EXCEPTION HANDLING IN JAVA

COMPILE TIME ERROR

LEXICAL ERROR

When we allow disallowed character in our code

```
if(x>y)
{

    //body

}
```

```
iif(x>y)
{

    //body

}
```

```
int a;
```

```
intt a;
```

SYNTAX ERROR

When the code is out of order

1

```
for(i=0; i<=10; i++)
```

→

```
for(i=0; i++; i<=10)
```

2. else without if

3.
```
if  x > y
 {
  x=10;
```

SEMANTIC ERROR

When meaning of the code is not clear

1. Variable not declared

2. Variable already defined

3. Type mismatch

RUN TIME ERROR

An abnormal condition during program execution that cannot be handled by user or programmer

Memory stack segment overflow

Linkage error

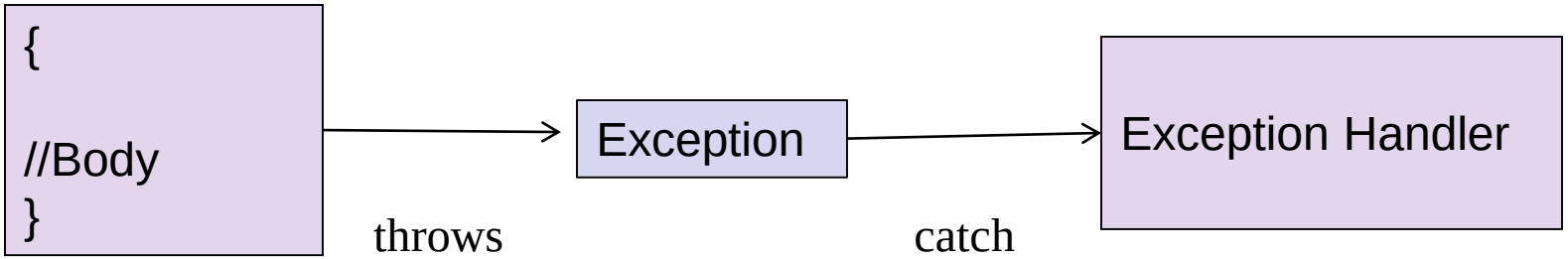
EXCEPTIONS

An abnormal condition during program execution that can be handled by programmer
Exceptions are thrown by JVM

1. Divide by zero
2. Attempt to access non existing file
3. Factorial of –ve numbers
4. Attempt to use invalid address

EXCEPTION HANDLING

A piece of code



1.Fix the abnormal condition

2.Prevent program from automatically terminating.

WHEN WE DON'T HANDLE EXCEPTIONS

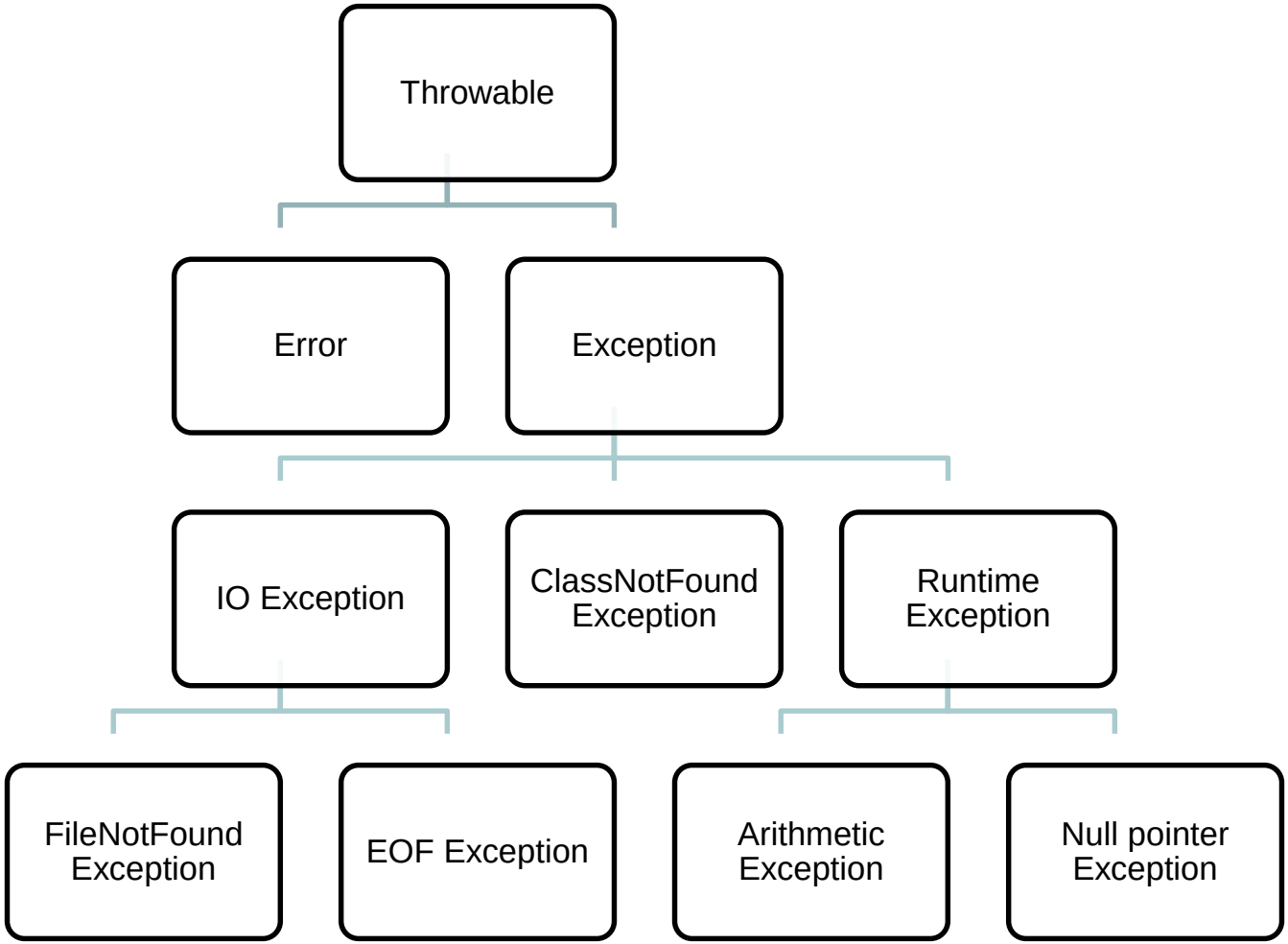
```
class ABC
{
  public static void main(String args[])
      {
        int d=0;
        int a=42/d;
       --------------------------------
      ----------------------------------
      }

}
```

Exception is caught by default handler

This causes execution of ABC to stop

EXCEPTION HIERARCHY



TRY and CATCH

SYNTAX

```
try
 {
      // block of code to monitor for errors
}
catch (Exception Type e)
{
     // exception handler for Exception Type
}
```

```
class XYZ
 {
        public static void main(String args[])
                {
                int d, a;
                try
                        {
                         d = 0;
                         a = 42 / d;
                         System.out.println (“Inside try block");
                         }
                catch (ArithmeticException  e)
                         {
                         System.out.println("Division by zero.");
                         }
                System.out.println("After catch statement.");
                }
}
```

Output

Division by zero
After catch statement.

Replace catch block with the following piece of code

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

Output

Exception: java.lang.ArithmeticException: / by zero
After catch statement.

TRY WITH MULTIPLE CATCH

Multiple catch blocks are used to catch different type of exceptions

```
Class XYZ
 {
public static void main(String args[])
 {
       try
        {
          int d=0;
          int a=42/d;
          int c[]={40}
          c[15]=100;
         }
      catch(ArithmeticException e)
        {
             System.out.println("Divide by zero);
        }

      catch(ArrayIndexOutOfBoundsException e)
        {
             System.out.println("Array index out of bounds);
       }
      System.out.println("After catch statement");
   }
 }
```

int d=10;

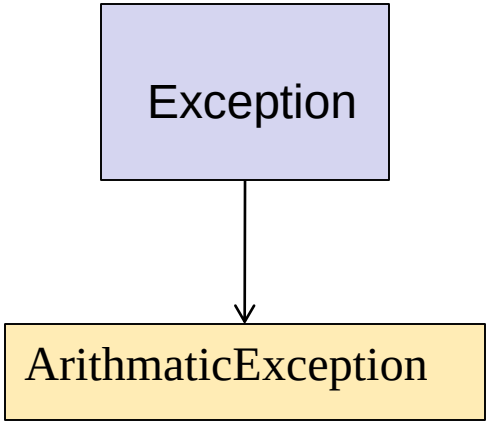
FOR MULTIPLE CATCH EXCEPTION SUBCLASS MUST APPEAR BEFORE EXCEPTION SUPERCLASS

```
public class XYZ{
   public static void main(String args[])
   {
       try
       {
          int d=0;
         int a=50/d;
       }
   catch(ArithmeticException e)
       {
          System.out.println("Aritmetic exception in my program");
       }

   catch(Exception e)
       {
          System.out.println(" Exception in my program");
       }

   }

}
```



NESTED TRY-CATCH

```
public class  XYZ {


   public static void main(String args[])
   {
      int a=0;

      try{
         int d=50/a;

            try
              {
               if(a==1)
                  a=a/(a-a);
               else if(a==2)
               {
                int c[]={15};
                c[40]=100;
               }
             }
         catch(ArrayIndexOutofBoundsException e)
            {
               System.out.println("Inner catch");
            }
        }  //END OF OUTER TRY
```

```
 catch(ArithmeticException e)
      {
         System.out.println("Outer catch");
      }

    }

}
```

int a=0;

int a=1;

int a=2;

THROW

Throw exception explicitly

```
public class  XYZ
{
   public static void main(String args[])
   {

      int a=1;
      try
         {
          throw new ArithmeticException("Testing");
          }

      catch(ArithmeticException e)
       {
        System.out.println(e);
      }

   }

}
```

java.lang.ArithmeticException: Testing

THROWS

If a method is capable of causing an exception that it does not handle, it must specify this behaviour so that callers of the method can guard themselves against that exception.

Syntax:

type method-name(parameter-list) throws *exception-list*

```
{
// body of method
}
```

```
public class  XYZ
{

    static void f1()  throws ArithmeticException
    {

        System.out.println("Inside f1");
        throw new ArithmeticException("Testing.......");
    }

    public static void main(String args[])
    {
      try{
           f1();
           }
     catch(ArithmeticException e)
          {
          System.out.println(e);
          }
    }
}
```

```
Inside f1
java.lang.ArithmeticException: Testing.......
```

FINALLY

Code Inside finally will be executed whether exception is caught or not

```
return_type method(parameter list)
{

CODE FOR FILE OPEN
try
{
OTHER LOGIC
}

CODE FOR FILE CLOSE

}
```

e

```
catch()
{



}
```

```
return_type method(parameter list)
{

CODE FOR FILE OPEN

try{
      OTHER LOGIC
    }

finally
   {
        CODE FOR FILE CLOSE
    }

}
```

e

```
catch()
{



}
```

```
public class XYZ {

        static void  f()
        {

         try {
              System.out.println("inside  method");
              throw new ArithmeticException(“Testing….");
             }

       finally
            {
             System.out.println(“inside finally");
            }
       }

        public static void main(String args[])
        {
        try {
                f();
             }

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

```
inside  method
inside finally
java.lang.ArithmeticException: Testing.....
```