Exception Handling

- The exception handling in java is one of the powerful mechanism to handle the runtime errors so that normal flow of the application can be maintained.

What is exception ?

- **Dictionary Meaning:** Exception is an abnormal condition.
- In java, exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

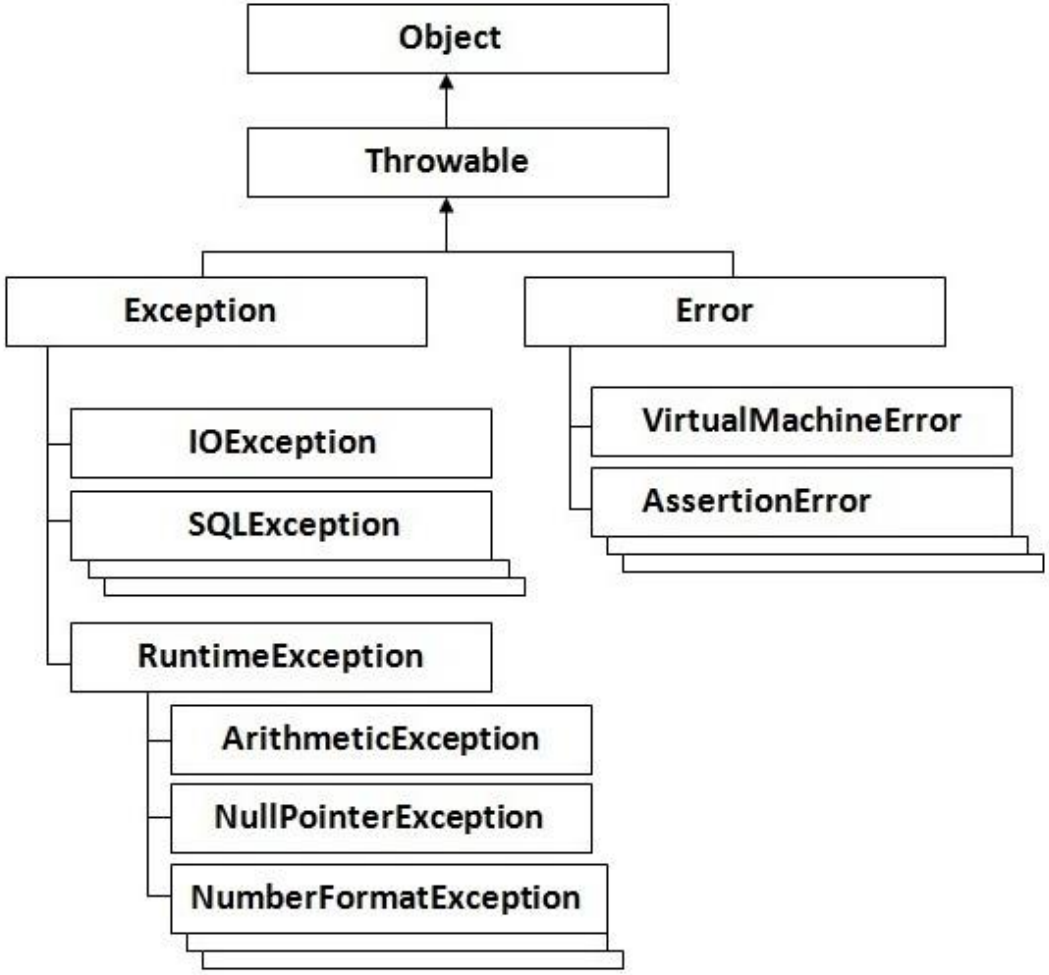
Exception Handling

Advantage of Exception Handling

The core advantage of exception handling is to maintain the normal flow of the application. Exception normally disrupts the normal flow of the application that is why we use exception handling.

```
statement 1;
statement 2;
statement 3;  //exception occurs
statement 4;
statement 5;
```

Exception Handling



Exception Handling

Types of Exception

There are mainly two types of exceptions: checked and unchecked where error is considered as unchecked exception. The sun microsystem says there are three types of exceptions:

1. Checked Exception
2. Unchecked Exception
3. Error

Exception Handling

1. **Checked Exception :**The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions e.g.IOException, SQLException etc. Checked exceptions are checked at compile-time.

2. **Unchecked Exception:**The classes that extend RuntimeException are known as unchecked exceptions e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc. Unchecked exceptions are not checked at compile-time rather they are checked at runtime.
3. **Error:** Error is irrecoverable e.g. OutOfMemoryError, VirtualMachineError, AssertionError etc.

Exception Handling

Common scenarios where exceptions may occur

1. Scenario where ArithmeticException occurs

If we divide any number by zero, there occurs an ArithmeticException.

```
int a=50/0;//ArithmeticException
```

2. Scenario where NullPointerException occurs

```
String s=null;
System.out.println(s.length());//NullPointerException
```

Exception Handling

3. Scenario where NumberFormatException occurs

The wrong formatting of any value, may occur NumberFormatException. Suppose I have a string variable that have characters, converting this variable into digit will occur NumberFormatException.

```
String s="abc";
int i=Integer.parseInt(s);//NumberFormatException
```

4. Scenario where ArrayIndexOutOfBoundsException occurs

```
int a[]=new int[5];
a[10]=50; //ArrayIndexOutOfBoundsException
```

Exception Handling

Java Exception Handling Keywords

There are 5 keywords used in java exception handling.

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

Exception Handling

try-catch : Java try block is used to enclose the code that might throw an exception. It must be used within the method.

Try

{

//code that may throw exception

}

catch(Exception_class_Name ref)

{

}

Exception Handling

catch block : Java catch block is used to handle the Exception. It must be used after the try block only.

Exception Handling

Multi catch block: If you have to perform different tasks at the occurrence of different Exceptions, use java multi catch block.

```
public class TestMultipleCatchBlock{
public static void main(String args[]){
 try{
  int a[]=new int[5];
  a[5]=30/0;
 }
 catch(ArithmeticException e){System.out.println("task1 is completed");}
 catch(ArrayIndexOutOfBoundsException e){System.out.println("task 2 completed");}
 catch(Exception e){System.out.println("common task completed");}
 System.out.println("rest of the code...");
 }
}
```

Exception Handling

Multi catch block: If you have to perform different tasks at the occurrence of different Exceptions, use java multi catch block.

```
public class TestMultipleCatchBlock{
public static void main(String args[]){
 try{
  int a[]=new int[5];
  a[5]=30/0;
 }
 catch(ArithmeticException e){System.out.println("task1 is completed");}
 catch(ArrayIndexOutOfBoundsException e){System.out.println("task 2 completed");}
 catch(Exception e){System.out.println("common task completed");}
 System.out.println("rest of the code...");
 }
}
```

Exception Handling

Nested try block : The try block within a try block is known as nested try block in java.

Why use nested try block ?

Sometimes a situation may arise where a part of a block may cause one error and the entire block itself may cause another error. In such cases, exception handlers have to be nested.

Exception Handling

```
class Excep6{
 public static void main(String args[]){
  try{
    try{
     System.out.println("going to divide");
     int b =39/0;
    }catch(ArithmeticException e){System.out.println(e);}
    try{
    int a[]=new int[5];
    a[5]=4;
    }
}
```

Exception Handling

```
catch(ArrayIndexOutOfBoundsException e)
{
System.out.println(e);
}

    System.out.println("other statement);
  }catch(Exception e){System.out.println("handeled");}

  System.out.println("normal flow..");
 }
```

Exception Handling

throw keyword : The Java throw keyword is used to explicitly throw an exception. We can throw either checked or uncheked exception in java by throw keyword. The throw keyword is mainly used to throw custom exception. We will see custom exceptions later.

The syntax of java throw keyword is given below.

throw exception;

throw new IOException("sorry device error);

Exception Handling

```
public class TestThrow1{
   static void validate(int age){
     if(age<18)
      throw new ArithmeticException("not valid");
     else
      System.out.println("welcome to vote");
   }
   public static void main(String args[]){
      validate(13);
      System.out.println("rest of the code...");
  }
}
```

Exception Handling

throws

- throws is a keyword in java language which is used to throw the exception which is raised in the called method to it's calling method throws keyword always followed by method signature.

return_type method_name() **throws** exception_class_name{

//method code

}

Exception Handling

```
public class DivZero{
public void division(int a, int b)throws ArithmeticException{
if(b==0){
ArithmeticException ae=new ArithmeticException("Does not enter zero for Denominator");
throw ae;
}
else{
int c=a/b;
System.out.println("Result: "+c);
}
}
public static void main(String[] args){
DivZero d=new DivZero();
d.division(10,0);
}
}
```

Exception Handling

Java finally block

- **Java finally block** is a block that is used *to execute important code* such as closing connection, stream etc.
- Java finally block is always executed whether exception is handled or not.
- Java finally block must be followed by try or catch block.

Exception Handling

```
class TestFinallyBlock1{
  public static void main(String args[]){
  try{
   int data=25/0;
   System.out.println(data);
  }
  catch(NullPointerException e){System.out.println(e);}
  finally{System.out.println("finally block is always executed");}
  System.out.println("rest of the code...");
  }
}
```

Difference between throw and throws in Java

No.	throw	throws
1)	Java throw keyword is used to explicitly throw an exception.	Java throws keyword is used to declare an exception.
2)	Checked exception cannot be propagated using throw only.	Checked exception can be propagated with throws.
3)	Throw is followed by an instance.	Throws is followed by class.
4)	Throw is used within the method.	Throws is used with the method signature.
5)	You cannot throw multiple exceptions.	You can declare multiple exceptions e.g. public void method()throws IOException,SQLException.