Runtime Stack Mechanism :→ (Explained in JVM Architecture)

→ Default Exception handling in Java :→

→ Inside a method, if any exception occures, the method in which it is raised is responsible to create Exception Object by including the following information.

(i) Name of Exception
(ii) Discription " "
(iii) Location at which Exception occures (Stack trace)

→ After creating Exception Object, Method hands overs the Object to the JVM

→ JVM will Cheek whether the method contains any exception handling code or not. If method doesnot contain exception handling code then JVM terminates that method abnormally and removes the corresponding entry from the Stack. Then, JVM Identifies caller method and checks whether caller method contains any exception handling code or not. If the caller method does not contain handling code then JVM terminates that caller method also abnormally and removes corresponding entry from the Stack. This process will be continued until the main method, and if the main method also doesn't contain the handling code then JVM terminats the main method also abnormally and removes corresponding entry from the Stack. Then JVM hand overs responsibility of exception handling to Default Exception

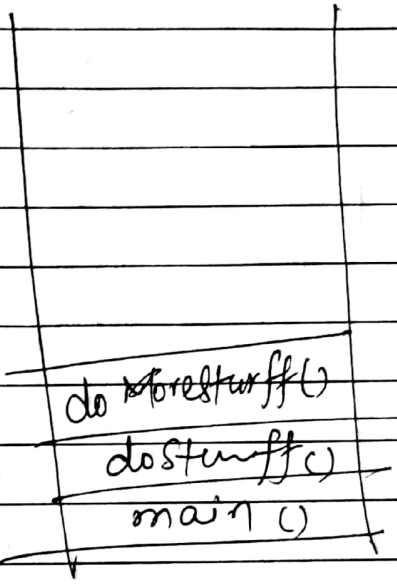
Handler, which is the part of JVM.

→ Default exception handler prints the exception information in the following format and terminates the programme abnormally.

Exception in thread "main" Name of Exception : Description
Stack Trace.

Ex:-

```
public class Test
{
    PSVm(String [] args)
    {
        doStuff();
    }
public static void doStuff()
    {
        doMoreStuff();
    }
p s v doMoreStuff()
    {
        S.O.p(10/0);
    }
}
```



Exception in "main" java.lang.ArithmeticException: / by 0
at Test.doMoreStuff() (line NO)
at Test.doStuff() (line NO)
at Test.main(); (line NO).

⇒ ~~If all the methods~~

⇒ If all methods terminated normally then only prog. termination is Normal termination else Abnormal termination.