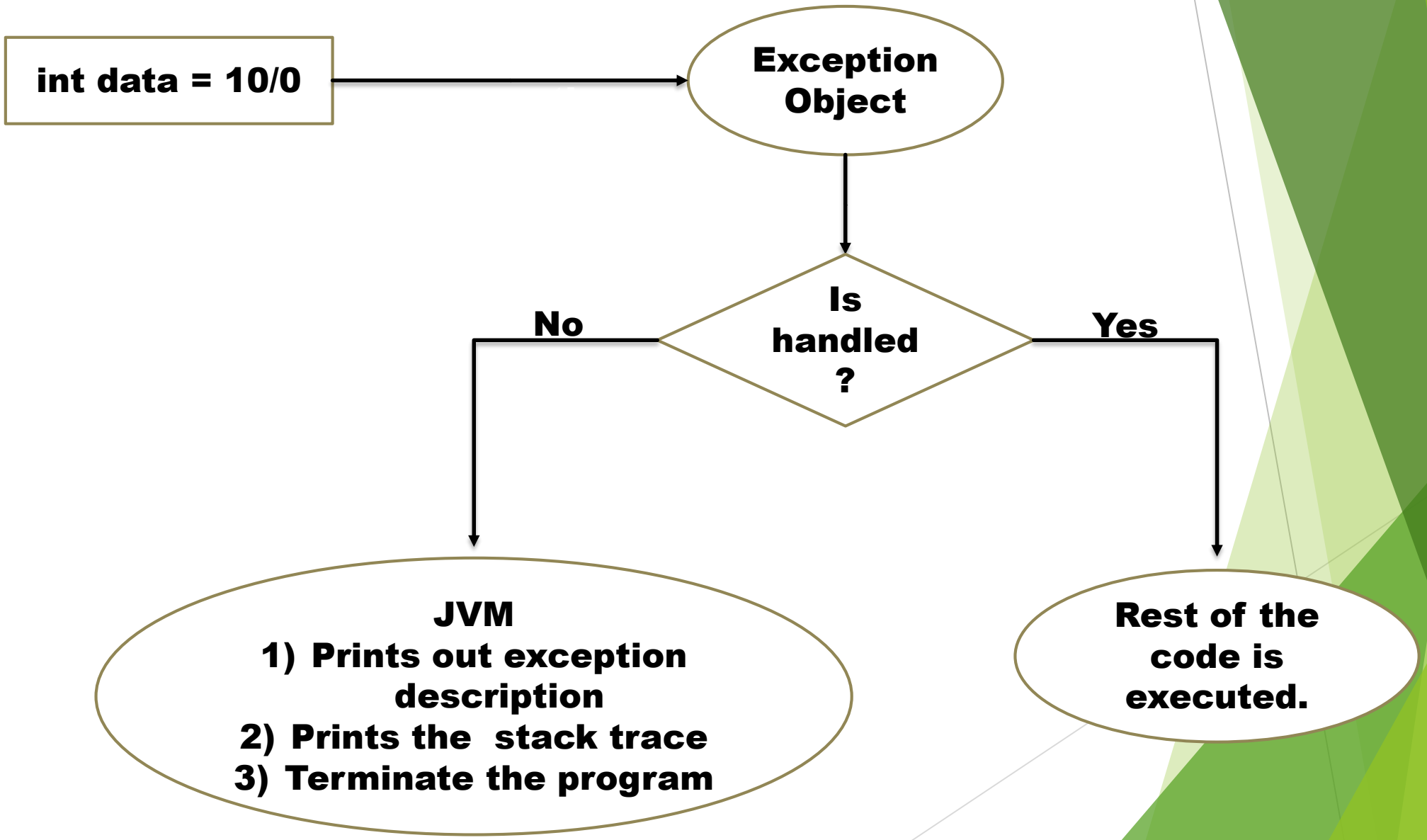
FUNDAMENTALS

A Java Exception is an object that describes an exceptional condition that has occurred in a piece of code.

Java Exception Handling is managed via five keywords :

1) **try**
2) **catch**
3) **throw**
4) **throws**
5) **finally**


int data = 10/0
Exception Object
Is handled ?
No
Yes
JVM
1) Prints out exception description
2) Prints the stack trace
3) Terminate the program
Rest of the code is executed.

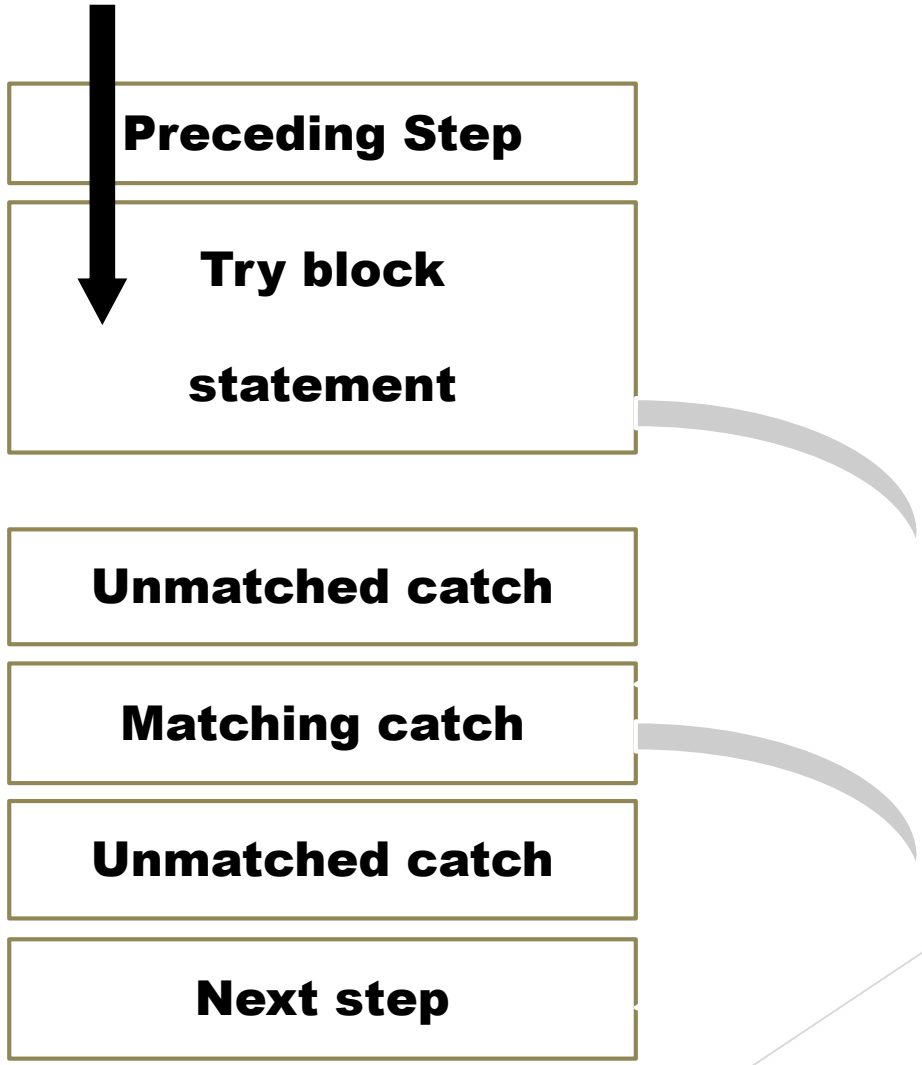
TRY – CATCH BLOCK

```
try {
        //statements that may cause an exception
}
 catch ( excetion(type) e(object) )
{
                //error handling code
}
```

MULTIPLE CATCH

```
try {     // Protected Code
}
catch ( ExceptionType1 e1 ){
          // catch block
}
catch ( ExceptionType1 e1 ){
          // catch block
}
.....
```

SEQUENCE OF EVENTS



```
class Example2 {
                public static void main ( String [] args ) {
try {           int a [] = new int [7] ;
                a [4] = 30 / 0 ;                }
catch ( ArithmeticException e ) {
                System.out.println (“Warning : ArithmeticException”);    }
catch ( ArrayIndexOutOfBoundsException e )  {
                System.out.println(“Warning : ArrayIndexOutOfBoundsException”);
catch ( Exception e )    {
                System.out.println(“Warning : Some other exception”);   }
System.out.println ( “Out of try-catch block ....”);
}
}
```

NESTED TRY CATCH

a) One try-catch block can be present in the another try's body . This is called Nesting of try catch blocks .

b) Each time a try catch block does not have a catch handler for a particular exception, the stack is unwound and the next try block's catch handlers are inspected for a match.

c) If no catch block matches, then the java runtime system will handle the exception.

Syntax of nested try catch

```
try
    {   statement 1;
        try {
                statement 2;     }
        catch ( Exception e1 ) {
                    // Exception Message       }
catch ( Exception e2 )  // catch of parent try block
{       //Exception Message
}
```

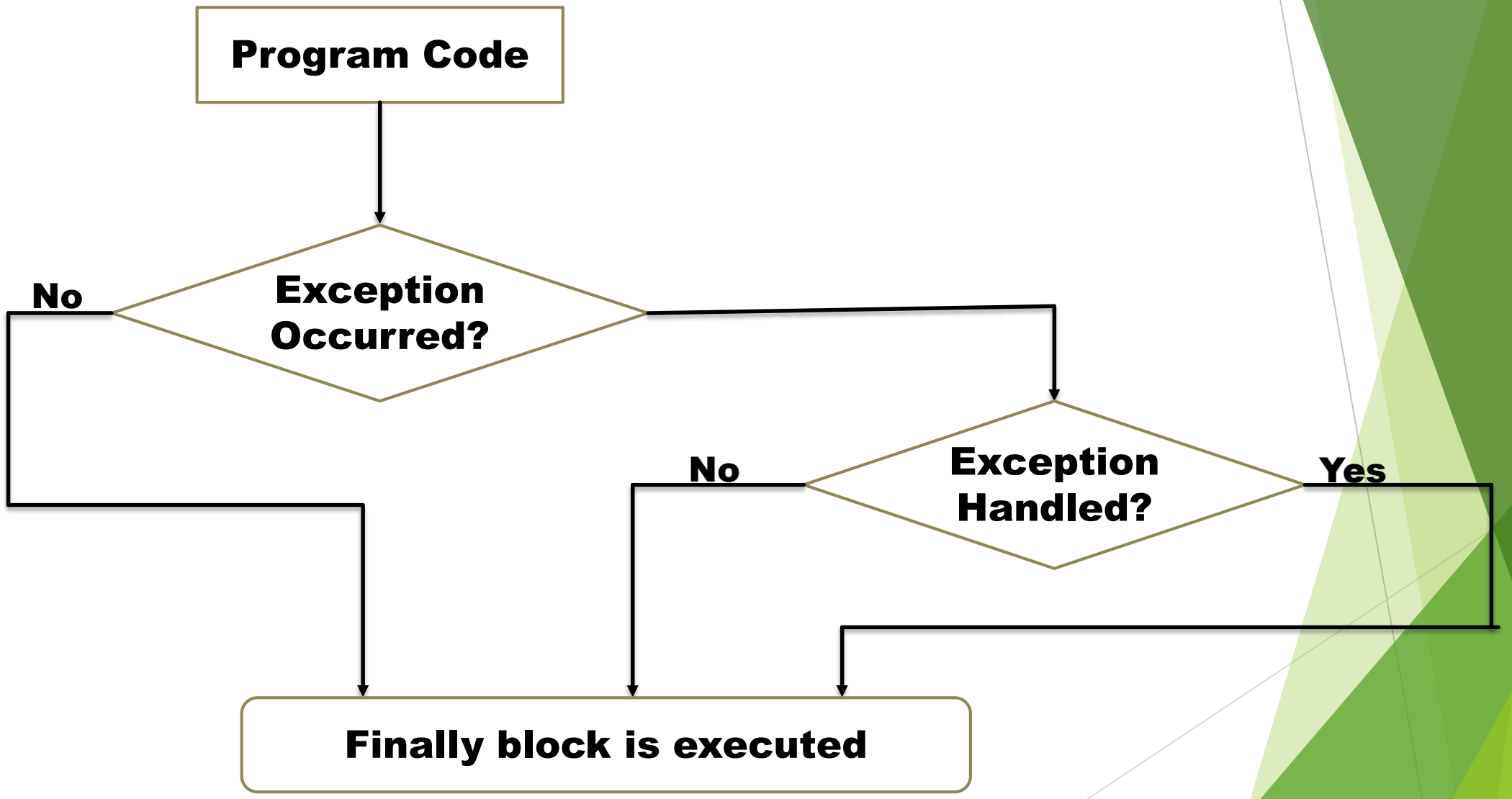
What is finally block

a) **A finally statement must be associated with a try statement.**
b) **It identifies a block of statement that needs to be executed regardless of whether or not an exception occurs within the try block.**
c) **It will run regardless of whether an exception was thrown and handled by the try and catch parts of the block.**

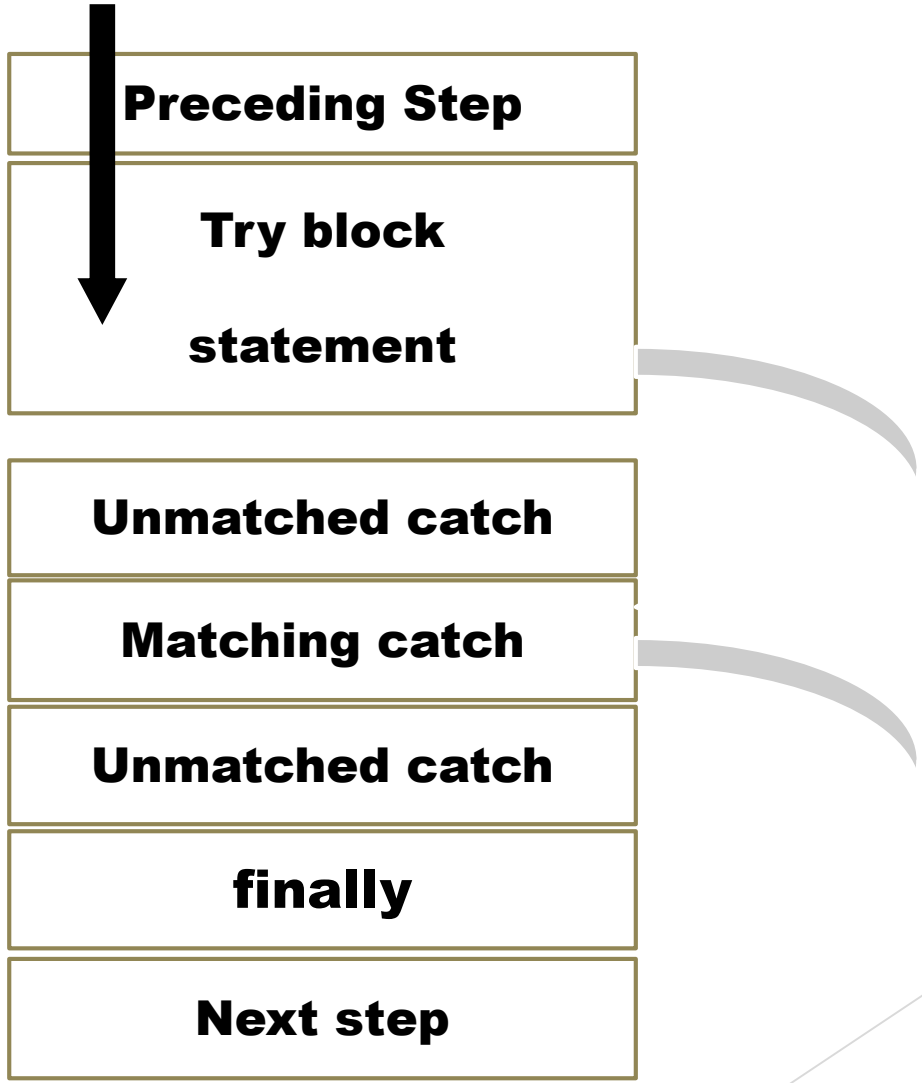
TRY – CATCH – FINALLY

```
try {
.........
}
Finally  {
.........
}
```

```
try {
.......
}
Catch (...) {
.........
}
Finally  {
}
```


Program Code
Exception Occurred?
No
Exception Handled?
No
Yes
Finally block is executed

SEQUENCE OF EVENTS



```
class Simple {
        public static void main (String [ ] args )
  try {
            int data = 25 / 0 ;
            System.out.println(data);
  }
  catch ( ArithmeticException e)   {
                        System.out.prinltn(e);
            }
  finally       {
                    System.out.println(“finally block is always executed”);
            }
  System.out.println(“ rest of the code.....”);
  }}
```

THROWING OUR OWN EXCEPTIONS
THROW KEYWORD

a) In java we have already defined exception classes such as ArithmeticException , NullPointerException etc.

b) These exceptions are implicitly thrown by JVM.

c) The throw keyword is used to explicitly throw an exception.

d) These exceptions are known as user-defined exceptions.

Syntax of throw keyword

```
Throw new AnyThrowableInstance ;

IOException e = new IOException();
throw e;
```

```
class MyException extends Exception    {
        public MyException ( String msg )     {
                                                super( msg );
                }
}
class TestMyException    {
        public static void main (String [ ] args )  {
             int age = -2 ;
        try {
                if (age < 0)
                     throw new MyException (“Age can’t be less than zero”);
                }
        catch (MyException e)     {
                                e.printStackTrace();
                     }
    }
```

THROWS KEYWORD

a) **The throws keyword is used to declare an exception.**

b) **It gives an information to the programmer that there may occur an exception.**

c) **So it is better for the programmer to provide the exception handling code so that normal flow can be maintained.**

Syntax of throws keyword

```
void method_name () throws exception_class_name {
.....                }
```

```
import java.io.* ;

class M    {
        void method () throws IOException   {
                throw new IOException (“device error”);
        }
}

class Test     {
            public static void main (String [] args ) throws IOException {
                Test t = new Test () ;
                t.method () ;
                System.out.println (“normal flow…..”);
            }
}
```

COMPARISON

throw keyword	throws keyword
throw is used to explicitly throw an exception.	throws is used to declare an exception.
checked exception cannot be propagated without throws.	checked exception can be propagated with throws.
throw is followed by an instance.	throws is followed by class.
throw is used within the method.	throws is used with the method signature.
you cannot throw multiple exception.	you can declare multiple exception.
	e.g. public void method () throws IOException, SQLException.