EXCEPTION HANDLING

Exception Handling:

Errors: A programming mistake is said to be an error.

There are three types of errors:

1) Compile time errors (Syntax errors)

2) Runtime errors (Exceptions)

3) Logical errors

Exception means run time error.

In exception handling we use the following keywords.

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

The syntax of try and catch blocks:

```
try
{
      =============      => Task code
}catch(ExceptionClassName ObjectReference)
{
      =============      => Error Message
}
```

try block must be associated with at least one catch block or finally block.

All exceptions are classes in Java.

Whenever exception occurs in a Java program, then the related exception class object is created by JVM, passed to exception handler (catch block) and exception handler code is executed.

Java By Venkatesh Mansani

There are two types of exceptions:

1) Checked Exceptions

2) Unchecked Exceptions

Checked Exceptions:

The exception classes that are derived from java.lang.Exception class are called checked exceptions.

Checked exceptions do not include java.lang.RuntimeException class and all its sub classes.

All checked exceptions must be handled explicitly otherwise compile time error occurs.

The Java compiler checks for try & catch blocks or throws clause for this kind of exceptions.

All application specific exceptions are comes under this category.

Unchecked Exceptions:

The exception classes that are derived from java.lang.RuntimeException class are called unchecked exceptions.

All unchecked exceptions are handled by system implicitly.

Handling unchecked exceptions are optional by programmer.

Unchecked exceptions are handled by programmer to display user friendly error messages only.

The Java compiler does not check for try & catch blocks or throws clause for this kind of exceptions.

All general exceptions comes under this category.

Java By Venkatesh Mansani

Example:

```
class Demo
{
	public static void main(String args[])
	{
		try{
		int x=Integer.parseInt(args[0]);
		int y=Integer.parseInt(args[1]);
		int z=x/y;
		System.out.println(z);
		}catch(ArrayIndexOutOfBoundsException ae)
		{
		    System.err.println("Please pass two arguments");
		}catch(NumberFormatException ne)
		{
				System.err.println("Please pass two numbers only");
		}catch(ArithmeticException ae)
		{
		System.err.println("Please pass second argument except zero");
		}
	}
}
```

Java By Venkatesh Mansani

Execution:

1) C:\> java Demo

Please pass two arguments

2) C:\> java Demo abc xyz

Please pass two numbers only

3) C:\> java Demo 10 0

Please pass second argument except zero

4) C:\> java Demo 10 2

5

Program to create & handle checked exception:

```
class NegativeNumberException extends Exception
{
}
class Demo
{
      void cube(a) throws NegativeNumberException
      {
            if(a>0)
              System.out.println(a*a*a);
            else
              throw new NegativeNumberException();
  }
  public static void main(String args[])
  {
```

Java By Venkatesh Mansani

```
        try{
        int x=Integer.parseInt(args[0]);
        Demo d=new Demo();
        d.cube(x);
        }catch(NegativeNumberException ne)
        {
                System.err.println(ne);
        }
}
}
```

Program to create unchecked exception:

```
class NegativeNumberException extends RuntimeException
{
}
class Demo
{
        void cube(a) throws NegativeNumberException
        {
             if(a>0)
               System.out.println(a*a*a);
             else
                throw new NegativeNumberException();
        }
        public static void main(String args[])
```

Java By Venkatesh Mansani

```
    {
        int x=Integer.parseInt(args[0]);
        Demo d=new Demo();
        d.cube(x);
    }
}
```

Try & Catch Blocks Examples:

1) try
```
{
}catch(Exception e)
{
}
```
The above catch block handles all exceptions because all exceptions are sub classes of java.lang.Exception class.

2) try
```
{
}catch(RuntimeException e)
{
}
```
The above catch block handles all unchecked exceptions because all unchecked exceptions are sub classes of java.lang.RuntimeException class.

3) try
```
{
}catch(IOException | InterruptedException e)
{
}
```

Java By Venkatesh Mansani

The above catch block handles IOException, InterruptedException, sub classes of IOException & sub classes of InterruptedException.

This concept is called as handling multiple exceptions with single catch blocks.

This concept is introduced in JDK 1.7 version in 2011.

throw keyword:

It is used to pass an object of exception class to a catch block.

throws keyword:

It is used to apply an exception to a method and it is also used to handle exception.

finally block:

It is used to perform cleanup activities.

Cleanup activities are closing a file, closing a database connection, closing a socket, .. etc.,

finally block is executed even exception occurs in a program.

Java By Venkatesh Mansani

By

Mr. Venatesh Mansani

Naresh i Technologies