Java throws keyword

The **Java throws keyword** is used to declare an exception. It gives an information to the programmer that there may occur an exception so it is better for the programmer to provide the exception handling code so that normal flow can be maintained.

Exception Handling is mainly used to handle the checked exceptions. If there occurs any unchecked exception such as NullPointerException, it is programmers fault that he is not performing check up before the code being used.

Syntax of java throws

```
return_type method_name() throws exception_class_name{
//method code
}
```

Which exception should be declared

Ans) checked exception only, because:

- **unchecked Exception:** under your control so correct your code.
- **error:** beyond your control e.g. you are unable to do anything if there occurs VirtualMachineError or StackOverflowError.

Advantage of Java throws keyword

Now Checked Exception can be propagated (forwarded in call stack).

It provides information to the caller of the method about the exception.

Java throws example

Let's see the example of java throws clause which describes that checked exceptions can be propagated by throws keyword.

```
import java.io.IOException;
class Testthrows1{
  void m()throws IOException{
    throw new IOException("device error");//checked exception
  }
  void n()throws IOException{
    m();
  }
  void p(){
   try{
    n();
   }catch(Exception e){System.out.println("exception handled");}
  }
  public static void main(String args[]){
   Testthrows1 obj=new Testthrows1();
   obj.p();
   System.out.println("normal flow...");
  }
}
```

Test it Now

Output:

```
exception handled
normal flow...
```

Rule: If you are calling a method that declares an exception, you must either caught or declare the exception.

There are two cases:

1. **Case1:**You caught the exception i.e. handle the exception using try/catch.
2. **Case2:**You declare the exception i.e. specifying throws with the method.

Case1: You handle the exception

- In case you handle the exception, the code will be executed fine whether exception occurs during the program or not.

```
import java.io.*;
class M{
 void method()throws IOException{
  throw new IOException("device error");
 }
}
public class Testthrows2{
   public static void main(String args[]){
    try{
     M m=new M();
     m.method();
    }catch(Exception e){System.out.println("exception handled");}

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

Test it Now

```
Output:exception handled
       normal flow...
```

Case2: You declare the exception

- A)In case you declare the exception, if exception does not occur, the code will be executed fine.
- B)In case you declare the exception if exception occures, an exception will be thrown at runtime because throws does not handle the exception.

A)Program if exception does not occur

```
import java.io.*;
class M{
 void method()throws IOException{
  System.out.println("device operation performed");
 }
}
class Testthrows3{
   public static void main(String args[])throws IOException{//declare exception
     M m=new M();
     m.method();

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

Test it Now

```
Output:device operation performed
       normal flow...
```

B)Program if exception occurs

```
import java.io.*;
class M{
 void method()throws IOException{
  throw new IOException("device error");
 }
```

```
}
class Testthrows4{
   public static void main(String args[])throws IOException{//declare exception
     M m=new M();
     m.method();

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

Test it Now

```
Output:Runtime Exception
```

Difference between throw and throws

Click me for details

Que) Can we rethrow an exception?

Yes, by throwing same exception in catch block.

← prev next →