Koding Exception Handling

a. Tanpa Exception Handling

Koding :

```
/**
 *
 * @author Risa_Ajeng
 *
 */
public class TanpaExceptionHandling {


    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here



{
System.out.println("\n********* PROGRAM TANPA EXCEPTION HANDLING *********\n");
int hasil = 9/0; //penyebab exception
System.out.println("Hasil pembagian = "+hasil);
System.out.println("Pernyataan setelah bebas dari exception.");
}
}
    }
```

b. Menangkap exception dengan blok try-catch

Koding :

```
/**
 *
 * @author Risa_Ajeng
 */
public class TryCatch {


    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    System.out.println("\n********* PROGRAM DENGAN EXCEPTION HANDLING *********\n");
try
{
    int hasil = 9/0; //penyebab exception
    System.out.println("Hasil pembagian = "+hasil);
    System.out.println("Pernyataan dalam blok try setelah bebas dari exception.");
}
    catch(ArithmeticException exc)
{
    System.err.println("ArithmeticException menangkap exception hasil pembagian oleh nol.");
    System.err.println("\nException yang ditangkap adalah : "+exc);
}
System.out.println("\nPernyataan di luar blok try- catch.");
}
}
```

c. Membuat blok try-catch-finally

Koding :

```
/**
 *
 * @author Risa_Ajeng
 */
public class TryCatchFinally {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        System.out.println("\n********* MENGGUNAKAN BLOK TRYCATCH-
FINALLY **********\n");
try
{
int hasil = 9/0; //penyebab exception
System.out.println("Hasil pembagian = "+hasil);
System.out.println("Pernyataan dalam blok try setelah bebas dari exception.");
}
catch(ArithmeticException exc)
{
System.err.println("ArithmeticException menangkap exception hasil pembagian oleh
nol.");
System.err.println("\nException yang ditangkap adalah: "+exc);
}
finally
{
System.out.println("\nPernyataan dalam blok finally.");
}
System.out.println("\nPernyataan di luar blok try-catchfinally.");
```

```
}
}
```

d. Membuat catch secara bertingkat

Koding :

```
/**
 *
 * @author Risa_Ajeng
 */
public class MultipleCatch {


    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        System.out.println("\n********* MENGGUNAKAN MULTIPLE CATCH*********\n");
try
{
int[] array = new int[9]; //deklarasi array berukuran 9 buah elemen
array[9] = 13; //penyebab exception

System.out.println("Elemen array indeks ke 9 adalah"+array[9]);

System.out.println("Pernyataan dalam blok try setelah bebas dari exception.");
```

```
}
catch(ArrayIndexOutOfBoundsException exc)
{
System.err.println("Anda mengakses array di luar indeks yang dideklarasikan.");
}
catch(NegativeArraySizeException exc)
{
System.err.println("Anda mendeklarasikan array dengan ukuran negatif.");
}
catch(Exception exc)
{
System.err.println("Anda melakukan pembagian bilangan oleh nol.");

    }


}
}
```

e. Melemparkan exception dengan klausa throw

Koding :

```
/**
 *
 * @author Risa_Ajeng
 */
public class KlausaThrow {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        String input = "Throw RuntimeException";
System.out.println("\n********* MENGGUNAKAN KLAUSA THROW*********\n");
try
{
if(input.equals("Throw RuntimeException"))
{
throw new RuntimeException("Melempar Exception");
}
else if(input==null)
{
throw new NullPointerException();
}
else
{
System.out.println("Input adalah : "+input);
}
System.out.println("\nPernyataan dalam blok try setelah bebas dari pelemparan exception.");
}
catch(Exception exc)
```

```
{
System.err.println("Exception ditangkap di sini.");
System.err.println("\nException yang ditangkap adalah: "+exc);

    }

}
}
```