Java & JEE Training

Day 13 – Exception Handling (Contd.)

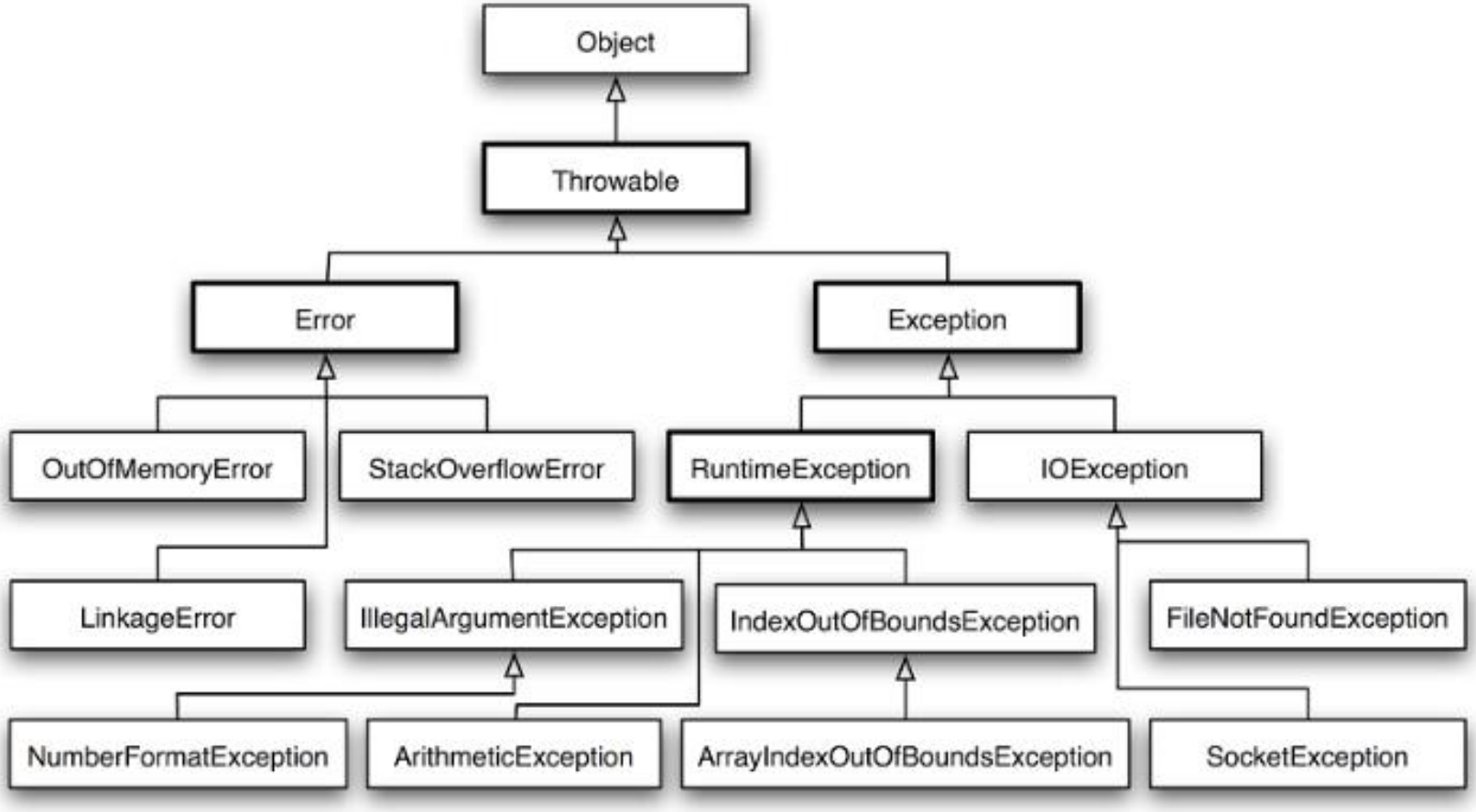
MindsMapped Consulting

Review of Exception handling concepts from last session…

- What is an exception? What is an error?
- Throwable class – parent of Exception and Error
- Types of Exception: Checked and Unchecked Exceptions

Checked vs Unchecked Exceptions – Hierarchy review

- **Checked Exceptions = Subclass of Exception**
- **Unchecked Exceptions = Subclass of RuntimeException**



Agenda

- throw and throws keywords
- Exception propagation – the cases of Checked and Unchecked Exceptions
- Defining your own custom Exception

Throwing exception… “throw” keyword

```
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 propagation example.. Unchecked exceptions are propagated through the method call stack

```
class TestExceptionPropagation1{
  void m(){
    int data=50/0;
  }
  void n(){
    m();
  }
  void p(){
   try{
    n();
   }catch(Exception e){System.out.println("exception handled");}
  }
  public static void main(String args[]){
   TestExceptionPropagation1 obj=new TestExceptionPropagation1();
   obj.p();
   System.out.println("normal flow...");
  }
}
```

Program which describes that checked exceptions are not propagated

```
class TestExceptionPropagation2{
  void m(){
    throw new java.io.IOException("device error");//checked exception
  }
  void n(){
    m();
  }
  void p(){
   try{
    n();
   }catch(Exception e){System.out.println("exception handeled");}
  }
  public static void main(String args[]){
   TestExceptionPropagation2 obj=new TestExceptionPropagation2();
   obj.p();
   System.out.println("normal flow");
  }
}
```

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.

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

Only checked exceptions should be declared, 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.

Java throws example

```
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...");
  }
}
```

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.

Difference between throw and throws in Java

```
void m(){
   throw new ArithmeticException("sorry");
}

void m()throws ArithmeticException{
  //method code
}

void m()throws ArithmeticException{
  throw new ArithmeticException("sorry");
}
```

Unchecked Exceptions? Why?

http://docs.oracle.com/javase/tutorial/essential/exceptions/runtime.html

Please read the article and let’s discuss this in next class.

Difference between final, finally and finalize

No.	final	finally	finalize
1)	Final is used to apply restrictions on class, method and variable. Final class can't be inherited, final method can't be overridden and final variable value can't be changed.	Finally is used to place important code, it will be executed whether exception is handled or not.	Finalize is used to perform clean up processing just before object is garbage collected.
2)	Final is a keyword.	Finally is a block.	Finalize is a method.

What is the output?

```
public class Test
{
   public static void aMethod() throws Exception
   {
      try /* Line 5 */
      {
         throw new Exception(); /* Line 7 */
      }
      finally /* Line 9 */
      {
         System.out.print("finally "); /* Line 11 */
      }
   }
   public static void main(String args[])
   {
      try
      {
         aMethod();
      }
      catch (Exception e) /* Line 20 */
      {
         System.out.print("exception ");
      }
      System.out.print("finished"); /* Line 24 */
   }
}
```

What is the output?

```
public class X
{
    public static void main(String [] args)
    {
        try
        {
            badMethod();
            System.out.print("A");
        }
        catch (Exception ex)
        {
            System.out.print("B");
        }
        finally
        {
            System.out.print("C");
        }
        System.out.print("D");
    }
    public static void badMethod() {}
}
```

What is the output?

```
public class X
{
   public static void main(String [] args)
   {
      try
      {
         badMethod(); /* Line 7 */
         System.out.print("A");
      }
      catch (Exception ex) /* Line 10 */
      {
         System.out.print("B"); /* Line 12 */
      }
      finally /* Line 14 */
      {
         System.out.print("C"); /* Line 16 */
      }
      System.out.print("D"); /* Line 18 */
   }
   public static void badMethod()
   {
      throw new RuntimeException();
   }
}
```

What is the output?

```
public class MyProgram
{
   public static void main(String args[])
   {
      try
      {
         System.out.print("Hello world ");
      }
      finally
      {
         System.out.println("Finally executing ");
      }
   }
}
```

What is the output?

```
class Exc0 extends Exception { }
class Exc1 extends Exc0 { } /* Line 2 */
public class Test
{
   public static void main(String args[])
   {
      try
      {
         throw new Exc1(); /* Line 9 */
      }
      catch (Exc0 e0) /* Line 11 */
      {
         System.out.println("Ex0 caught");
      }
      catch (Exception e)
      {
         System.out.println("exception caught");
      }
   }
}
```