Exception

Prof. Debashis Hati
Email :dhatifcs@kiit.ac.in
Cell No: 9437028209/ 6370961683

What is an Exception?

Exception

 It is an abnormal condition that has occurred in a code s

 it is a run time error.

Problem associated with an exception If it is not handled properly, the program halts abruptly. You have not got the desired solution.

Java Exception Handling mechanism

When an abnormal condition arises , an exceptional object is created and it is thrown from that method or code sequence.

Java provides 5 keywords to handle the exception

1 and 2 try …….. catch

3 finally

4 throw

5throws

try….catch

The program statements that has monitored for exception, are put into the try catch block.

Syntax :

```
try {
        1……………
        2…………….
        3…………….
        4……………..
} catch( Exception Type e){
        …………..
        ……………
}
```

Line 2, Exception is occurred and the exceptional object is thrown.

Line 3 and 4 are not executed.

The exceptional object is checked with the catch block. If it is matched, the code of the catch block executed (Exception handler) otherwise JVM handles the exception.

try….multiple catch

Sometime, the code segments may generate more than one exception, in that case more than one catch block is used.

```
try {
        1……………
        2…………….
        3…………….
        4……………..
} catch( Exception Type1 e){
        ………….
        ……………
} catch( Exception Type2 e){
        ………….
        ……………
}
```

.

Try….catch…finally

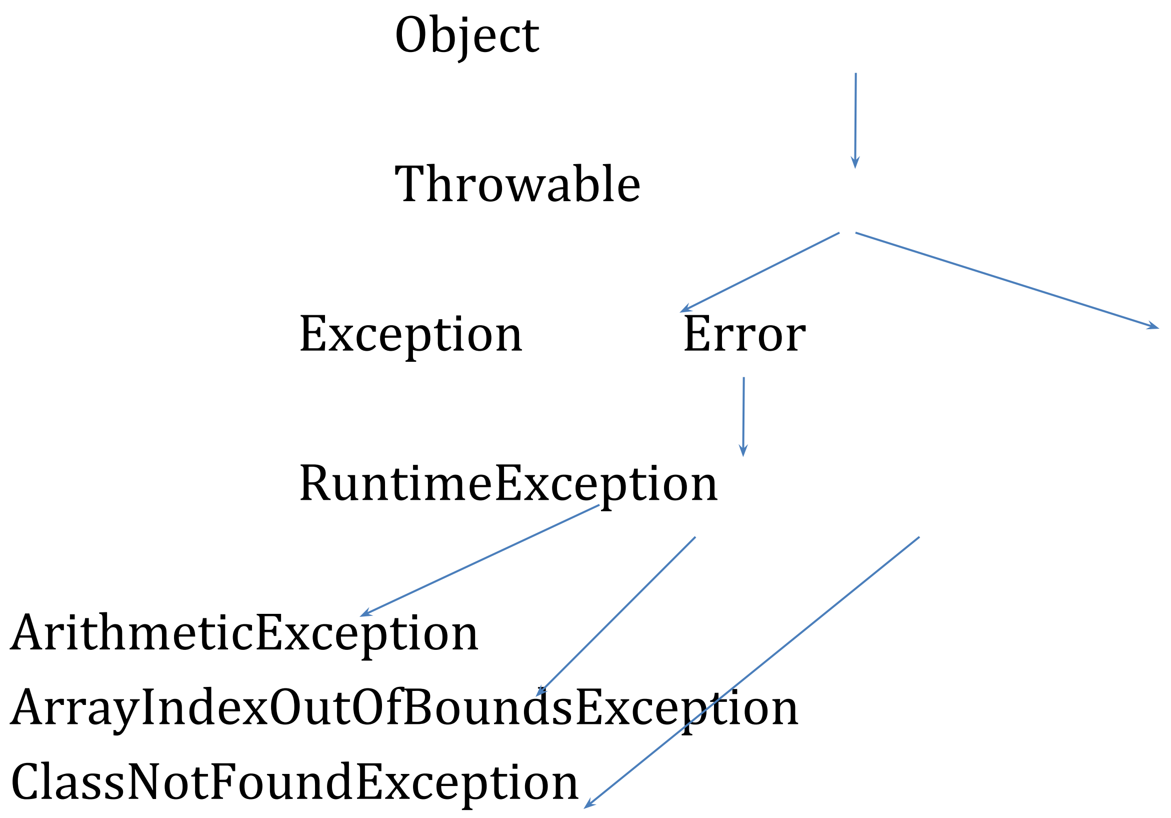
Sometime, there is a need of executing a task whether an exception is thrown or not. In that situation, finally block is used.

```
try {
        1……………
        2…………….
        3…………….
        4……………..
} catch( Exception Type1 e){
        …………..
        ……………
} catch( Exception Type2 e){
        …………..
        ……………
} finally{
        …………..
        ……………
```

finally block is a optional. A try block may have multiple catches , but it has a single finally block.

.

Exception Types



A user's program generally handles the Exception class and it's sub classes.

Error type exception is generated by the java runtime system which is not handled by the user's program.

Uncaught Exception

Intentionally, write a code to genearte an exception and let’s see what is happened???

```
class Demo{
  public static void main(String x[]){
        int a= 0;
        int b= 4/a;
        System.out.println(“Not executed”);
  }
}
```

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

Handle the Exception using try catch

```
class Demo{
  public static void main(String x[]){

        int a= 0;
        int b= 4/a;
        System.out.println(“Not executed”);
  }
}
```

Handle the Exception using try catch

```
class Demo{
 public static void main(String x[]){
        try{
                int a= 0;
                int b= 4/a;
                System.out.println(“Not executed”);
        }catch ( ArithimeticException e){


        }


 }
}
```

Handle the Exception using try catch

```
class Demo{
 public static void main(String x[]){
        try{
                int a= 0;
                int b= 4/a;
                System.out.println(“Not executed”);
        }catch ( ArithimeticException e){
                  System.out.println(“ Division by zero error”);
        }


 }
}
```

Handle the Exception using try catch

```
class Demo{
  public static void main(String x[]){
        try{
                int a= 0;
                int b= 4/a;
                System.out.println("Not executed");
        }catch ( ArithimeticException e){
                 System.out.println(" Division by zero error");
        }
                 System.out.println(" out side catch block");
  }
}
```

Handle the Exception using try catch

```
class Demo{
 public static void main(String x[]){
      try{
            int a= 0;
            int b= 4/a;
            System.out.println(“Not executed”);
      }catch ( ArithimeticException e){
             System.out.println(“ Division by zero error”);
      }
             System.out.println(“ out side catch block”);
 }
}
```

```
Division by zero error
out side catch block
```

Multiple catch blocks

```
class Demo{
  public static void main(String x[]){
          try{
                  int a= x.length;
                  int b= 4/a;
                  int c[] ={40};
                  c[5]= 50;
                  System.out.println(“Not executed”);
          }catch ( ArithimeticException e){
             System.out.println(“ Division by zero error”);

          } catch ( ArrayIndexOfBoundsException e){
             System.out.println(“ Array size error”);
          }

             System.out.println(“ out side catch block”);
  }
}
java Demo

java Demo  Hello
```

Important :Multiple catch blocks

```
class Demo{
  public static void main(String x[]){
          try{
                  int a= x.length;
                  int b= 4/a;
                  System.out.println(“Not executed”);
          }catch ( Exception e){
              System.out.println(“ it is an exception”);

          } catch (ArithimeticException e){
              System.out.println(“ Division by zero error”);
          }

              System.out.println(“ out side catch block”);
  }
}
```

Important :Multiple catch blocks

```
class Demo{
  public static void main(String x[]){
          try{
                 int a= x.length;
                 int b= 4/a;
                 System.out.println(“Not executed”);
          }catch ( Exception e){
             System.out.println(“ it is an exception”);

          } catch (ArithimeticException e){
             System.out.println(“ Division by zero error”);
          }

             System.out.println(“ out side catch block”);
  }
}
```

compilation error : Unreachable code generates compilation error.

Important :Multiple catch blocks

```
class Demo{
  public static void main(String x[]){
        try{
              int a= x.length;
              int b= 4/a;
              System.out.println(“Not executed”);
        }catch ( Exception e){
           System.out.println(“ it is an exception”);

        } catch (ArithimeticException e){
           System.out.println(“ Division by zero error”);
        }

           System.out.println(“ out side catch block”);
  }
}
```

Exception subclass must use first than the super class.

Important :Multiple catch blocks

```
class Demo{
 public static void main(String x[]){
        try{
               int a= x.length;
               int b= 4/a;
               System.out.println(“Not executed”);
        }catch (ArithimeticException e){
           System.out.println(“Division by zero”);

        } catch (Exception e){
           System.out.println(“ it is an exception ”);
        }

           System.out.println(“ out side catch block”);
 }
}
```

Division by zero
out side catch block

Nested try   catch

```
try {

          try {

          }catch(  ) {

          }catch ( ) {

          }

}catch ( ) {

}
```

throw statement

Previously, the exception is created by the java runtime system and it is automatically thrown.

It is also possible to throw the exception explicitly or programmatically.

Syntax :

throw throwable instance;

 object of throwable or it’s subclasses.

any non thr0wable object can’t be used.

Effect :

1. **The flow of execution stops immediately after the throw statement.**
2. **Nearest catch block is inspected.**
3. **If there is match, the exception will be caught and will be handled.**
4. **If there is no match, java runtime system will handle the exception**

Example :throw statement

```
class Demo {
  static void fi(){
          try {
          throw new NullPointerException(“demo exception”);
          System.out.println(“Not Executed”);
          }catch (NullPointerException e){
                   System.out.println(“caught in the method”);
                  throw e;
          }
  }
  public static void main(String x[]){
          try {
                  f1();
          }catch (NullPointerException e){
                   System.out.println(“Recaught in main”);
          }     }     }
```

throws

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

Syntax : return type method_name(param list) throws exceptuion1, exception2 ,… {

```
        }
class  Demo{
        static void f1(){
              System.out.println(“Inside method”);
              throw new IllegalAccessException(“Excep”);
              }
        public static void main(String x[]){
              f1();
        }
 }
```

Comparison of two different types exception

```
class  Demo{
        static void f1(){
               System.out.println(“Inside method”);
               throw new NullPointerException(“Excep1”);
               }
        public static void main(String x[]){
               f1();
        }            }
class  Demo{
        static void f1(){
               System.out.println(“Inside method”);
               throw new IllegalAccessException(“Excep2”);
               }
        public static void main(String x[]){
               f1();
        }            }
```

Java Built-in Exception classes

 available / exists in java.lang package

** it is 2 types**

1. **Unchecked Exception The compiler does not check to see if the method handles it or not or includes in throws list.**

2. **Checked Exception must be included in the throws list otherwise compilation error occurs.**

User Defined Exception

You can define your own exception and your exception must follow

- **Your exception class must be subclass of Exception class**
- **It can override the methods of it’s super class like Throwable**

Example

Implement a Triangle class having three data members a,b,c as it's sides. This class includes appropriate constructors and two methods

find_area() : is used to return the area of the triangle.

find_perimeter() : is used to return the perimeter of the triangle.

Implement an user defined exception class NoTriangleFormException which is thrown when object is created with legal values(a+b>c and b+c>a and c+a>b). Write down the necessary java code to test the functionality of Triangle class.

Implementation

Implement a Triangle class having three data members a,b,c as it's sides. This class includes appropriate constructors and two methods

find_area() : is used to return the area of the triangle.

find_perimeter() : is used to return the perimeter of the triangle.

Implement an user defined exception class **NoTriangleFormException** which is thrown when object is created with illegal values(a+b>c and b+c>a and c+a>b). Write down the necessary java code to test the functionality of Triangle class.

Implementation

```
class NoTriangleFormException extends Exception
 {
        NoTriangleFormException(){
              super();
       }
       String toString(){
              return " No Triangle formed"
       }
 }
```

Implementation

```
   class Triangle {
    int a,b,c;
    Triangle(int p,int q,int r) throws NoTriangleFormException{
          if (((p+q)>r )&&((q+r)>p)&&((r+p)>q))
                a=p; b=q;c=r:
          else
                throw NoTriangleFormException();
    }
    void find_Area() {}
    void find_Perimeter() {}
}
class Demo{
}
```

Implementation

```
class  Demo_Triangle {
        public static void main(String x[]){
                try {
                Triangle t1= new Triangle(3,4,5);
                }catch(NoTriangleFormException e){
                        System.out.println)e);
                }
                t1.find_Area();
                t1.find_Perimeter();


 }
}
```

Assignments, 29/3/2022

1. **check uncaught ArithmeticException**
2. **Handle with try and catch**
3. **Check with multiple try**
4. **Use of toString() method**
5. **Implement a class Money having data members Rupee and Paise. Include appropriate constructors and following methods**

   Money add(Money) is used to add two Money Object.

   Overload the toString() method to display Money object in Rs.10.50paise" format.

   Write down the necessary java code to test the functionality of Money class.

6. **Implement a vector class has a data member of integer array. Include appropriate constructor and vector add(vector) which is used to add two vector objects. Write down the necessary java code to test the functionality of vector class. Vector class has a toString() which is used to display the content of vector object.**

   Exmp: Vector display format [1,2,3,4,5,6]

Implementation-2

```
class NegativeAmountException extends Exception {
  NegativeAmountException(){
      super();}
 String toString(){}
 }

class MinimumBalanceException extends Exception{
  MinimumBalanceException(){
      super();}
 String toString(){}


}
```

Implementation-2

```
class Account {
  String name;
  int accno;
  float balance;
  Account(){}
   Account(String x, int y, float z) throws  NegativeAmountException,
MinimumBalanceException{
          if (z <0)
          throw NegativeAmountException();
          elseif ( z< 2000.00f)
                  throw MinimumBalanceException();
                  else
                  // initialize
  }
  void withdraw( float cash) throws NegativeAmountException,
MinimumBalanceException {
  }
  void  deposite( float cash) throws   NegativeAmountException {
  }      }
```

Implementation-2

```
class Account {
  void withdraw( float cash) throws NegativeAmountException,
MinimumBalanceException {
  if (cash <0.0f)
        throw NegativeAmountException();
        elseif (( balance- cash)< 2000.00f)
              throw MinimumBalanceException();
              else
                    balance=balance-cash;
  }

}
```

Implementation-2

```
class Account {
 void  deposite( float cash) throws    NegativeAmountException {
        if (cash <0)
        throw NegativeAmountException();
        else
              balance=balance+cash


 }

}
```

Implementation-2

```
   class Demo {
    public static void main(String x[]){
        // input three data members
        try {
            Account a1=new Account( “abc”,1, 3000)
        }catch(NegativeAmountException e){
            System.out.println(e);

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

finally block

- The code within finally block is executed after try- catch.
- The code must be executed whether exception is thrown or not.
- finally can not be used alone.
- A single finally is used with a try – catch.

There are 3 different scenarios where finally is used.

1. Exception is thrown.
2. There is no exception
3. Before the method is returned.

finally Example

```
class   finallyDemo{
      static void proA(){
            try {
                  System.out.println(“Inside proA”);
                  throw new  RuntimeException(“demo”);
            }finally {
                   System.out.println(“ proA’s  finally”);
            }
      }
```

finally Example

```
static void proB(){
      try {
            System.out.println(“Inside proB”);
      }finally {
             System.out.println(“ proB’s  finally”);
      }
}
```

finally Example

```
 static void proC(){
      try {
            System.out.println("Inside proC");
            return;
      }finally {
             System.out.println(" proB's  finally");
      }
 }
public static void main(String x[]){
      try {
            procA();
      }catch(Exception e){System.out.println(e);}
      proB();  proC():}}
```

Assignemnt, 12/4/2022

2.

Implement an Account class having three data members name, accno,balance. This class includes appropriate constructors and two methods

void withdraw(float cash) : is used for withdrawal.

void deposite() : is used for deposites.

Implement two user defined exception classs **NegativeAmountException and MinimumBalanceException.**

NegativeAmountException and MinimumBalanceException are thrown while creating the object.

Both exceptions are associated with withdrawal and only NegativeAmountException is associated with deposites. Write down the necessary java code to test the functionality of Account class.

Assignemnt, 12/4/2022

3.

Implement a Stack class .This class includes appropriate data members s and constructors and two methods

void push(int): is used to push into stack.

int pop() : is used for pop.

Implement two user defined exception classs StackFullException and StackeEmptyException

StackFullException is **associated with push** and StackeEmptyException **is associated with pop.**

Write down the necessary java code to test the functionality of Stack class.