Exception Handling in JAVA

Some errors can halt the execution of a program and the program is terminated from the statement where the error has occurred.These run time errors are called exceptions.Programmer has to handle the exceptions to have a smooth execution of a program.

Some situations where exception may come are

1. Trying to divide a number by zero
2. Try to read from a file which does not exists
3. The file which we are trying to open does not exists
4. Trying to print an element of an array which is not there etc…

try and catch block

In JAVA exceptions can be handled by try and catch block.The statements which may raise an exceptions are written inside try block.The exceptions raised by try block are handled by catch block.In catch block the exception is handled by printing some message.

e.g.

try

{

//statements which cause exception

}

catch(exception exceptionname)

{

//statements printing msgs

}

e.g. If you try to divide a number by zero it would cause an exception this type of exception is Arithmetic Exception.(In JAVA all exceptions have got name).

Program to explain ArithmeticException

```
import java.io.*;
class exception
{
public static void main(String []args)
{
```

```
int a=2,b=0;
try
{
c=a/b;
}
catch(ArithmeticException e)
{
System.out.println("error"+e);
}
}
}
```

Program to explain ArrayIndexOutOfBounds Exception

```
import java.io.*;
class exception
{
public static void main(String []args)
{
int d[]={10,20,30};
try
{
System.out.println("val="+d[4]);
}
catch(ArrayIndexOutOfBoundsException e)
{
System.out.println("error"+e);
}
}
}
```

The finally Block

If in a program we want to execute some set of statements if the exception raised or not then such statements are kept in finally block.This block gurantees the execution for its statements whether exception is raised or not and also successfully caught or not by the programmer.

Program to show use of finally block

```
import java.io.*;
class exception
{
```

```
public static void main(String []args)
{
int d[]={10,20,30};
try
{
System.out.println("val="+d[4]);
}
catch(ArrayIndexOutOfBoundsException e)
{
System.out.println("error"+e);
}
finally
{
System.out.println(" In finally block");
}
}
```

NOTE : - catch block cannot handle more than one exception.If we want to handle more than one exception we need to write than many catch block.

Program to explain multiple exceptions

```
import java.io.*;
class exception
{
public static void main(String []args)
{
int a=2,b=0,c,d[]={10,20,30};
try
{
System.out.println("val="+d[4]);
c=a/b;
}
catch(ArrayIndexOutOfBoundsException e)
{
System.out.println("error"+e);
}
catch(ArithmeticException e)
{
System.out.println("error"+e);
}
}
}
```

Following diagram explains Hierarchy of Exception

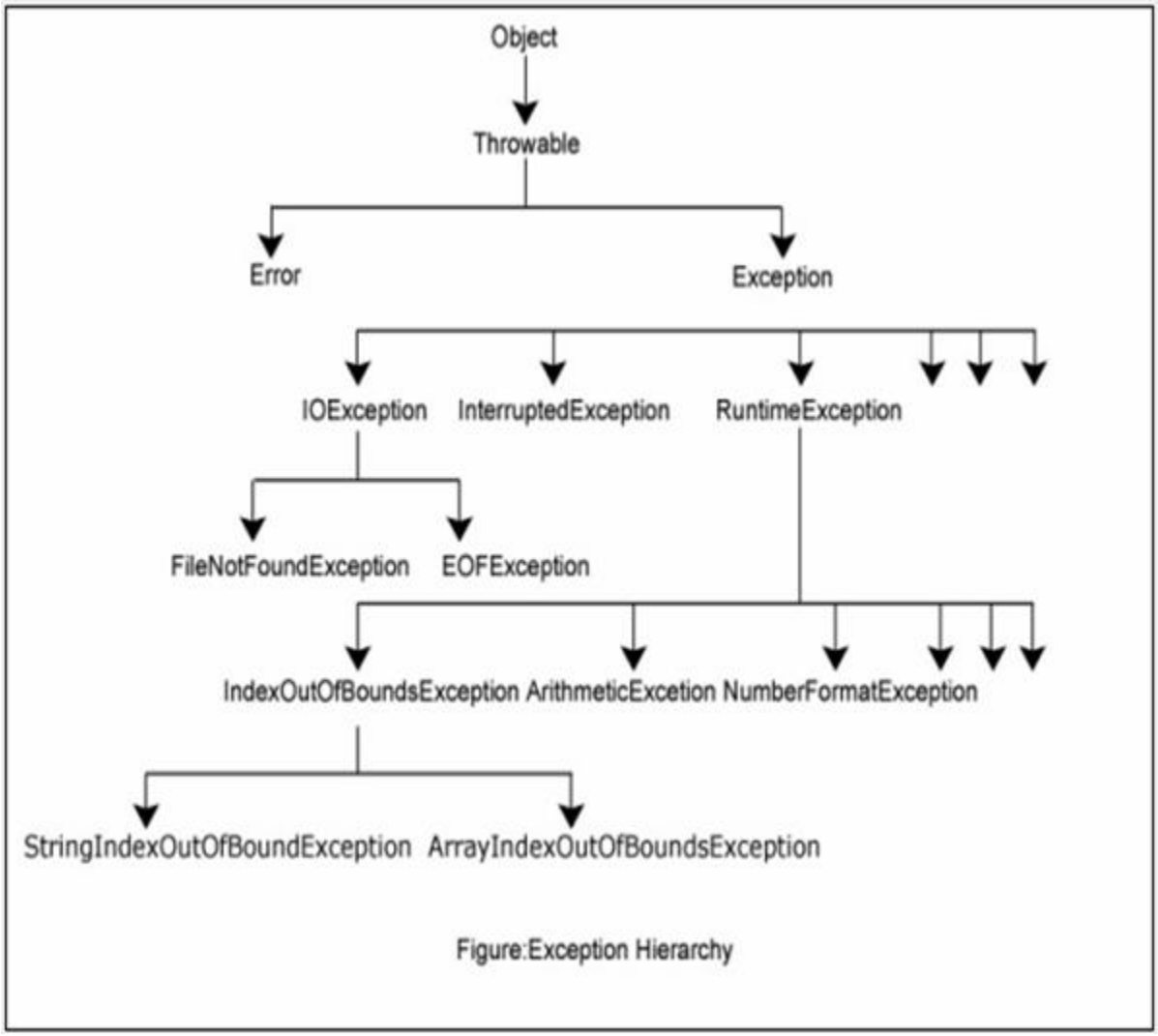

Figure:Exception Hierarchy

Difference between checked exceptions and unchecked exceptions

Checked exceptions are exceptions that are not subclasses of RuntimeException class and must be handled by the programmerexplicitly and if not the code does not compile whereas unchecked exceptions are the subclasses of RuntimeException which even when not handled by the programmer the program compiles .The difference between checked and unchecked exceptions is compilation.

Use of throws keyword

Throws keyword is used to tell the possible exception may occur when the function is executed

Syntax

return type functionname(parameter list) throws Exception

Program which explains use of throws keyword

```
import java.io.*;
class  Throwtest
{
public static void display()
{
try
{
FileInputStram fis = new FileInputStram(“student.txt”);
}
Catch(FileNotFoundException e)
{
System.out.println(“ Problem in opening file” +e);
}
public static void show() throws FileNotFoundException
{
FileInputStream fis= new FileInputStream(“employee.txt”);
System.out.println(“File successful opened”);
}
Public static void main(String args[]) throws FileNotFoundException
{
display();
show();
}
}
```

Use of throw keyword

We can pass information to the user as exception message.Suppose in calculation of marks if a student is having marks more than hundered we can print the statement of exceeding marks on screen.This message can be passed as an exception message using any one of the predefined exception classes.We can use exclusively a keyword throw

e.g.

```
public class Studentmarks

{

public static void main(String args[])
```

```
{
int m1=105;
if(m1>100)
throw  new Exception(“Marks can not be more than 100”);
else
System.out.println(“Valid Marks”);
}
}
```

Throw is a keyword in JAVA used with exceptions with this keyword we can explicitly throw any exception.

User defined Exception

User can design exception we can throw it at appropriate situation.

e.g.

```
import java.io.*;
class myexception extends Exception
{
myexception(String s)
{
super(s);
}
}
class user
{
public static void main(String argv[])throws myexception
{
int marks=180;
if( marks > 100 )
{
throw new myexception ("marks are more than 100 ");
}
else
System.out.println("marks "+marks);
```

```
}
}
}
}
```

In the above example a userdefined exception called myexception which extended from Exception class.Super is a function using which a constructor of a base class can be called.Whenever marks are exceeding 100 exception is thrown with the keyword throw along with the message.