Build COMPETENCY
across your TEAM

SYNERGETICS
GET IT RIGHT

Microsoft Partner
Gold Cloud Platform
Silver Learning

Exception Handling

Google Cloud Platform

Windows Azure

amazon web services

APACHE kafka
A distributed streaming platform

Apache Software Foundation

APACHE Spark

Java

hadoop

Context And Dependency Injection

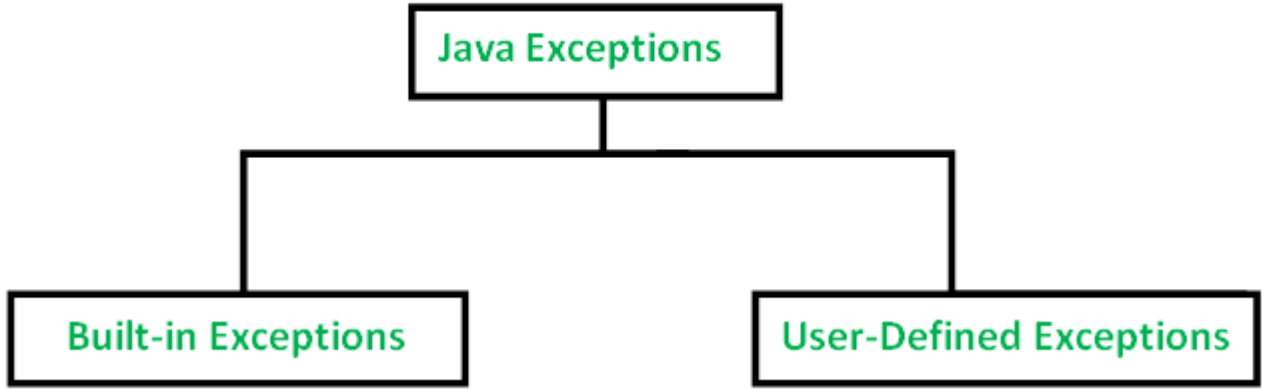
SYNERGETICS GET IT RIGHT | Microsoft Partner Gold Cloud Platform Silver Learning

- Exception Types
- Exception Hierarchy
- Try-catch-finally
- Try-with-resources
- Multi catch blocks
- Throwing exceptions using throw
- Declaring exceptions using throws
- User defined Exceptions

Exception Types

Types of Exception in Java

- Java defines several types of exceptions that relate to its various class libraries. Java also allows users to define their own exceptions.



Built-in Exceptions

- **Arithmetic Exception**
  It is thrown when an exceptional condition has occurred in an arithmetic operation.
- **ArrayIndexOutOfBoundException**
  It is thrown to indicate that an array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of the array.
- **ClassNotFoundException**
  This Exception is raised when we try to access a class whose definition is not found
- **FileNotFoundException**
  This Exception is raised when a file is not accessible or does not open.
- **IOException**
  It is thrown when an input-output operation failed or interrupted
- **InterruptedException**
  It is thrown when a thread is waiting , sleeping , or doing some processing , and it is interrupted.

User-Defined Exceptions

- Sometimes, the built-in exceptions in Java are not able to describe a certain situation. In such cases, user can also create exceptions which are called ‘user-defined Exceptions’.

- Following steps are followed for the creation of user-defined Exception.

- The user should create an exception class as a subclass of Exception class. Since all the exceptions are subclasses of Exception class, the user should also make his class a subclass of it.

class MyException extends Exception

User-Defined Exceptions

- We can write a default constructor in his own exception class.

```
MyException(){}
```

- We can also create a parameterized constructor with a string as a parameter. We can use this to store exception details.

```
MyException(String str)
{
   super(str);
}
```

User-Defined Exceptions

- To raise exception of user-defined type, we need to create an object to his exception class and throw it using throw clause, as:

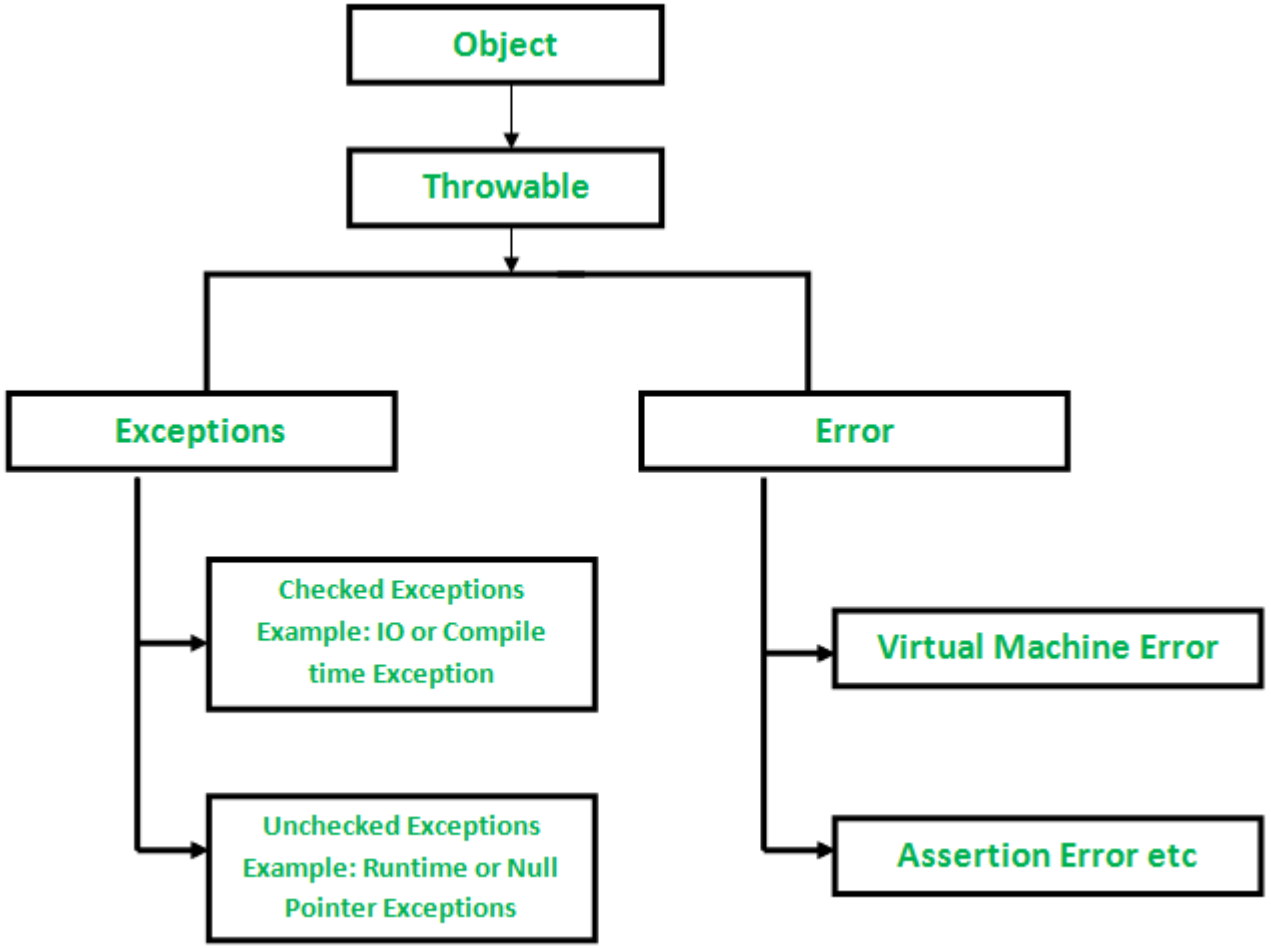
MyException me = new MyException("Exception details");

throw me;

- The following program illustrates how to create own exception class MyException.
- Details of account numbers, customer names, and balance amounts are taken in the form of three arrays.
- In main() method, the details are displayed using a for-loop. At this time, check is done if in any account the balance amount is less than the minimum balance amount to be ept in the account.
- If it is so, then MyException is raised and a message is displayed "Balance amount is less".

Exception Hierarchy

The exception hierarchy



Types of Java Exceptions

There are mainly two types of exceptions: checked and unchecked. Here, an error is considered as the unchecked exception. According to Oracle, there are three types of exceptions:

- Checked Exception
- Unchecked Exception
- Error

Difference between Checked and Unchecked Exceptions

- Checked Exception

The classes which directly inherit Throwable class except RuntimeException and Error are known as checked exceptions e.g. IOException, SQLException etc. Checked exceptions are checked at compile-time.

- Unchecked Exception

The classes which inherit RuntimeException are known as unchecked exceptions e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

- Error

Error is irrecoverable e.g. OutOfMemoryError, VirtualMachineError, AssertionError etc.

SYNERGETICS GET IT RIGHT | Microsoft Partner Gold Cloud Platform Silver Learning

Try-catch-finally

Try-catch-finally

```
public class JavaExceptionExample{
  public static void main(String args[]){
   try{
      //code that may raise exception
      int data=100/0;
   }catch(ArithmeticException e){System.out.println(e);}
   //rest code of the program
   System.out.println("rest of the code...");
  }
}
```

Try-catch-finally

Keyword	Description
try	The "try" keyword is used to specify a block where we should place exception code. The try block must be followed by either catch or finally. It means, we can't use try block alone.
catch	The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.
finally	The "finally" block is used to execute the important code of the program. It is executed whether an exception is handled or not.
throw	The "throw" keyword is used to throw an exception.
throws	The "throws" keyword is used to declare exceptions. It doesn't throw an exception. It specifies that there may occur an exception in the method. It is always used with method signature.

Multi catch blocks

- In Java 7, catch block has been improved to handle multiple exceptions in a single catch block.
- If you are catching multiple exceptions and they have similar code, then using this feature will reduce code duplication.

```
catch (IOException ex) {
    logger.error(ex);
    throw new MyException(ex.getMessage());
catch (SQLException ex) {
    logger.error(ex);
    throw new MyException(ex.getMessage());
}
```

```
catch(IOException | SQLException ex){
    logger.error(ex);
    throw new MyException(ex.getMessage());
}
```

Throwing exceptions using throw

Throwing exceptions using throw

- The throw keyword in Java is used to explicitly throw an exception from a method or any block of code.

- We can throw either checked or unchecked exception. The throw keyword is mainly used to throw custom exceptions.

throw Instance

Example:

```
throw new ArithmeticException("/ by zero");
```

Note: This exception i.e, Instance must be of type Throwable or a subclass of Throwable

Throwing exceptions using throw

- The flow of execution of the program stops immediately after the throw statement is executed and the nearest enclosing try block is checked to see if it has a catch statement that matches the type of exception.
- If it finds a match, controlled is transferred to that statement otherwise next enclosing try block is checked and so on.
- If no matching catch is found then the default exception handler will halt the program.

Throwing exceptions using throw

```
class ThrowExcep
{
    static void fun()
    {
        try
        {
            throw new NullPointerException("demo");
        }
        catch(NullPointerException e)
        {
            System.out.println("Caught inside fun().");
            throw e; // rethrowing the exception
        }
    }
}
```

```
public static void main(String args[])
{
    try
    {
        fun();
    }
    catch(NullPointerException e)
    {
        System.out.println("Caught in main.");
    }
}
```

Declaring exceptions using throws

Declaring exceptions using throws

- throws is a keyword in Java which is used in the signature of method to indicate that this method might throw one of the listed type exceptions.

- The caller to these methods has to handle the exception using a try-catch block.

Declaring exceptions using throws

- In a program, if there is a chance of rising an exception then compile always warn us about it and compulsorily we should handle that checked exception, Otherwise we will get compile time error saying **unreported exception XXX must be caught or declared to be thrown**.

- To prevent this compile time error we can handle the exception in two ways:
  - By using try catch
  - By using throws keyword

- We can use throws keyword to delegate the responsibility of exception handling to the caller (It may be a method or JVM) then caller method is responsible to handle that exception.

Declaring exceptions using throws

```
class tst
{
    public static void main(String[] args)
    {
        Thread.sleep(10000);
        System.out.println("Hello World");
    }
}
```

error: unreported exception InterruptedException; must be caught or declared to be thrown

- In the above program, we are getting compile time error because there is a chance of exception if the main thread is going to sleep, other threads get the chance to execute main() method which will cause InterruptedException.

Declaring exceptions using throws

```
class tst
{
    public static void main(String[] args)throws InterruptedException
    {
        Thread.sleep(10000);
        System.out.println("Hello World");
    }
}
```

Output: Hello World

Important points about throws keyword

- throws keyword is required only for checked exception and usage of throws keyword for unchecked exception is meaningless.
- throws keyword is required only to convince compiler and usage of throws keyword does not prevent abnormal termination of program.
- By the help of throws keyword we can provide information to the caller of the method about the exception.

SYNERGETICS GET IT RIGHT | Microsoft Partner Gold Cloud Platform Silver Learning

User defined Exceptions

User Defined Exception

- User Defined Exception or custom exception is creating your own exception class and throws that exception using ‘throw’ keyword.
- This can be done by extending the class Exception.
- We require to create a custom exception to represent the exception in a proper way and there is no inbuilt exception that could be used.
  - Eg, NegativeBalanceException, UnEligableToVoteException…
- To Create a Checked UserDefinedException extend from RuntimeException or its subtype and to create a UnCheckedUserDefinedException extend from Exception class or its subtype but not from the RuntimeException.

Example

```
class ZeroSalaryException extends Exception
{
        public ZeroSalaryException(){
        }

        public ZeroSaaryException(String msg){
                super(msg);
        }
}
```

- We can throw this exception by saying throw new ZeroSalaryException

SYNERGETICS GET IT RIGHT | Microsoft Partner
Gold Cloud Platform
Silver Learning

Q & A

Contact: amitmahadik@synergetics-india.com

SYNERGETICS GET IT RIGHT | Microsoft Partner
Gold Cloud Platform
Silver Learning

Thank You