Using Exceptions - Java

```
/*Author: Stacey Pegram
Description: This program is a demonstration of error handling as it creates throw
and catch methods.
Summary: If a potential issue occurs when a method is being processed (such as a
delay or fail in the completion of the method), error handling will be incorporated
which can use a try...catch block. Error handling techniques/rules may be added for
the try...catch block in order to set specific actions for different types of errors
detected. If error handling methods are not used, then a program cursor may call the
Main function of a program and processing will restart, a program may fail and close
(crash), or a generic error message may be displayed when an issue occurs in a
program.*/



public class UsingExceptions
{
        public static void main( String[] args )
        {
        try
        {
                method1(); call method
        } //end try
        catch ( Exception exception ) //catch exception thrown
        {
        System.err.printf( "%s\n\n", exception.getMessage() );
        exception.printStackTrace(); //print exception stack trace


/*I think of stacktrace as being similar to a monitoring and logging tool as it will
tack and provide information about methods and other elements involved with
encountered errors*/

StackTraceElement[] traceElements = exception.getStackTrace()

System.out.println( "\nStack trace from getStackTrace:" );
System.out.println( "Class\t\tFile\t\t\tLine\tMethod" );

for ( StackTraceElement element : traceElements )
        {
System.out.printf( "%s\t", element.getClassName() );
System.out.printf( "%s\t", element.getFileName() );
System.out.printf( "%s\t", element.getLineNumber() );
System.out.printf( "%s\t", element.getMethodName() );
        }
}


public static void method1() //throws exception to Main
{
method2();//call method2
}

public static void method2() throws Exception
{
method3();
}

public static void method3() throws Exception
{
throw new Exception( "Exception thrown in method3" );
}
```

Divide By Zero with Exception Handling

```
/*Author: Stacey Pegram
Description: This program is a demonstration of error handling as it creates throw
and catch methods. In this case, the java.util.InputMismatchException class is used
to determine if an error being handled in the try...catch block is a valid integer.
Exceptions (and specified actions set for them) are used for cases where values
entered are not valid integers and when a denominator is equal to zero*/
import java.util.InputMismatchException;import java.util.Scanner;
//ArithmeticException class is in package java lang
public class DivideByZeroWithExceptionHandling
{
        public static int quotient( int numerator, int
denominator )
                throws ArithmeticException //for exception handling
        //specifies the exceptions method should throw
        {
        return numerator / denominator;
        }
public static void main( String[] args )

{
Scanner scanner = new Scanner( System.in );
//scanner object obtaining input

boolean continueLoop = true;

do
{
try //read an int for numerator and an int for a denominator
{
System.out.print( "Please enter an integer numerator: " );
int numerator = scanner.nextInt();
System.out.print( "Please enter an integer denominator: ");
int denominator = scanner.nextInt();

//calculate for quotient
int result = quotient( numerator, denominator );
System.out.printf( "\nResult: %d / %d = %d\n", numerator,
denominator, result ); //displays formula with input values used
                        //continueLoop is false;
}
//catch method uses object created passed into method
catch ( InputMismatchException inputMismatchException )
//prints error message if value input is not an int
{
System.err.printf( "\nException: %s\n",
inputMismatchException );
scanner.nextLine();
System.out.println(
"You must enter integers. Please try again.\n" );
}

//catch method for exception handling of quotient formula
//prints error message when denominator is zero
catch
  {
 System.err.printf( "\nException: %s\n", arithmeticException
  System.out.println(
 "Zero is an invalid denominator. Please try again.\n" );
   }
 }
 }}
```