→ Internally, Default Exception handler will use printStackTrace() method to print the exception information to the console.

⇒ **TRY With Multiple catch block:**

→ The way of handling an exception is varried from exception to exception. Hence, for every exception type, it is highly recommandate to take separate catch block i.e. try with multiple catch blocks is possible and recommandate to use.

```
try
{
    // Risky code
}
catch (Exception e)
{
}
```

Worst programming practice

```
try
{
    // Risky code
}
catch (ArithmeticException e)
{
    perform alternate arithmetic opn
}
catch (SQLException e)
{
    use mysql db instead of oracle db.
}
catch (FileNotFoundException e)
{
    use local file instead of remote file
}
catch (Exception e)
{
    // default Exception handling
}
```

Best practice programming

⇒ If try with multiple catch blocks presents then the order of catch block is very important. We have to take child first and then parent. Otherwise we will get compile Time Error saying.

Exception xxx has already been caught.

⇒ We can't declare two catch blocks for the same exception otherwise we will get C.T.Error saying Exception xxx has already been caught.

⇒ **final, finally and finalize()**

final :- final is a modifier applicable for classes, methods and variables.

⇒ If a class declared as final then we can't extend that class i.e we can't create child class for that class i.e. Inheritance is not possible for final classes.

⇒ If a method is final then we can't override that method in the child class.

⇒ If a variable declared as final then we can't perform re-assignment for that variable.

⇒ **finally :⇒**

⇒ finally is a block always associated with try-catch to maintain clean up code.

⇒
```
try {
    // Risky code
} catch (Exception e)
{
    // Handling code
} finally {
    clean up code
}
```