Basic Java : Exception Handling

Persistent University

Objectives :

- At the end of this module, you will be able to understand :
  - Traditional Error Handling Techniques
  - Importance of Exception Handling
  - Exception Handling Framework

Tradition error handling techniques

- Use of Boolean functions (which return TRUE/FALSE), integer functions (returns –1 on error) and other return arguments and special values [1] .

- Use of if –else – if construct.

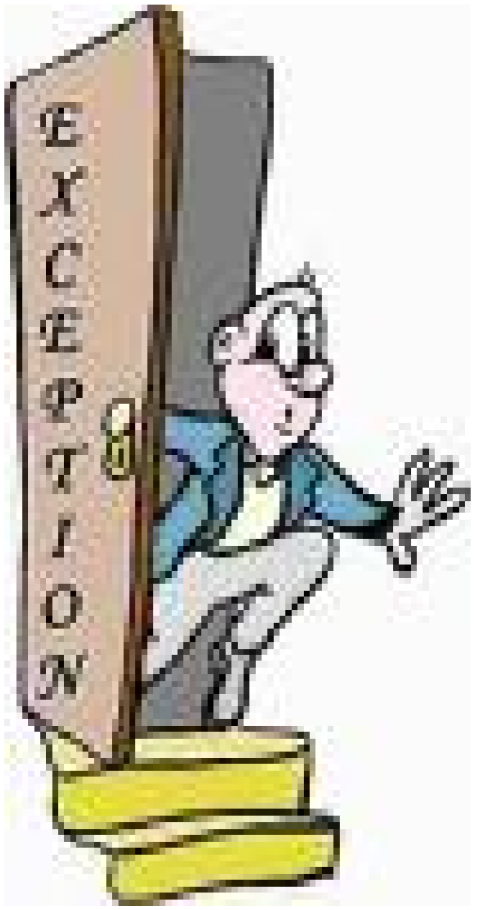
Drawbacks of traditional error handling techniques

- Reduced readability

- Error handling code and business logic is not separated

Exception handling

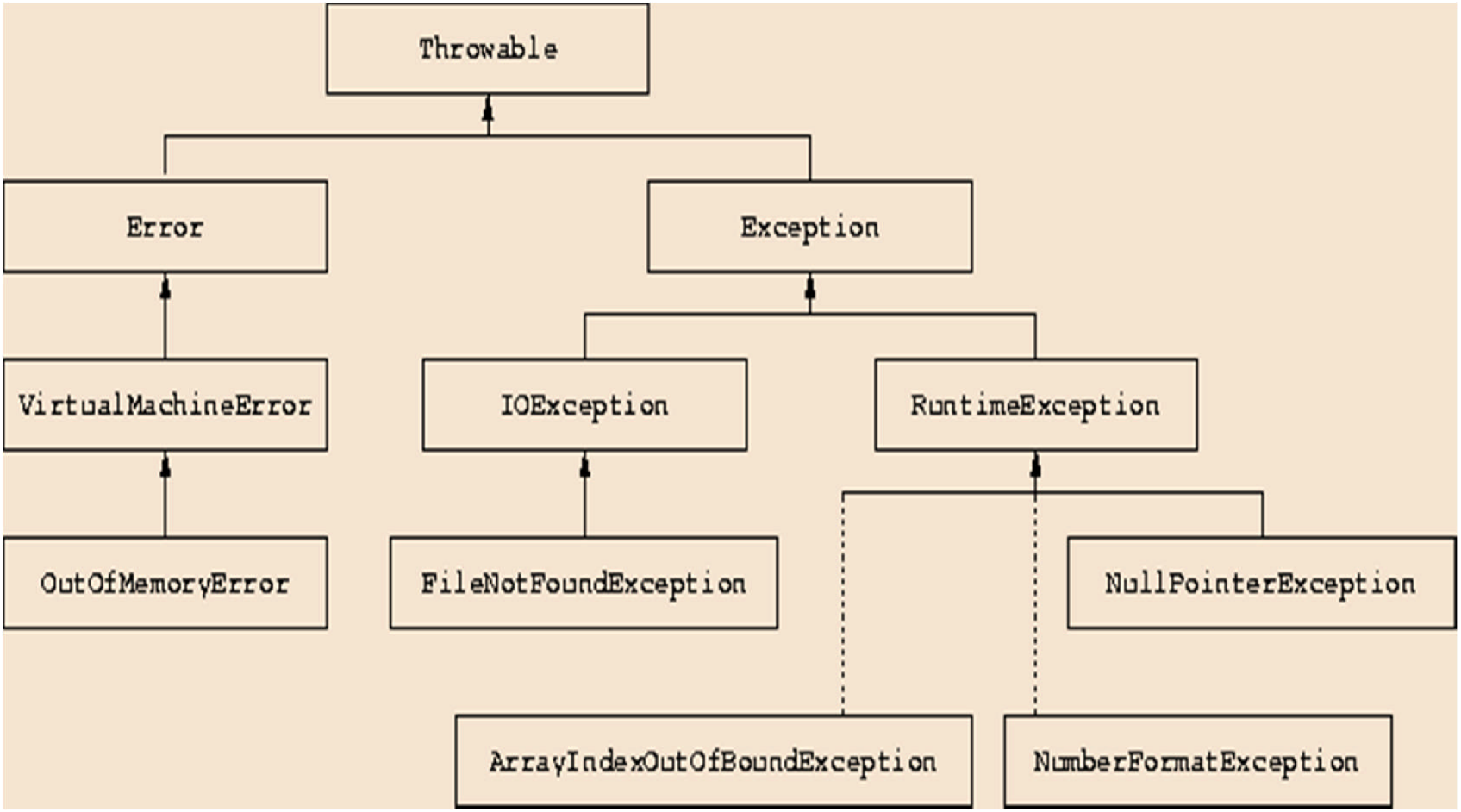
- What is an exception ?
- When does an exception occur ?
- How are exceptions managed
- What happens if we don't



Advantages of Exception Handling

- Separating Error-Handling Code from business logic
- Error propagation on the call stack
- Only required part of code will be executed

Hierarchy of exception classes

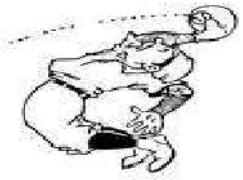


Exception handling in Java

```
try {
    /* statements to be monitored for exceptions */
}

catch(<exception type to be caught>) {
    /* actions to be taken when an exception is caught */
}

finally {
    /* operations to be performed irrespective of whether
    an exception occurs or not */
}
```



Exception handling in Java

- Try with one catch

```
int a,b,c;
a = 10;
b = 0;
try{
      c = a/b;
      System.out.println("c is: " + c);
}
catch(ArithmeticException e){
      System.out.println("Operation is not supported");
}
```

Exception handling in Java

- Try with multiple catch blocks

```
int a,b,c ;    a = 20;   b = 5;
int arr[] = new int[5];
try{
    c = a/b;
    System.out.println("c is: " + c);
    arr[9] = 10;
    System.out.println("arr[9] is: " + arr[9]);
}
catch(ArithmeticException e){
    System.out.println("Value of b can not be 0");  }
catch(ArrayIndexOutOfBoundsException e){
   System.out.println("Wrong array index");
}
```

ArithmeticException will be thrown if user enters value of b as 0

ArrayIndexOutOfBoundsException will be thrown

The finally clause

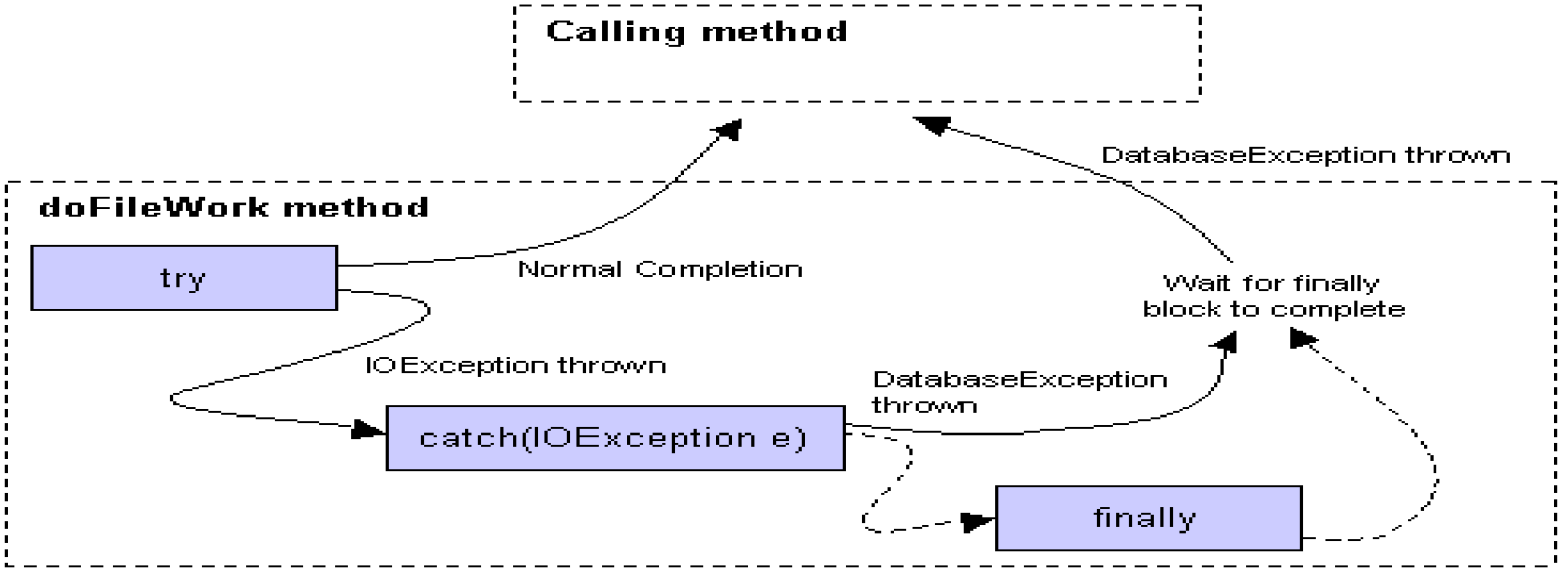
```
try {
    statements;
} catch(Exception ex) {
    handle ex;
} finally {
    finallyStatements;
}
```

- Occasionally you may want some code to be executed irrespective of whether an exception occurs or not.
- 3 possibilities
  - No exception arises in the try block
  - Exception occurs in the try block that is caught in a catch clause.
  - Exception occurs in the try block that is not caught by any catch block.

The finally clause……

- If a finally block returns a value, then that return supersedes any previous return in the try-catch block.
- If an exception was thrown in the try or catch blocks that was not caught, then execution of a return in the finally block prevents the exception from being thrown to caller.
- If an exception was thrown in the try-catch block, an exception thrown in the finally block will supersede the prior exception.
- finally blocks are not guaranteed to run to completion.
- The finally block is a good place for releasing shared resources, defensive code, and audit trail logging.

The finally clause….an example



- Finally will never execute if
  - you call System.exit(0) in the catch block, or
  - code in the catch block executes indefinitely.

Summary : Session

With this we have come to an end of our session, where we discussed about ….

- Traditional error handling mechanism
- Need of Exception handling
- Hierarchy of exceptions
- Exception handling framework

At the end of this session, we see that you are now able to answer following questions:

- What is exception handling framework?
- How to implement exception handling in applications?

PERSISTENT

Reference Material : Websites & Blogs

- http://www.tutorialspoint.com/java/java_exceptions.htm
- http://www.javatpoint.com/exception-handling-in-java
- http://tutorials.jenkov.com/java-exception-handling/index.html

Reference Material : Books

- ***Head First Java***
  - *By: Kathy Sierra, Bert Bates*
  - *Publisher: O'Reilly Media, Inc.*

- ***Java Complete Reference***
  - *By Herbert Schildt*

Key Contacts :

Java Interactive :

- Asif Immanad

asif_immanad@persistent.co.in

- Nisha Waikar

nisha_waikar@persistent.co.in

- Varsha Mulik

varsha_mulik@persistent.co.in

Persistent University :

- Poorva Kulkarni

poorva_kulkarni@persistent.co.in

Vice President :

- Shubhangi Kelkar

shubhangi_kelkar@persistent.co.in

Thank You !!!

Persistent University