Scala Try Catch

Scala provides try and catch block to handle exception. The try block is used to enclose suspect code. The catch block is used to handle exception occurred in try block. You can have any number of try catch block in your program according to need.

Scala Try Catch Example

In the following program, we have enclosed our suspect code inside try block. After try block we have used a catch handler to catch exception. If any exception occurs, catch handler will handle it and program will not terminate abnormally.

```
class ExceptionExample{

   def divide(a:Int, b:Int) = {

      try{

         a/b

      }catch{

         case e: ArithmeticException => println(e)

      }

      println("Rest of the code is executing...")

   }

}

object MainObject{

   def main(args:Array[String]){

      var e = new ExceptionExample()

      e.divide(100,0)

   }

}
```

Output:

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

Rest of the code is executing...
```

Scala Try Catch Example 2

In this example, we have two cases in our catch handler. First case will handle only arithmetic type exception. Second case has Throwable class which is a super class in exception hierarchy. The second case is able to handle any type of exception in your program. Sometimes when you don't know about the type of exception, you can use super class.

```
class ExceptionExample{

   def divide(a:Int, b:Int) = {

      try{

         a/b

         var arr = Array(1,2)

         arr(10)

      }catch{

         case e: ArithmeticException => println(e)

         case ex: Throwable =>println("found a unknown exception"+ ex)

      }

      println("Rest of the code is executing...")

   }

}

object MainObject{

   def main(args:Array[String]){

      var e = new ExceptionExample()

      e.divide(100,10)


   }

}
```

Output:

```
found a unknown exceptionjava.lang.ArrayIndexOutOfBoundsException: 10
```

Rest of the code is executing...