Day 31

Q. Write a Java program to illustrate multiple catch block using command line argument.

Answer:

The catch block is used to handle the exception which is raised in try block. A single try block may contain more than one catch block. Below example shows how to use to multiple catch block.

```
class Check_Exception
{
        public static void main(String[] args)
        {
                try
                {
                        int a = Integer.parseInt(args[0]);
                        int b = Integer.parseInt(args[1]);
                        int c = a / b;
                        System.out.println("Result: "+c);
                }
                catch (ArithmeticException ae)
                {
                        System.out.println("Enter second number except zero.");
                }
                catch(ArrayIndexOutOfBoundsException ai)
                {
                        System.out.println("Both numbers are required.");
                }
                catch(NumberFormatException ne)
                {
                        System.out.println("Enter only integer value.");
                }
                finally
                {
                        System.out.println("Finally Block");
                }
        }
}
```

Q. Write a program to illustrate the throws keywords in Java.

Answer:

Throws is used for method declaration and denotes which exception can be thrown by this method. This example shows how checked exception propagate by throws keyword.

```
public class ThrowsDemo
{
     static void throwMethod1() throws NullPointerException
     {
          System.out.println ("Inside throwMethod1");
```

```
        throw new NullPointerException ("Throws_Demo1");
    }
    static void throwMethod2() throws ArithmeticException
    {
        System.out.println("Inside throwsMethod2");
        throw new ArithmeticException("Throws_Demo2");
    }
    public static void main(String args[])
    {
        try
        {
            throwMethod1();
        }
        catch (NullPointerException exp)
        {
            System.out.println ("Exception is: " +exp);
        }
        try
        {
            throwMethod2();
        }
        catch(ArithmeticException ae)
        {
            System.out.println("Exception is: "+ae);
        }
    }
}
```

Q. Write a program to demonstrate the chained exception in Java.

Answer:

Chained Exception:
It allows to relate one exception with another exception. i.e. one exception describes the cause of another exception.

```
public class ChainedException
{
        public static void main(String[] args)
        {
                try
                {
                        Exception ae = new Exception("Exception");
                        ae.initCause(new ArithmeticException("Cause of the ArithmeticException"));
                        throw ae;
                }
                catch (Exception ae)
                {
                        System.out.println(ae);
                        System.out.println(ae.getCause());
        }
        try
        {
            NumberFormatException ex = new
NumberFormatException("NumberFormatException");
            ex.initCause(new NullPointerException("Cause of the NullPointerException"));
            throw ex;
        }
        catch(NumberFormatException ex)
        {
            System.out.println(ex);
            System.out.println(ex.getCause());
        }
    }
}
```

Q. Write a program to create custom exception in java.

Answer:

Exception class is defined in Java library. But we can also create our own exception. This example of exception, shows how to create a custom exception and use it.

```
public class CustomException extends Exception
{
    public CustomException(String msg)
    {
        super(msg);
    }
}
```

```
public class CustomDemo
{
    public static void main(String args[]) throws Exception
    {
        CustomDemo custom = new CustomDemo();
        custom.display();
    }
    public void display() throws CustomException
    {
        for(int i=2;i<20;i=i+2)
        {
            System.out.println("i= "+i);
            if(i==8)
            {
                throw new CustomException("My Exception Occurred");
            }
        }
    }
}
```