Android Development with Kotlin

Exceptions in Kotlin

What is an exception?

Exception is a runtime problem which occurs in the program and leads to program termination. This may occur due to running out of memory space, array out of bond, condition like divided by zero. To handle this type of problem during program execution the technique of exception handling is used.

What is exception handling?

Exception handling is a technique which handles the runtime problems and maintains the flow of program execution. In Kotlin, all exception classes are descendants of class Throwable. To throw an exception object, Kotlin uses the throw expression.

```
throw MyException("this throws an exception")
```

There are four different keywords used in exception handling. These are:

- try
- catch
- finally
- throw

try: try block contains set of statements which might generate an exception. It must be followed by either catch or finally or both.

catch: catch block is used to catch the exception thrown from try block.

finally: finally block always execute whether exception is handled or not. So it is used to execute important code statement.

throw: throw keyword is used to throw an exception explicitly.

Kotlin Unchecked exceptions:

Unchecked exception is that exception which is thrown due to mistakes in our code. This exception type extends RuntimeException class. The Unchecked exception is checked at run time. Following are some example of unchecked exceptions:

- **ArithmeticException:** thrown when we divide a number by zero.
- **ArrayIndexOutOfBoundExceptions:** thrown when an array has been tried to access with incorrect index value.
- **SecurityException:** thrown by the security manager to indicate a security violation.
- **NullPointerException:** thrown when invoking a method or property on a null object.

Kotlin try catch:

Kotlin try-catch block is used for exception handling in the code. The try block encloses the code which may throw an exception and the catch block is used to handle the exception. This block must be written within the method. Kotlin try block must be followed by either catch block or finally block or both.

```
try{
//code that may throw exception
}catch(e: SomeException){
//code that handles exception
}


try { // try catch with finally
    // some code
}
catch (e: SomeException) {
    // handler
}
finally {
    // optional finally block
}
```

Problems without exception handling:

```
fun main(args: Array<String>){

val data = 20 / 0   //may throw exception

println("code below exception ...")

}
```

Output:

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
at ExceptionHandlingKt.main(ExceptionHandling.kt:2)
```

This above program generates an exception, which causes rest of code below the exception not executable.

Solution by Exception Handling:

```
fun main(args: Array<String>){

    try {

        val data = 20 / 0  //may throw exception

    } catch (e: ArithmeticException) {

        println(e)

    }

    println("code below exception...")

}
```

Output:

```
java.lang.ArithmeticException: / by zero

code below exception...
```

Kotlin try-catch block as an expression:

As we already know, expression always returns a value. We can use kotlin try-catch block as an expression in our program. The value returned by the expression will be either last expression of try block or last expression of catch block. If an exception occurs in the code, then catch block returns the value.

```
fun test(a: Int, b: Int) : Any {
    return try {
        a/b
        //println("The Result is: "+ a / b)
    }
    catch(e:Exception){
        println(e)
        "Divide by zero not allowed"
```

```
    }
}
// main function
fun main(args: Array<String>) {
    // invoke test function
    var result1 = test(10,2  ) //execute try block
    println(result1)
    var result = test(10,0 )   // execute catch block
    println(result)
}
```

In the above code, we have used try-catch as an expression. Declare a function test on the top of program and it return a value using try-catch block. We have invoked the test function from main method and passed the parameter values (10,2) **The test function evaluate the arguments and return try value (10/2 = 5). But in next call, we passed (b=0) and this time exception is caught and returns expression of catch block.**

Kotlin Finally Block:

```
fun main(args : Array<String>){
    try{
        var ar = arrayOf(1,2,3,4,5)
        var int = ar[6]
        println(int)
    }
  finally {
        println("This block always executes")
    }
}
```

Output:

```
java.lang.ArithmeticException: / by zero

This block always executes
```

Kotlin throw keyword:

```
fun main(args: Array<String>) {
    test("abcd")
    println("executes after the validation")
}
fun test(password: String) {
    // calculate length of the entered password and compare
    if (password.length < 6)
        throw ArithmeticException("Password is too short")
    else
        println("Strong password")
}
```

Output:

```
Exception in thread "main" java.lang.ArithmeticException: Password is
too short
```