→ In try-catch-finally, Order is important.

→ Whenever we are writting try, compulsary we should write either catch or finally otherwise we will get C.T.E i.e. try without catch or finally is invalid.

→ Whenever we are writting catch block, compulsary try block must be required i.e. catch without try is invalid.

→ Whenever we are writting finally block, compulsary we should write try block. i.e. finally without try is invalid.

→ Inside try-catch and finally block, we can declare try-catch and finally blocks i.e, Nesting of try-catch-finally is valid.

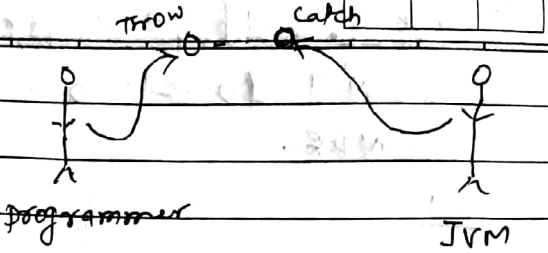
→ For try-catch and finally blocks, curly braces ({, }) are mandatory.

(23)
```
try
    s.o.p("try")
catch(x e)
{   s.o.p("catch");
}
finally
{
}
```
✗

(24)
```
try
{
}
catch(x e)
    s.o.p("catch");
finally
{
}
```
✗

(25)
```
try
{
}
catch(x e)
{
}
finally
    s.o.p("finally");
```
✗

⇒ Throw :→



→ Sometimes, we can create Exception Object explicitly, we can hand over to the JVM manually. For this we have to use throw keyword.

```
throw new ArithmeticException("/ by zero");
```

- `throw` → Hand-over our created exception Object to the JVM manually.
- `new ArithmeticException("/ by zero")` → creation of ArithmeticException Object explicitly.

→ Hence, the main Objective of throw keyword is to hand over our created Exception Object to the JVM manually. Hence, the result of the following 2 programme is exactly same.

```
class Test
{
P s v main(String[] args)
{
    s.o.p(10/0);
}
}
```

```
Exception in Thread "main"
J.l.AE : / by Zero at
    Test.main()
```

⇒ In this case, main method is responsible to create Exception Object and hand over to the JVM.

```
class Test
{
P s v main(String[] args)
{
    throw new AE("/ by zero");
}
}
```

```
Exception in thread "main"
J.l.AE : / by zero at
    Test.main().
```

→ In this case, programmer is responsible creating exception object explicitly and hand over to the JVM manually.

NOTE :- Best use of throw keyword is for user defined or customized exception.