CS F213
Object Oriented Programming

BITS Pilani
Hyderabad Campus

Prof.R.Gururaj
CS&IS Dept.

Exception Handling

Ch.10 of R1.

The Complete Reference- Java, 7th Edition, Herbert Schildt, Tata McGraw Hill Publishing.

OR Ch.10 of T2 Complete Reference 11th Edition.

And also refer to Class notes.

Content

innovate achieve lead

1. Exception Handling Fundamentals
2. Exception Types in Java
3. Use of try-catch
4. Nested try statements
5. Keywords- throw, throws, finally
6. Creating own exception subclasses
7. Examples

Exception

Exception

Is an abnormal condition that arises in a code sequence at run time.

It is an error condition.

Exception handling in Java

This brings run-time error handling mechanism into object-oriented world.

Java exception is an object that describes the error condition that has occurred in a piece of code.

Code throws appropriate exception object and its caught and processed properly.

Ex:

```
class Exc{
        public static void main(String args[ ]) {
            int d,a;
            d=0;    a=42 / d;
            System.out.println("Divide 42 by zero.");
         }
}
```

```
java.lang.ArithmeticException: / by zero
        at Exc.main(Exc.java: 4)
```

Java uses "*try- catch – throw – throws – finally*" keywords.

Program Statements that you want to monitor for exception are contained within a '***try***' block.

Your code can catching exception using '***catch***' and handle it in some rational manner.

To manually throw an exception, use the key word '***throw***'.

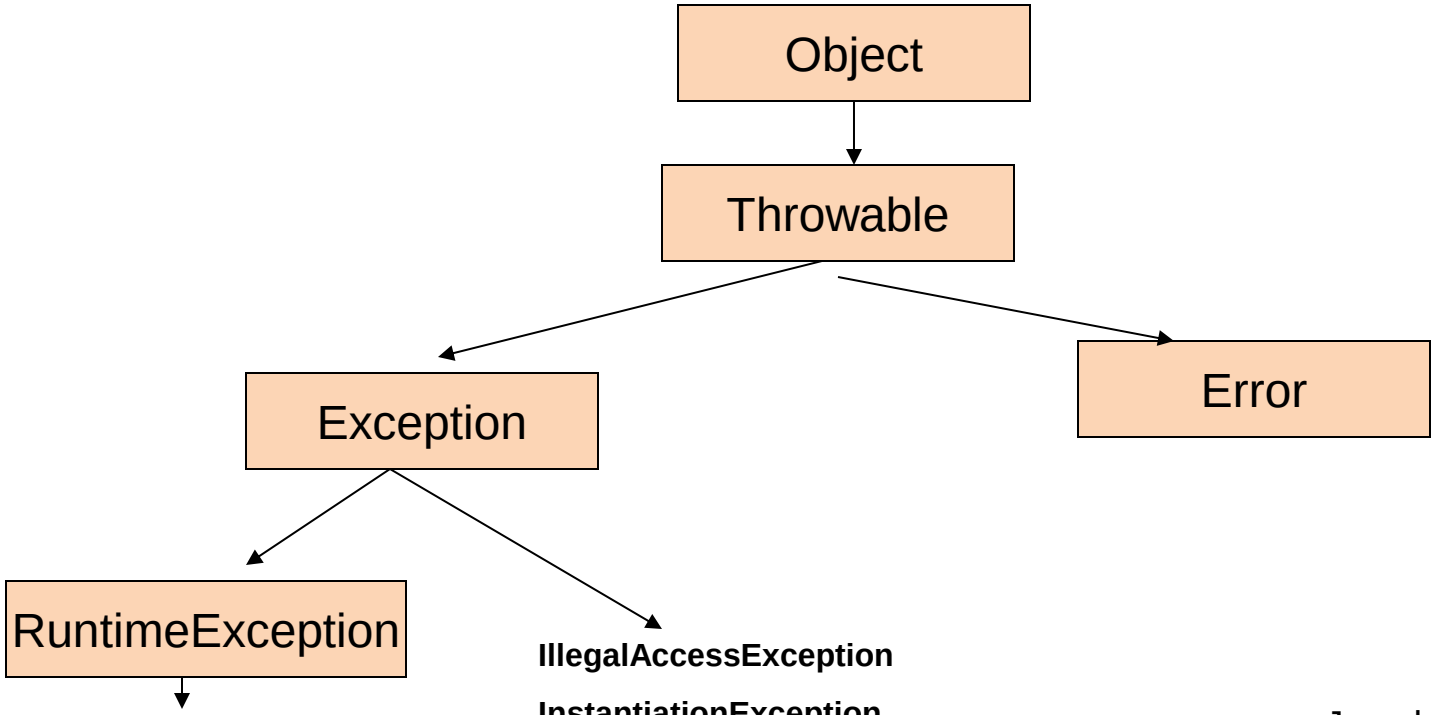
Any exception thrown out of a method must be specified as such by a '***throws***' clause.

Any code that absolutely must be executed before a method returns is put in a '***finally***' block.

The general form of Exception handling in Java

```
{
    try{
            // code expected to throw an exception
    }
    catch(Exception type1){
            // exception handler for type1
    }
    catch (Exception type2){
            // exception handler for type2
    }
    finally{  // code to be executed before try block ends
    }
}
```

The Exception types in Java



Java.lang package

All exception types are subclasses of built-in class ***Throwable*** which is subclass of ***Object*** .

Class 'Trowable' has two subclasses that partition exceptions into two distinct branches.

One branch is headed by ***Exception***. This class is used for exceptional conditions that user programs should catch.

A subclass of ***Exception,*** is called ***RuntimeException***. Exceptions of this type are automatically defined for the programs that you write and include things such as division by zero and invalid array indexing.

Exceptions of type **Error** are used by the Java run-time system to indicate errors having to do with the run-time environment, itself. Stack overflow is an example of such an error.

Exception objects should be throwable i.e, the class called ***Throwable*** should be in the hierarchy of Exception class.

User created exception are generally subclasses of class ***Exception.***

Using *try* and *catch*

```
try{
        Statement 1;
        Statement 2;
}
catch(Exception e)
{
        exception handler code ;
}
```

The ***catch*** should be immediately after ***try***.

There should be no statements in between two, else it is an error.

Exception object is said to be thrown out of ‘try’ block which is caught in the ‘catch’ block.

The control does not return to try because we are not calling ‘catch’ as a method.

Sample

```
class ExDemo1
{    public static void main(String args[ ])
     {   int d,a;
         try{
                        d=0;
                        a=42 / d;
                        System.out.println("Divide 42 by zero.");
          }
          catch (ArithmeticException e)
          { System.out.println("Division by zero is not allowed");
           }
          System.out.println("After catch statement.");
      }
}
```

```
C:\Users\Admin\JavaPrograms>java ExDemo1
Division by zero is not allowed
After catch statement.
```

```
class ExDemo2
{
    public static void main(String args[ ])
    {
        int a[]={20,40,60,80};

        System.out.println("last element in array is :"+a[4]);
       System.out.println("last Statement");
       }
}
```

```
C:\Users\Admin\JavaPrograms>java ExDemo2
Exception in thread "main"
    java.lang.ArrayIndexOutOfBoundsException: 4
        at ExDemo2.main(ExDemo2.java:8)
```

```
class ExDemo3
{
      public static void main(String args[ ])
      {
          int a[]={20,40,60,80};
                try{
               System.out.println("last element in array is :"+a[4]);
                System.out.println("\nlast statement in try ");
                }
                catch(Exception e)
                {
                 System.out.println("\n I handle the Exception on my own");
                }
                finally
                {
                System.out.println("Leaving try block bye..");
                }
                System.out.println("\nlast statement in program :");
       }
}
```

```
C:\Users\Admin\JavaPrograms>java ExDemo3

 I handle the Exception on my own:
Leaving try block bye..

last statement in program :
```

Ex

```
class ExDemo4
{    public static void main(String args[ ])
     {    int a[]={20,40,60,80};
          try{
              System.out.println("last element in array is :"+a[4]);
              System.out.println("\nlast statement in try ");
              }
              catch(ArithmeticException ae)
              {  System.out.println("\n I handle this array index out Exception \n:");
               }
              finally
              {System.out.println("In finally block code..");
              }
              System.out.println("\nlast statement in main program :");
      }
}
```

```
C:\Users\Admin\JavaPrograms>java ExDemo4
In finally block code..
Exception in thread "main"
    java.lang.ArrayIndexOutOfBoundsException: 4
        at ExDemo4.main(ExDemo4.java:8)
```

A ***try*** without ***catch*** or ***finally*** is an error at compile-time.

```
class ExDemo1A
{
     public static void main(String args[ ])
     {
         int d,a;
         try{
                          d=0;
                          a=42 / d;
                          System.out.println("Divide 42 by zero.");
        }
       System.out.println("After catch statement.");
        }
}
C:\Users\Admin\JavaPrograms>javac ExDemo1A.java
ExDemo1A.java:7: error: 'try' without 'catch', 'finally' or resource declaration
s
         try{
         ^
1 error
```

A *try* without *catch* and only with *finally* is OK.

```
class ExDemo1A
{
	public static void main(String args[ ])
	{
	    int d,a;
	   try{
	                        d=0;   a=42 / d;
	                        System.out.println("Divide 42 by zero.");
	   }
	   finally
	   {System.out.println("In finally ");}
	    System.out.println("After catch statement.");
	}
}
```

```
C:\Users\Admin\JavaPrograms>java ExDemo1A
In finally
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at ExDemo1A.main(ExDemo1A.java:9)
```

A ***finally*** without ***try*** will not be compiled.

A ***catch*** without ***try*** will not be compiled.

Multiple catches for a '*try*' block

If there is a single catch for a try block, the exception handling will be generalized.

Different types of exceptions should be handled in different ways i.e., more than one *catch* block for a single *try*.

Ex

```
class EX{
    method(){
      //code
      try{
         //code expected to throw exception.
      }
      catch(     ){
            //handler.
      }
      catch(  ) {
         //handler.
      }
    }
}
```

```
class ExDemo5
{     public static void main(String args[ ])
      {     int a[]={20,40,60,80};
                try{System.out.println("last element in array is :"+a[4]);
                 System.out.println("\nlast statement in try ");
                 }
                 catch(ArithmeticException ae)
                 {
                  System.out.println("\n I handle this Arithmetic Exception \n");
                 }
                 catch(ArrayIndexOutOfBoundsException aie)
                 {
                  System.out.println("\n I handle this array index out Exception \n");
                 }
                 finally
                 {
                 System.out.println("In finally block code.");
                 }
                 System.out.println("\nlast statement in main program :");
       }
}
```

C:\Users\Admin\JavaPrograms>java ExDemo5

I handle this array index out Exception

In finally block code.

last statement in main program :

```
class ExDemo6
{
    public static void main(String args[ ])
    {
        boolean b=true; boolean a=false;

        if (b){ System.out.println("\n It is TRUE");}
        else{System.out.println("\n It is FALSE");}
        if (b==true  & a==false)
        {System.out.println("\n It is OK");}
       }
}
```

```
C:\Users\Admin\JavaPrograms>java ExDemo6

It is TRUE

It is OK
```

```
class ExDemo7
{
    public static void main(String args[ ])
    {
        boolean b=true; boolean a=false;
       if (b){ System.out.println("\n It is TRUE");}
        else{System.out.println("\n It is FALSE");}
        if (b==true  & a==false)
        {System.out.println("\n It is OK");}
        if (b==true  & b==false)
        {System.out.println("\n It is NOT Correct");}
      }
}
```

```
C:\Users\Admin\JavaPrograms>java ExDemo7

It is TRUE

It is OK
```

```
class ExDemo8
{
    public static void main(String args[ ])
    {
        boolean b=true;

        if (b){ System.out.println("\n It is TRUE");return;}
        else{System.out.println("\n It is FALSE");return;}

        System.out.println("\n Last statemant");
      }
}
```

```
C:\Users\Admin\JavaPrograms>javac ExDemo8.java
ExDemo8.java:10: error: unreachable statement
            System.out.println("\n Last statemant");
            ^
1 error
```

In multiple catch the catch block which can catch all types exceptions must be put in the end because if non of the catch blocks catches the exception then only it should catch it.

It is important to remember that Exception subclasses must come before their super class.

This is because a catch statement that uses a super class will catch exceptions of that type and all its subclasses. Thus a subclass would never be reached, if it comes after the super class.

In java un reachable code is an error.

```
class ExDemo{
        public static void main(String org[])
       {     //code
            try{
                        //code expected to throw arithmetic exception.
                }
            catch(ArithmeticException e )
            {           //handler.            }
             catch(ArrayIndexOutOfBoundsException aie)
            {           //handler.            }
            catch(Exception e )
            {   //handler.                    }
    }
}
```

```
class ExDemo9
{     public static void main(String args[ ])
      {
          int a[]={20,40,60,80};
              try{
                 System.out.println("last element in array is :"+a[4]);
                System.out.println("\nlast statement in try ");
                }
                catch(Exception e)
                {  System.out.println("\n I handle this  Exception \n:");}
                catch(ArithmeticException ae)
                {  System.out.println("\n I handle this Arithmetic Exception \n:");}
                catch(ArrayIndexOutOfBoundsException ae)
                {  System.out.println("\n I handle this array index out Exception \n:");    }
                finally
                {System.out.println("In finally block code..");}
                System.out.println("\nlast statement in main program :");
       }
}
```

```
C:\Users\Admin\JavaPrograms>javac ExDemo9.java
ExDemo9.java:15: error: exception ArithmeticException has already
    been caught
                catch(ArithmeticException ae)
                ^
ExDemo9.java:19: error: exception ArrayIndexOutOfBoundsException
    has already been caught
                catch(ArrayIndexOutOfBoundsException ae)
                ^
2 errors
```

```
class ExDemo9
{     public static void main(String args[ ])
      {
          int a[]={20,40,60,80};
              try{
                 System.out.println("last element in array is :"+a[4]);
                  System.out.println("\nlast statement in try ");
               }
               catch(ArithmeticException ae)
               {  System.out.println("\n I handle this Arithmetic Exception \n:");}
               catch(ArrayIndexOutOfBoundsException ae)
               {  System.out.println("\n I handle this array index out Exception \n:");    }
               catch(Exception e)
               {  System.out.println("\n I handle this  Exception \n:");}
               finally
               {System.out.println("In finally block code..");}
               System.out.println("\nlast statement in main program :");
       }
}
```

C:\Users\Admin\JavaPrograms>java ExDemo10

I handle this array index out Exception

In finally block code..

last statement in main program :

```
class ExDemo11
{     public static void main(String args[ ])
      {
            int a[]={20,40,60,80};
            try{
               System.out.println("last element in array is :"+a[4]);
               System.out.println("\nlast statement in try ");
               }
               try{ System.out.println("last element in array is :"+a[4]);}
               catch(ArrayIndexOutOfBoundsException ae)
               {  System.out.println("\n I handle this array index out Exception \n:");    }
               catch(Exception e)
               {  System.out.println("\n I handle this Generic Exception \n:");}
               finally
               {
               System.out.println("In finally block code..");
               }
               System.out.println("\nlast statement in main program :");
      }
}
```

```
C:\Users\Admin\JavaPrograms>javac ExDemo11.java
ExDemo11.java:6: error: 'try' without 'catch', 'finally' or resource
    declarations
            try{
            ^
1 error
```

```
class ExDemo12
{    public static void main(String args[ ])
     {       int a[]={20,40,60,80};
             try{
               System.out.println("last element in array is :"+a[4]);
               System.out.println("\nlast statement in try ");
               }
               catch(ArrayIndexOutOfBoundsException ae)
               {  System.out.println("\n I handle this array index out Exception \n:");
               }
               finally
               {System.out.println("In finally block code..");
               }
               try{ System.out.println("last element in array is :"+a[4]/0);
               }
               System.out.println("\nlast statement in main program :");
      }
}
```

```
C:\Users\Admin\JavaPrograms>javac ExDemo12.java
ExDemo12.java:18: error: 'try' without 'catch', 'finally' or resource
    declarations
            try{ System.out.println("last element in array is :"+a[4]/0);}

1 Error      ^
```

```
class ExDemo13
{    public static void main(String args[ ])
     {      int a[]={20,40,60,80};
             try{
               System.out.println("last element in array is :"+a[4]);
               System.out.println("\nlast statement in try ");
               }
               catch(ArrayIndexOutOfBoundsException ae)
               {  System.out.println("\n I handle this array index out Exception \n:");     }
               finally
               {System.out.println("In finally block code..");}
               try{ System.out.println("last element in array is :"+a[2]/0);}
               catch(ArithmeticException ae)
               {  System.out.println("\n I handle this Arithmetic Exception \n:");}
               System.out.println("\nlast statement in main program :");
      }
}
```

```
C:\Users\Admin\JavaPrograms>java ExDemo13

 I handle this array index out Exception
:
In finally block code..

 I handle this Arithmetic Exception
:

last statement in main program :
```

Nesting *try*

The ***try***s can be nested.

In case of nested try any exception object is thrown in inner try block and there is no matching *catch* for that particular exception , then it searches for a *catch* block after the outer *try* block.

Ex

```
class NestedTryDemo
{
    public static void main(String org[])
    {
        try{
                int b=100;
                System.out.println("outer try");
                try{        int d=b/0;              }
               catch(ArrayIndexOutOfBoundsException e)
               {   System.out.println(e);   }
               System.out.println("end of inner try");
         }
        catch(Exception e)
         {        System.out.println(e);     }
        System.out.println(“Last statement in main metod:”);
    }
}
```

C:\Users\Admin\JavaPrograms>java ExDemo14
outer try
java.lang.ArithmeticException: / by zero

Last statement in main metod:

```
class ExDemo15
{
    public static void main(String org[])
    {
        try{        int b=100;
                    System.out.println("outer try");
                    try{        int d=b/0;              }
                    catch(ArrayIndexOutOfBoundsException e)
                    {   System.out.println(e +"\n");   }
                    finally{System.out.println("In the finally block of inner try");}
                    System.out.println("end of inner try");
         }
        catch(Exception e)
        {  System.out.println(e+"\n");}
        finally{System.out.println("In the finally block of outer try");}
        System.out.println("Last statement in main metod:");
    }
}
```

```
C:\Users\Admin\JavaPrograms>java ExDemo15
outer try
In the finally block of inner try
java.lang.ArithmeticException: / by zero

In the finally block of outer try
Last statement in main metod:
```

```
class ExDemo16
{
    public static void main(String org[])
    {
        int a[]={20,40,60,80};
        try{
            int b=100;
            System.out.println("start outer try");
            try{        a[5]=400;   System.out.println("in nested try\n");          }
            catch(ArrayIndexOutOfBoundsException e)
            {   System.out.println(e +"\n");   }
            finally{System.out.println("In the finally block of inner try");}
            System.out.println("end of inner try");
         }
        catch(Exception e)
        {  System.out.println(e+"\n");}
        finally{System.out.println("In the finally block of outer try");}
        System.out.println("Last statement in main metod:");
    }
}
```

```
C:\Users\Admin\JavaPrograms>java ExDemo16
start outer try
java.lang.ArrayIndexOutOfBoundsException: 5

In the finally block of inner try
end of inner try
In the finally block of outer try
Last statement in main metod:
```

```
class ExDemo17
{
    public static void main(String org[])
    {
        int a[]={20,40,60,80};
        try
        { a[5]=400;
        System.out.println("in  try\n");          }
        catch(ArrayIndexOutOfBoundsException e)
        {   System.out.println(e +"\n");
            a[7]=800;
        }
        finally{System.out.println("In the finally block of ");}
        System.out.println("Last statement in main metod:");
    }
}
```

```
C:\Users\Admin\JavaPrograms>java ExDemo17
java.lang.ArrayIndexOutOfBoundsException: 5

In the finally block
Exception in thread "main"
    java.lang.ArrayIndexOutOfBoundsException: 7
        at ExDemo17.main(ExDemo17.java:11)
```

```
class ExDemo17
{
    public static void main(String org[])
    {       int a[]={20,40,60,80};
            try
            { a[5]=400;   System.out.println("in nested try\n");          }
            catch(ArrayIndexOutOfBoundsException e)
            {    System.out.println(e +"\n");
                 a[7]=800;  System.out.println("last in catch \n");
            }
            finally{System.out.println("In the finally block");}
            System.out.println("Last statement in main metod:");
    }
}
```

```
C:\Users\Admin\JavaPrograms>java ExDemo17
java.lang.ArrayIndexOutOfBoundsException: 5

In the finally block
Exception in thread "main"
    java.lang.ArrayIndexOutOfBoundsException: 7
        at ExDemo17.main(ExDemo17.java:11)
```

Using *throw*

```
try{
        ------;
        throw exceptionobject;
        }
catch( exception )
{
        //handler.
}
```

```
class ExDemo18
{
    public static void main(String orgs[])
    {
            int a;
            a= Integer.parseInt(args[0]);
            a=a*2;
            System.out.println("value is :"+a);
    }
}
```

```
class ExDemo18
{
    public static void main(String args[])
    {
            int a;
            a= Integer.parseInt(args[0]);
            a=a*2;
            System.out.println("value is :"+a);
    }
}
```

C:\Users\Admin\JavaPrograms>java ExDemo18 20

value is :40

```
C:\Users\Admin\JavaPrograms>java ExDemo18 ab
Exception in thread "main" java.lang.NumberFormatException: For input
    string: "ab"
        at
    java.lang.NumberFormatException.forInputString(NumberFormatExcepti
    on. java:65)
        at java.lang.Integer.parseInt(Integer.java:580)
        at java.lang.Integer.parseInt(Integer.java:615)
        at ExDemo18.main(ExDemo18.java:6)
```

```
class ExDemo19
{
    public static void main(String args[])
    {
            int a;
            a= Integer.parseInt(args[0]);
            try{
            if (a >100) throw new Exception();
            }
            catch(Exception e){
            System.out.println("Exeption is:"+e);}
            System.out.println("\n Last in main");
    }
}
```

```
C:\Users\Admin\JavaPrograms>java ExDemo19  120
Exeption is:java.lang.Exception

 Last in main
```

```
class ExDemo19
{
    public static void main(String args[])
    {       int a;
            a= Integer.parseInt(args[0]);
            try{
            if (a >100) throw new Exception();
            }
            catch(Exception e){ a=100;}
            a=a*2;
            System.out.println("\n Doubled Value of a is:"+a);
    }
}
```

C:\Users\Admin\JavaPrograms>java ExDemo19 130

Doubled Value of a is:200

```
C:\Users\Admin\JavaPrograms>java ExDemo19  50

Doubled Value of a is:100
```

Use of *throws*

If a method is capable of causing an exception that it does not handle, it must specify this behavior so that callers of the method can guard themselves against that exception.

We do this by including a ***throws*** clause in the method's decleration.

A throws clause lists the types of exceptions that a method might throw.

```
void method(  ) throws ABCException
   {//ABC exception can be thrown which is not handled
   here.
                  ------ ;
   }


    try {
                  method( );
        }
        catch( ABCException e ){
                  // handle the Exception e.
        }
```

```
class ExDemo20
{
	public static void main(String args[])
	{
		Line lin= new Line(100);
		lin.doubleLength();
	}
}

class Line
{
	int length;
	Line(int l)
	{length=l;}
	void doubleLength()
	{
	 if(length>100) throw new Exception();
	  System.out.println("\n Doubled Value of a is:"+length*2);
	 }
}
```

```
C:\Users\Admin\JavaPrograms>javac ExDemo20.java
ExDemo20.java:18: error: unreported exception Exception;
    must be caught or declared to be thrown
        if(length>100) throw new Exception();
                       ^
1 error
```

```
class ExDemo20
{
	public static void main(String args[])
	{
		Line lin= new Line(100);
		try{  lin.doubleLength();   }
		catch(Exception e){System.out.println("\n Sorry Value is greater than 100");}

	}
}

class Line
{
	int length;
	Line(int l)
	{length=l;}
	void doubleLength() throws Exception
	{
	 if(length>100) throw new Exception();
	  System.out.println("\n Doubled Value of a is:"+length*2);
	}
}
```

```
C:\Users\Admin\JavaPrograms>java ExDemo20

Doubled Value of a is:200
```

C:\Users\Admin\JavaPrograms>java ExDemo20

Sorry Value is greater than 100

Unchecked exceptions:

Inside the standard package java.lang, Java defines several exception classes.

Unchecked exceptions: They need not be included in any method's throws list.

The compiler does not check to see if a method handles or throws these exception.

Ex:

ArithmeticException

ArrayIndexOutOfBoundException

ArrayStoreException

ClassCastException

NullPointerException

NumberFormatException

Etc.,

Checked_exceptions:

innovate achieve lead

These must be included in a method's throws list if that method can generate one of these exceptions and does not handle it, itself.

Ex:

ClassNotFoundException

IllegalAccessException

InstantiationException

InterruptedException Etc.

User Created Exceptions

User can create his own exception classes provided the classcalled ***Throwable*** is present in the hierarchy.

Generally we create exceptions by subclassing the class called ***Exception***.

```
class InvalidAgeException  extends Exception
{
      int s;
      InvalidAgeException()
     {
       s=333;
     }
     public String toString()
     {
       return    s+": Invalid Age Exception: Age Meaningless:";
     }

}
```

```
class ExDemo21                    // Program to test InvalidAgeException
{     public static void main(String a[])
      {
        try
          {
             insertAge(11);
           }
       catch(InvalidAgeException iae)
       {
           System.out.println("caught"+iae);
        }
      }
       static void insertAge(int a) throws InvalidAgeException
      {
          if(a<1 ||  a>100)
                throw new InvalidAgeException();
        else
                System.out.println("Inserted Data for Age:"+a);
      }
}
```

```
C:\Users\Admin\JavaPrograms>java ExDemo21
Caught  333: Invalid Age Exception: Age Meaningless:
```

Summary

Exception handling Java

Try catch finally

Throw

Throws

Exception hierarchy

Nested try blocks

User Defined Exceptions

Examples