that extend RuntimeException are known as unchecked exceptions.
* Unchecked exceptions are not checked at compile time.
* Example: ArithmeticException, NullPointerException etc.

1. What is a finally block? Is there a case when finally will not execute?

Finally block is a block that always executes a set of statements. It is always associated with a try block regardless of any exception that occurs or not.

Finally will not be executed if the program exits either by calling System. exit() or by causing a fatal error that causes the process to abort.

1. Will the finally block get executed when the return statement is written at the end of the try block and catch?

The finally block always gets executed even when the return statement is written at the end of the try block and the catch block. It always executes, whether there is an exception or not. There are only a few situations in which the finally block does not execute, such as VM crash, power failure, software crash, etc. If you don’t want to execute the finally block, you need to call the System. exit() method explicitly in the finally block.

1. How does an exception propagate in the code?

If an exception is not caught, it is thrown from the top of the stack and falls down the call stack to the previous procedure. If the exception isn’t caught there, it falls back to the previous function, and so on, until it’s caught or the call stack reaches the bottom. The term for this is Exception propagation.

1. Throw

* Throw is used to explicitly throw an exception.
* Checked exceptions cannot be propagated with throw only.
* Throw is followed by an instance.
* Throw is used within the method.
* You cannot throw multiple exceptions

1. Throws

* Throws is used to declare an exception.
* Checked exception can be propagated with throws.
* Throws are followed by class.
* Throws are used with the method signature.
* You can declare multiple exceptions e.g. public void method() throws IOException, SQLException.

1. . What is the exception hierarchy in java?

Throwable is a parent class of all Exception classes. There are two types of Exceptions: Checked exceptions and UncheckedExceptions or RunTimeExceptions. Both types of exceptions extend the Exception class whereas errors are further classified into Virtual Machine error and Assertion error.

1. How to create a custom Exception?

To create your exception extend the Exception class or any of its subclasses.

* class New1Exception extends Exception { } // this will create Checked Exception
* class NewException extends IOException { } // this will create Checked exception
* class NewException extends NullPonterExcpetion { } // this will create UnChecked exception

1. What are the important methods of Java Exception Class?

Exception and all of its subclasses don’t provide any specific methods and all of the methods are defined in the base class Throwable.

* String getMessage() – This method returns the message String of Throwable and the message can be provided while creating the exception through its constructor.
* String getLocalizedMessage() – This method is provided so that subclasses can override it to provide locale-specific messages to the calling program. Throwable class implementation of this method simply uses the getMessage() method to return the exception message.
* Synchronized Throwable getCause() – This method returns the cause of the exception or null if the cause is unknown.
* String toString() – This method returns the information about Throwable in String format, the returned String contains the name of the Throwable class and localized message.
* void printStackTrace() – This method prints the stack trace information to the standard error stream, this method is overloaded and we can pass PrintStream or PrintWriter as an argument to write the stack trace information to the file or stream.

1. Can we write multiple catch blocks under a single try block?

Yes, we can have multiple catch blocks under a single try block but the approach should be from specific to general.

1. What is OutOfMemoryError in Java?

OutOfMemoryError is the subclass of java.lang.The rich generally occurs when our JVM runs out of memory.

ИЗКЛЮЧЕНИЯ

1. Грешка

Грешката е невъзстановено условие, възникващо по време на изпълнение. Например OutOfMemory грешка. Тези JVM грешки, които не можете да поправите по време на изпълнение. Макар че грешката може да бъде уловена в блока за улов изпълнението на приложението ще спре и не подлежи на възстановяване.

2. Изключение

Изключенията са условия, които възникват поради лош вход или човешка грешка и т.н. например FileNotFoundException ще бъдат хвърлени, ако указаният файл не съществува. Или NullPointerException ще се проведе, ако опитате да използвате null препратка. В повечето случаи е възможно да се възстанови от изключение (вероятно като дадете на потребителя обратна връзка за въвеждане на правилното стойности и т.н.

3. Как можете да се справите с изключенията на Java?

Има пет ключови думи, използвани за обработка на изключения в Java:

  опитвам

  улов

  накрая

  хвърлям

  Хвърля

4. Проверено изключение

        Класовете, които разширяват класа Throwable с изключение на RuntimeException и Error, са известни като проверени изключения.

        Проверените изключения се проверяват по време на компилиране.

        Пример: IOException, SQLException и др.

5. Непроверено изключение

        Класовете, които разширяват RuntimeException, са известни като непроверени изключения.

        Непроверените изключения не се проверяват по време на компилиране.

        Пример: АритметичноИзцепване, NullPointerИзцепция и т.н.

6. Какво е най-накрая блок? Има ли случай, когато накрая няма да изпълни?

Накрая блок е блок, който винаги изпълнява набор от изявления. Тя винаги е свързана с опитайте блок независимо от всяко изключение, което се случва или не.

Накрая няма да бъде изпълнена, ако програмата излезе нито чрез извикване на System. изход() или чрез причиняване на фатална грешка, която причинява спиране на процеса.

7. Ще се изпълни ли най-накрая блокът, когато декларацията за връщане е написана в края на блока за опит и улов?

Най-накрая блок винаги се изпълнява, дори когато декларацията за връщане е написана в края на блока за опит и блока за улов. Тя винаги изпълнява, независимо дали има изключение или не. Има само няколко ситуации, в които най-накрая блок не изпълнява, като VM катастрофа, прекъсване на захранването, софтуерна катастрофа и т.н. Ако не искате да изпълните най-накрая блока, трябва да се обадите на Системата. метода изход() изрично в окончателното блокиране.

8. Как се размножава изключение в кода?

Ако не се хване изключение, то се хвърля от горната част на стека и пада надолу по стека за обаждания до предишната процедура. Ако изключението не е уловено там, то пада обратно към предишната функция и т.н., докато не бъде уловена или стека за обаждания достигне дъното. Терминът за това е размножаване на Изключение.

9. Хвърляне

        Хвърляне се използва за изрично хвърляне на изключение.

        Проверените изключения не могат да се размножават само с хвърляне.

        Хвърляне е последвано от инстанция.

        Хвърляне се използва в рамките на метода.

        Не можете да хвърляте няколко изключения

10. Хвърля

        Хвърляния се използва за деклариране на изключение.

        Проверено изключение може да се размножва с хвърляния.

        Хвърляния са последвани от час.

        Хвърлянията се използват с подписа на метода.

        Можете да декларирате множество изключения например метод публична празнота() хвърля IOException, SQLException.

11. . Каква е йерархията на изключенията в java?

Хвърленото е родителски клас от всички класове Изключение. Има два типа изключения: Проверени изключения и НепроверениВъздействия или RunTimeExceptions. И двата вида изключения разширяват класа изключение като има предвид, че грешките допълнително се класифицират в грешка на Виртуалната машина и грешка в Assertion.

12. Как да създадете персонализирано Изключение?

За да създадете изключението си, удължите класа Изключение или някоя от неговите подкласове.

        клас New1Exception разширява Изключение { } // това ще създаде Проверено изключение

        клас NewException разширява IOException { } // това ще създаде Проверено изключение

        клас NewException разширява NullPonterExcpetion { } // това ще създаде UnChecked изключение

13. Кои са важните методи на Java Изключение Клас?

Изключение и всичките му подкласове не предоставят никакви конкретни методи и всички методи са определени в базовия клас Throwable.

        Низ getMessage() – Този метод връща съобщението Низ от Throwable и съобщението може да бъде предоставено, докато създавате изключението чрез неговия конструктор.

        String getLocalizedMessage() – Този метод е предоставен, така че подкласовете да могат да го заместят, за да предоставят специфични за езиковата променлива съобщения на повикващата програма. Хвърляне клас изпълнение на този метод просто използва getMessage() метод за връщане на съобщението за изключение.

        Синхронизирани Throwable getCause() – Този метод връща причината за изключението или null, ако причината е неизвестна.

        String toString() – Този метод връща информацията за Throwable in String формат, върнатия Низ съдържа името на класа Throwable и локализираното съобщение.

        void printStackTrace() – Този метод отпечатва информацията за проследяване на стека към стандартния поток грешки, този метод е претоварен и можем да преминем PrintStream или PrintWriter като аргумент за запис на информацията за проследяване на стека към файла или потока.

14. Можем ли да напишем няколко блока за улов под единичен трик блок?

Да, можем да имаме няколко блока за улов под един-единствен опит блок но подходът трябва да бъде от специфични към общи.

15. Какво е OutOfMemory Грешка в Java?

OutOfMemoryError е подкласа на java.lang.Богатите по принцип се случва, когато на нашия JVM свърши паметта.