Object Oriented Programming

Exception Handling

Course Instructor: Sadaf Anwar

Exceptions

1. Exception is an **abnormal condition** that arises when executing a program.
2. An exception is an object that describes an **exceptional condition (error)** that has occurred when executing a program.
3. For Example:
   1. Opening a non-existing file in your program
   2. Network connection problem
   3. Bad input data provided by user etc.

Exception Handling

1. **Handles** the run time error.
2. Maintain the **normal flow** of the application.
3. How?
4. Scenario:

- statement 1;
- statement 2;
- statement 3;
- statement 4;
- **statement 5;//exception occurs**
- statement 6;
- statement 7;
- statement 8;
- statement 9;

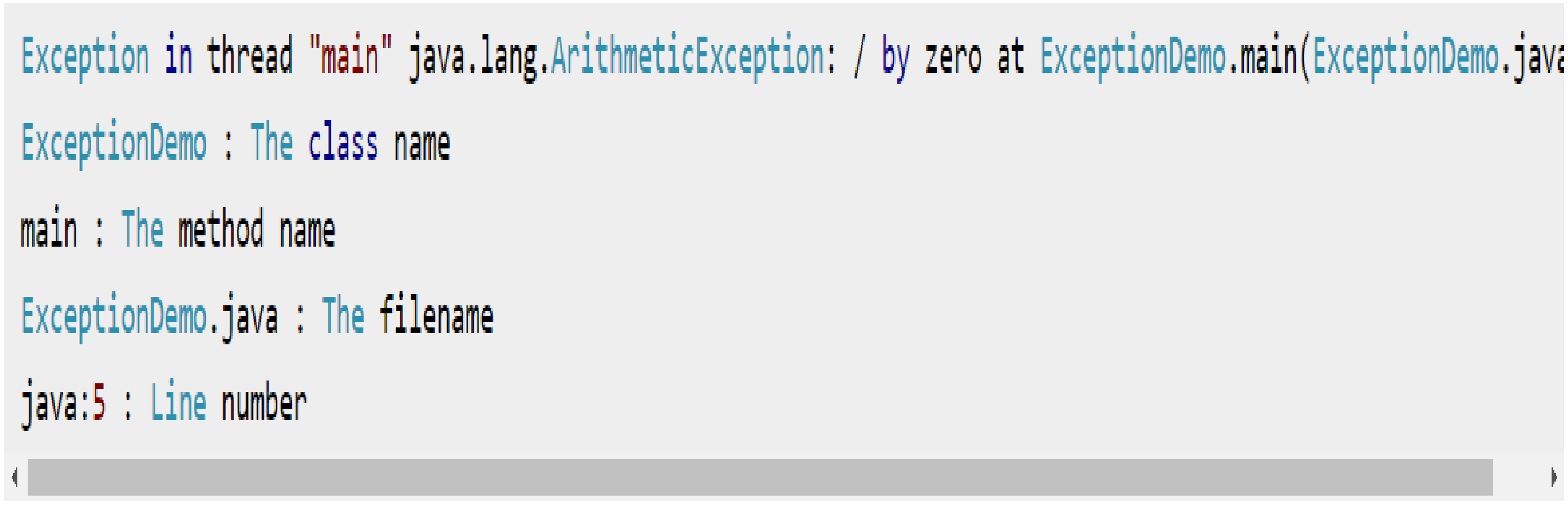
Exception Hierarchy

- All exceptions are sub-classes of the build-in class Throwable.
- Exception – exceptional conditions that programs should catch
- Throwable contains two immediate sub-classes:
    - The class includes:

      a**) Runtime Exception** – defined automatically for user programs to include

      b) **User-defined Exception** classes

Look at ERROR…



```
Exception in thread "main" java.lang.ArithmeticException: / by zero at ExceptionDemo.main(ExceptionDemo.java

ExceptionDemo : The class name

main : The method name

ExceptionDemo.java : The filename

java:5 : Line number
```

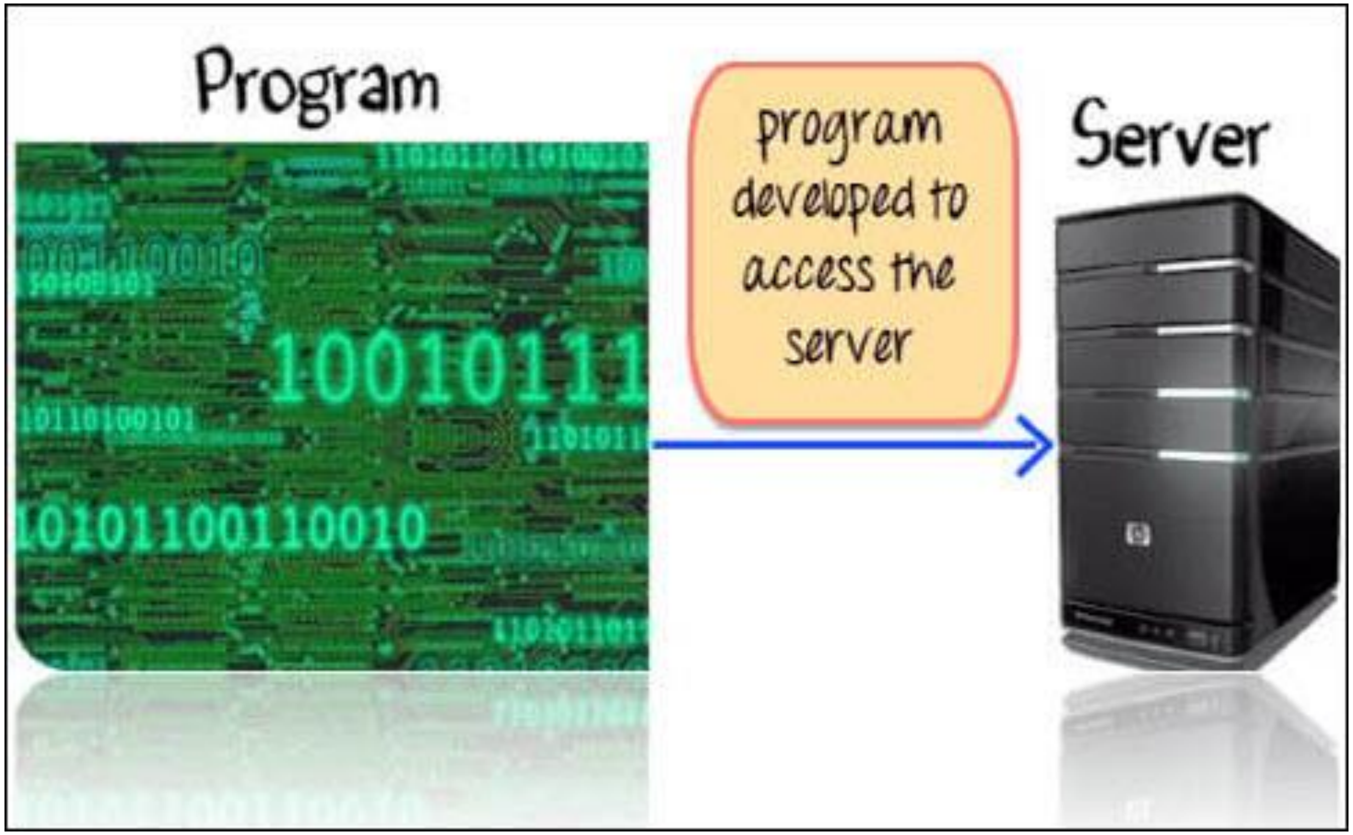
Difference between error and exception

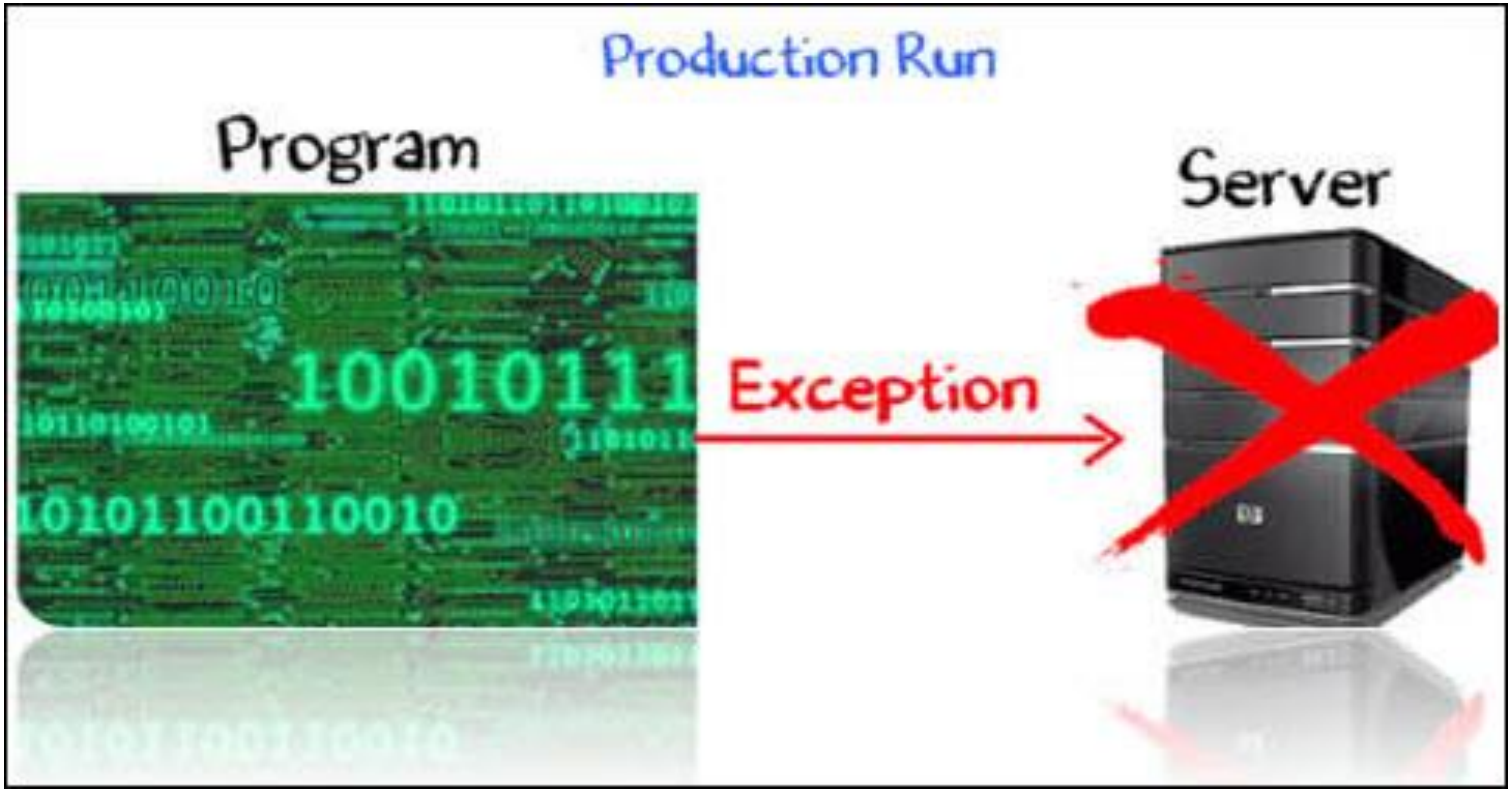
- **Errors** indicate that something **severe enough has gone wrong**, the application should crash rather than try to handle the error.
- **Exceptions** are events that occurs in the code. A **programmer can handle** such conditions and take necessary corrective actions

Exception Constructs

- Five constructs are used in exception handling: try, catch, finally, throw and throws
  - **try** – a block surrounding program statements to monitor for exceptions
  - **catch** – together with try, catches specific kinds of exceptions and handles them in some way
  - **finally** – specifies any code that absolutely must be executed whether or not an exception occurs

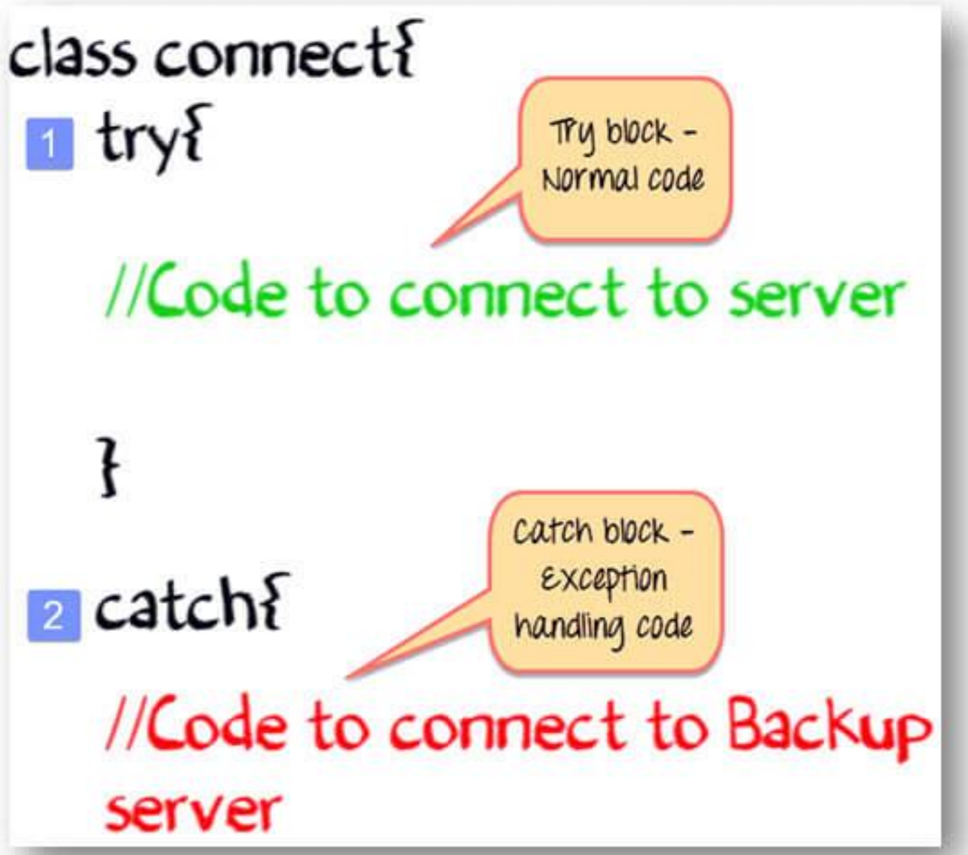
Example




Production Run
Program
10010111
10101100110010
Exception
Server

Try Catch Block

1. The normal code goes into a **TRY** block.
2. The exception handling code goes into the **CATCH** block

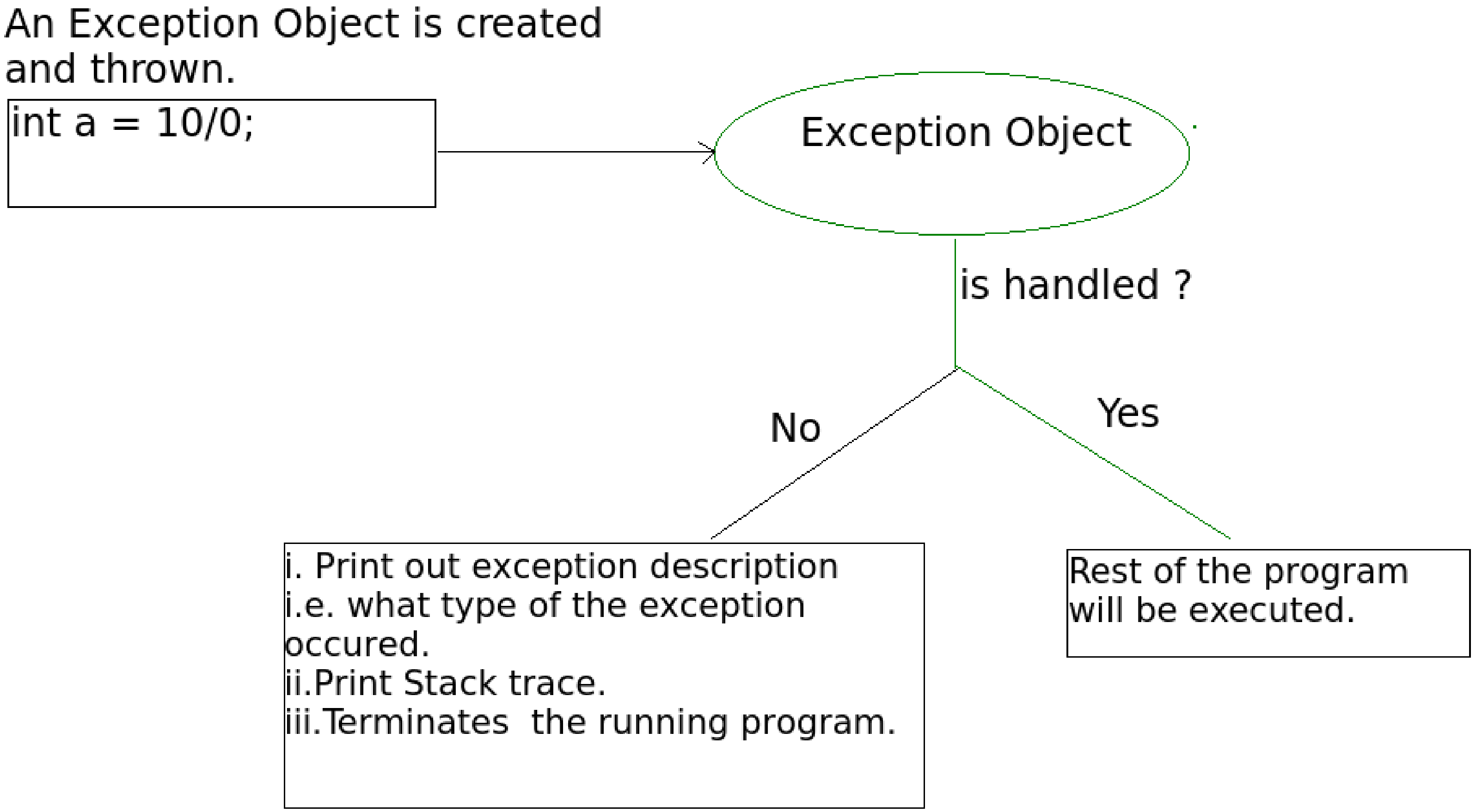


```
class connect{
1 try{
    //Code to connect to server
  }
2 catch{
    //Code to connect to Backup
    server
```

Exception-Handling Block

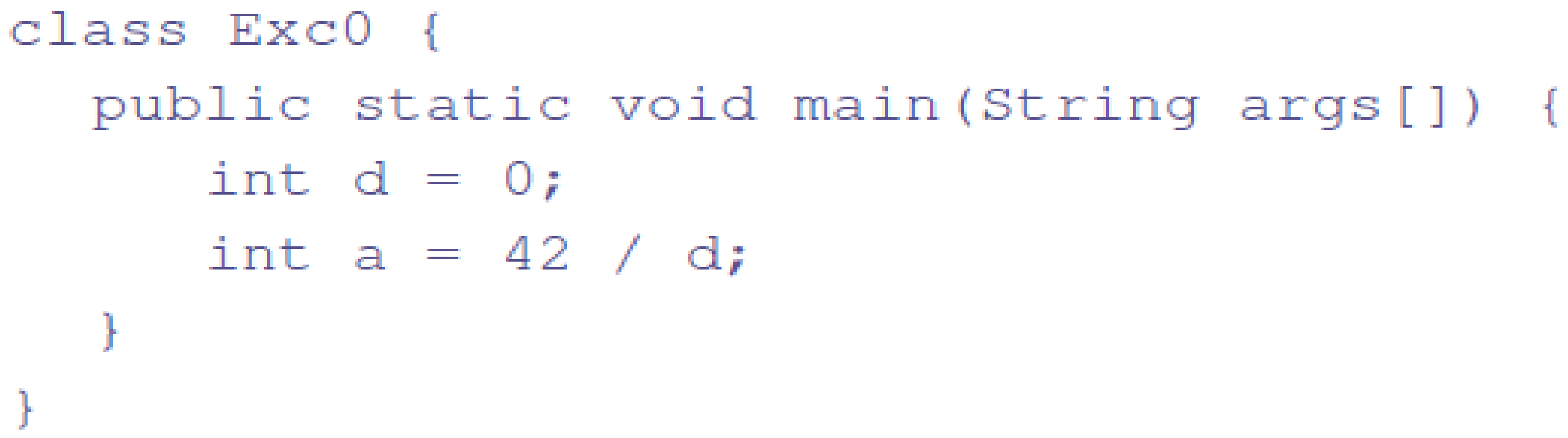
- General form:

```
try { … }
catch(Exception1 ex1) { … }
catch(Exception2 ex2) { … }
…
finally { … }
```


An Exception Object is created and thrown.
int a = 10/0;
Exception Object
is handled ?
No
Yes
i. Print out exception description i.e. what type of the exception occured.
ii.Print Stack trace.
iii.Terminates the running program.
Rest of the program will be executed.

Uncaught Exception

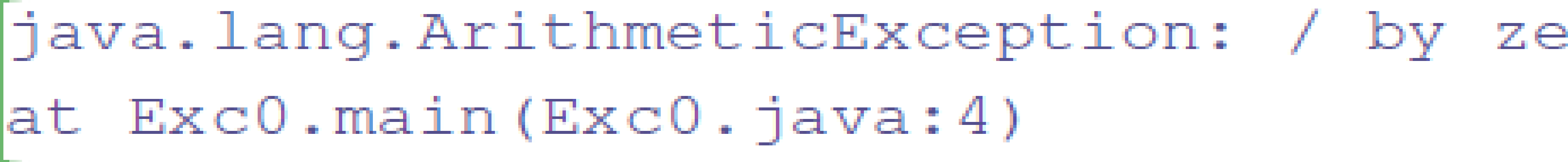
- What happens when exceptions are not handled?



```
class Exc0 {
  public static void main(String args[]) {
     int d = 0;
     int a = 42 / d;
  }
}
```

Default Exception Handler

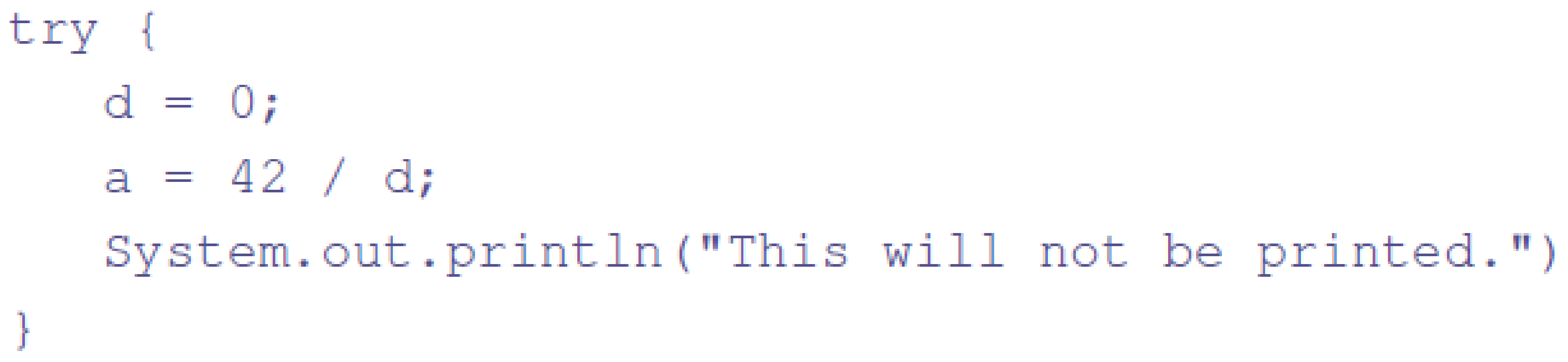
- This default handler:
  - displays a string describing the exception,
  - terminates the program



```
java.lang.ArithmeticException: / by ze
at Exc0.main(Exc0.java:4)
```

Try and Catch

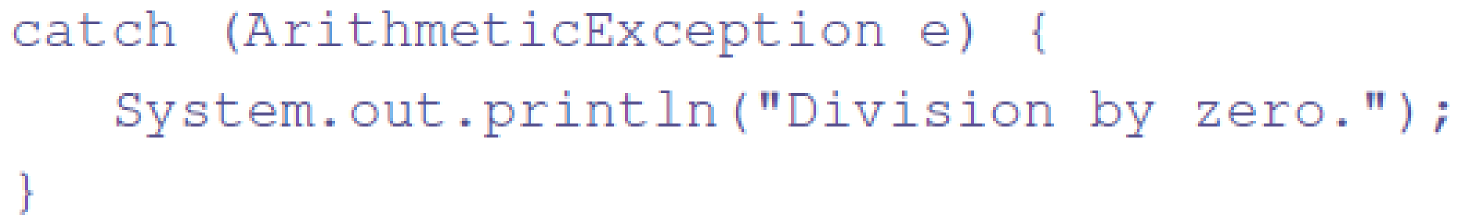
- Try and catch:
  - try surrounds any code we want to monitor for exceptions
  - catch specifies which exception we want to handle and how.
- When an exception is thrown in the try block:



```
try {
    d = 0;
    a = 42 / d;
    System.out.println("This will not be printed.")
}
```

Try and Catch

- control moves immediately to the catch block:



```
catch (ArithmeticException e) {
    System.out.println("Division by zero.");
}
```

Exception Display

- All exception classes inherit from the Throwable class.

```
try { … }
catch (ArithmeticException e) {
    System.out.println("Exception: " + e);
}
```

- The following text will be displayed:

```
Exception: java.lang.ArithmeticException: / by zero
```

Multiple Catch Clauses

- When more than one exception can be raised by a single piece of code,
- several catch clauses can be used with one try block:

```
try { statement(s)}
catch (ExceptiontType name){
        statement(s)}
catch (ExceptiontType name){
        statement(s)}
```

Example: Multiple Catch Order

- A try block with two catch clauses:

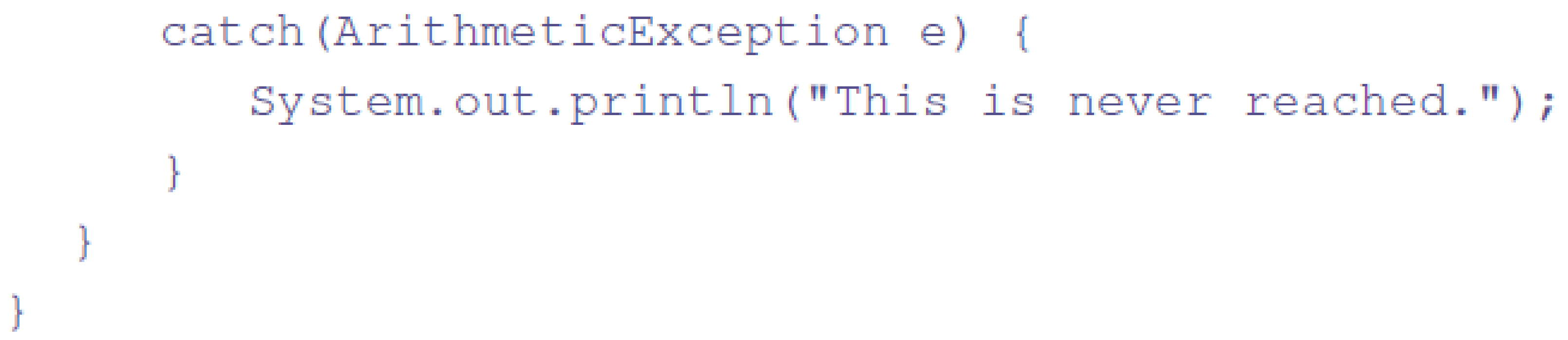
```
class SuperSubCatch {
  public static void main(String args[]) {
     try {
        int a = 0;
        int b = 42 / a;
```

- This exception is more general but occurs first:

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

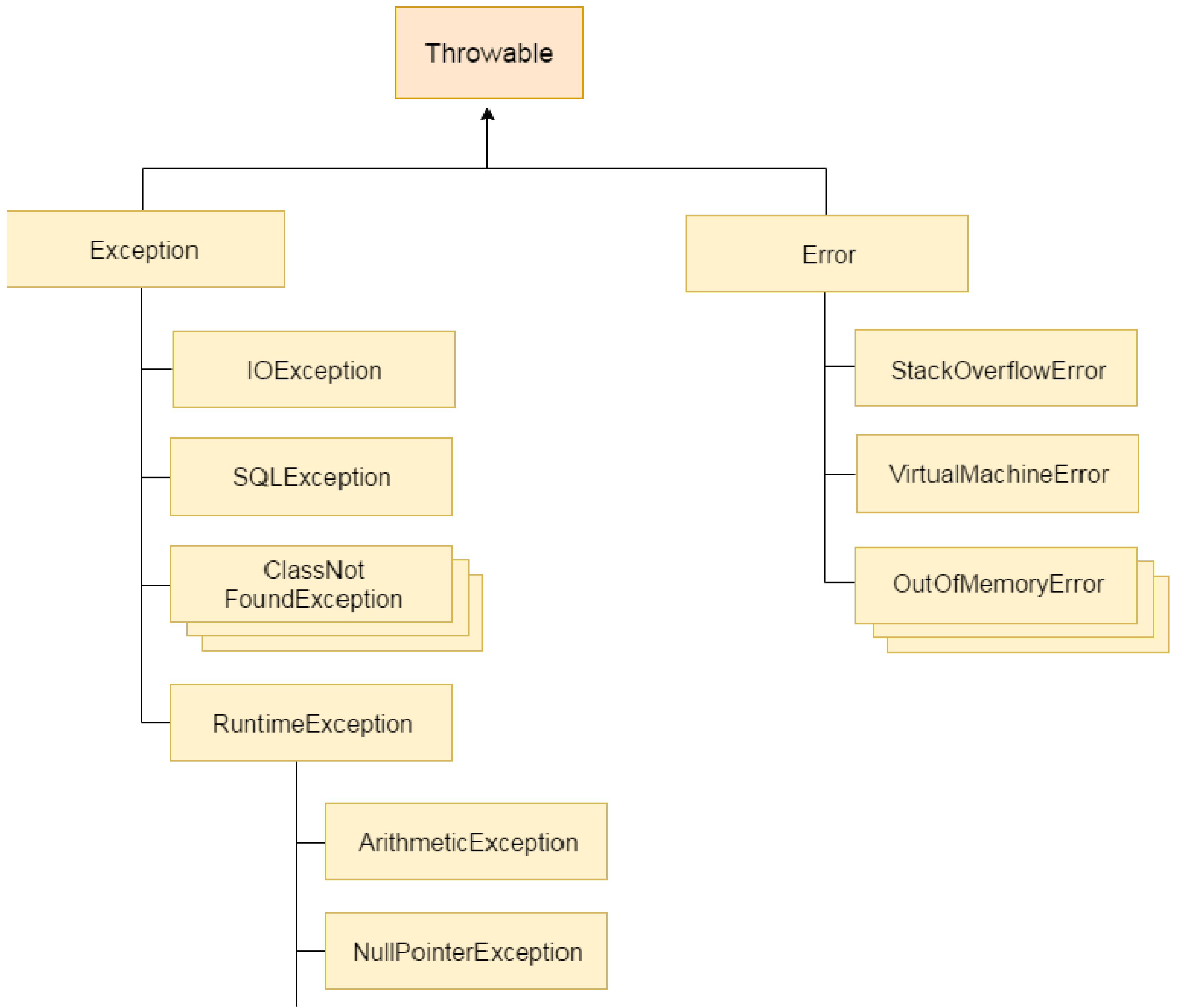
Example: Multiple Catch Order

- This exception is more specific but occurs last:



```
            catch(ArithmeticException e) {
               System.out.println("This is never reached.");
            }
      }
  }
```

- The second clause will never get executed. A compile-time error (unreachable code) will be raised.


Throwable
Exception
Error
IOException
SQLException
ClassNot FoundException
RuntimeException
ArithmeticException
NullPointerException
StackOverflowError
VirtualMachineError
OutOfMemoryError

1) How ArithmeticException occurs

```java
int a=50/0;//ArithmeticException
```

2) How NullPointerException occurs

```java
String s=null;
System.out.println(s.length());//NullPointerException
```

3) How NumberFormatException occurs

```java
String s="abc";
int i=Integer.parseInt(s);//NumberFormatException
```

4) How ArrayIndexOutOfBoundsException occurs

```java
int a[]=new int[5];
a[10]=50; //ArrayIndexOutOfBoundsException
```

Your task

Throw vs Throws

Questions