



ሐምሌ, 2013 ዓ.ም

# Object Oriented Programming



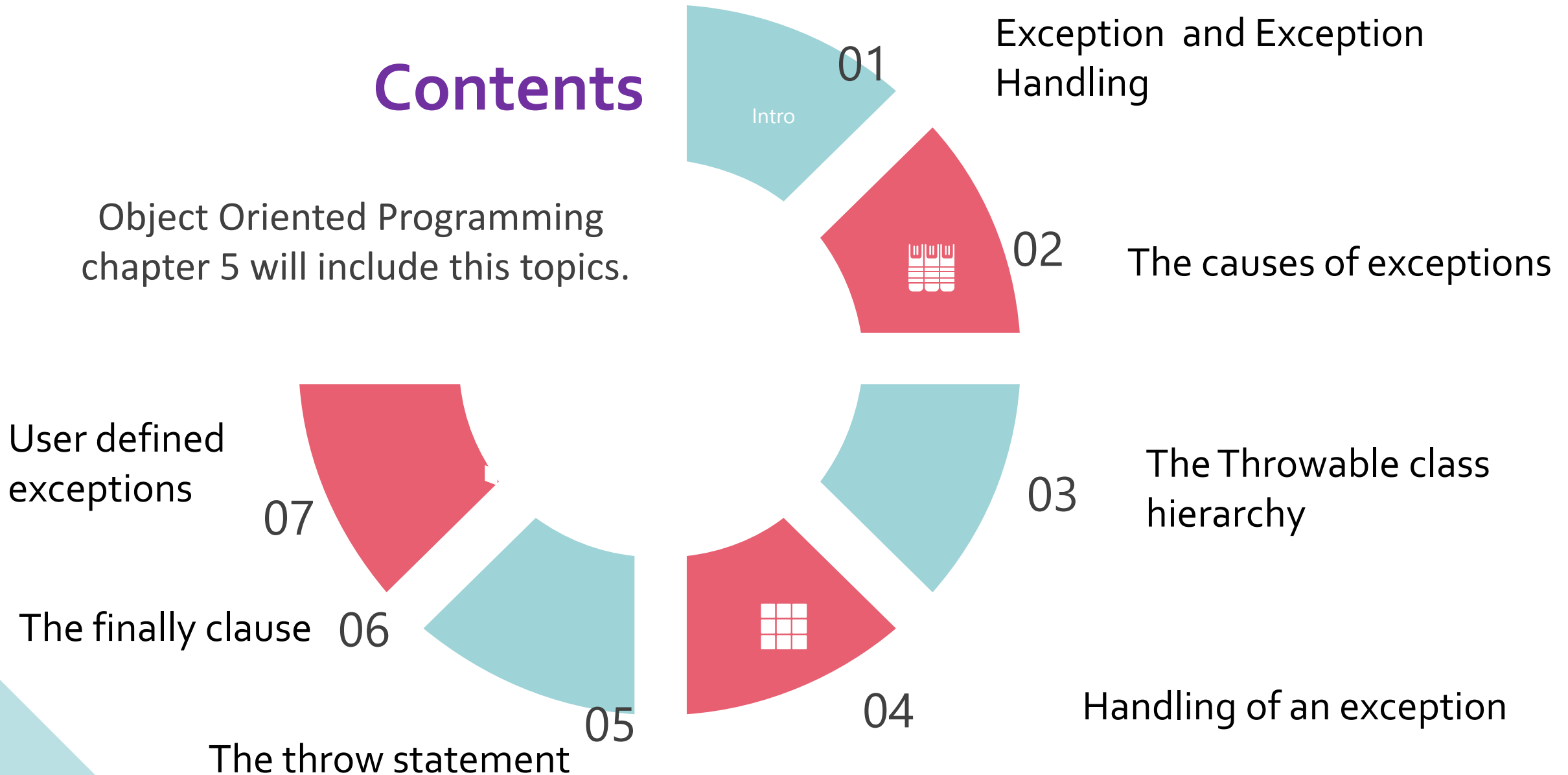
## Chapter Five

Wollo University, Kombolcha Institute of Technology.  
College of Informatics.  
Department of Information System.

By Daniel G.

# Contents

Object Oriented Programming  
chapter 5 will include this topics.



# Basics questions In this chapter

- 1. What is exception?*
- 2. What is exception handling?*
- 3. What are the cause of exception?*
- 4. How to handle exception?*



- ❖ **An Exception** is an **event** which occurs during the **execution** of a program, that **disrupts** the normal program **flow** and may cause a program to **fail**.
- ❖ It is an **object** which is **thrown** at **runtime**.

### ❖ Some **examples** of exception

- ✓ Performing **illegal arithmetic** (division by zero)
- ✓ **Illegal arguments** to methods
- ✓ Accessing an **out-of-bounds array** element
- ✓ Writing to a read-only file
- ✓ Opening a non-existing file

## Example

```
public class ExceptionExample {  
    public static void main(String args[]) {  
        String[] greek = {"Alpha", "Beta"};  
        System.out.println(greek[2]);  
    }  
}
```

The out put will be:

```
java.lang.ArrayIndexOutOfBoundsException
```

```
public static int average(int[] a) {  
    int total = 0;  
    for(int i = 0; i < a.length; i++) {  
        total += a[i];  
    }  
    return total / a.length;  
}
```

Program **throws** an Exception and **fails**

`java.lang.ArithmeticException`: / by zero

## ❖ Exception Message Details

Exception message format

[**exception class**]: [additional **description** of exception]  
at [**class**].[**method**](**[file]:[line number]**)

### Example:

```
java.lang.ArrayIndexOutOfBoundsException: 2  
at ExceptionExample.main(ExceptionExample.java:4)
```



- ❖ **Exception Handling** is a mechanism to handle runtime errors such as `ClassNotFoundException`, `IOException`, `SQLException`, `RemoteException`, etc.
- ❖ All of these problems have to be handled using **exception handling**.
- ❖ Use a **try-catch** block to handle exceptions that are **thrown**

```
try {  
    // code that might throw exception  
}  
catch ([Type of Exception] e) {  
    // what to do if exception is thrown  
}
```

# Cont....

ከምሌ, 2013 ዓ.ም

```
public static int average(int[] a) {  
    int total = 0;  
    for(int i = 0; i < a.length; i++) {  
        total += a[i];  
    }  
    return total / a.length;  
}
```

# Cont....

ሐምሌ, 2013 ዓ.ም

```
public static void printAverage(int[] a) {  
    try {  
        int avg = average(a);  
        System.out.println("the average is: " + avg);  
    }  
    catch (ArithmeticException e) {  
        System.out.println("error calculating average");  
    }  
}
```

- ❖ Handle **multiple possible exceptions** by multiple successive catch blocks

```
try {  
    // code that might throw multiple exception  
}  
  
catch (IOException e) {  
    // handle IOException and all subclasses  
}  
  
catch (ClassNotFoundException e2) {  
    // handle ClassNotFoundException  
}
```

# Advantage of Exception Handling

ሐምሌ, 2013 ዓ.ም

- ❖ The **core advantage** of **exception** handling is **to maintain the normal flow of the application**.
- ❖ An **exception** normally **disrupts** the normal flow of the application that is why we use **exception handling**.

- ❖ An **abnormal execution** condition was synchronously detected by the Java virtual machine. Such conditions arise because:
  - ✓ **Evaluation** of an expression **violates** the normal semantics of the language, such as an integer divide by zero
  - ✓ An **error occurs** in loading or linking part of the program
  - ✓ Some **limitation on a resource** is exceeded, such as using too much memory

Keyword	Description
Try	The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally.
Catch	The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.

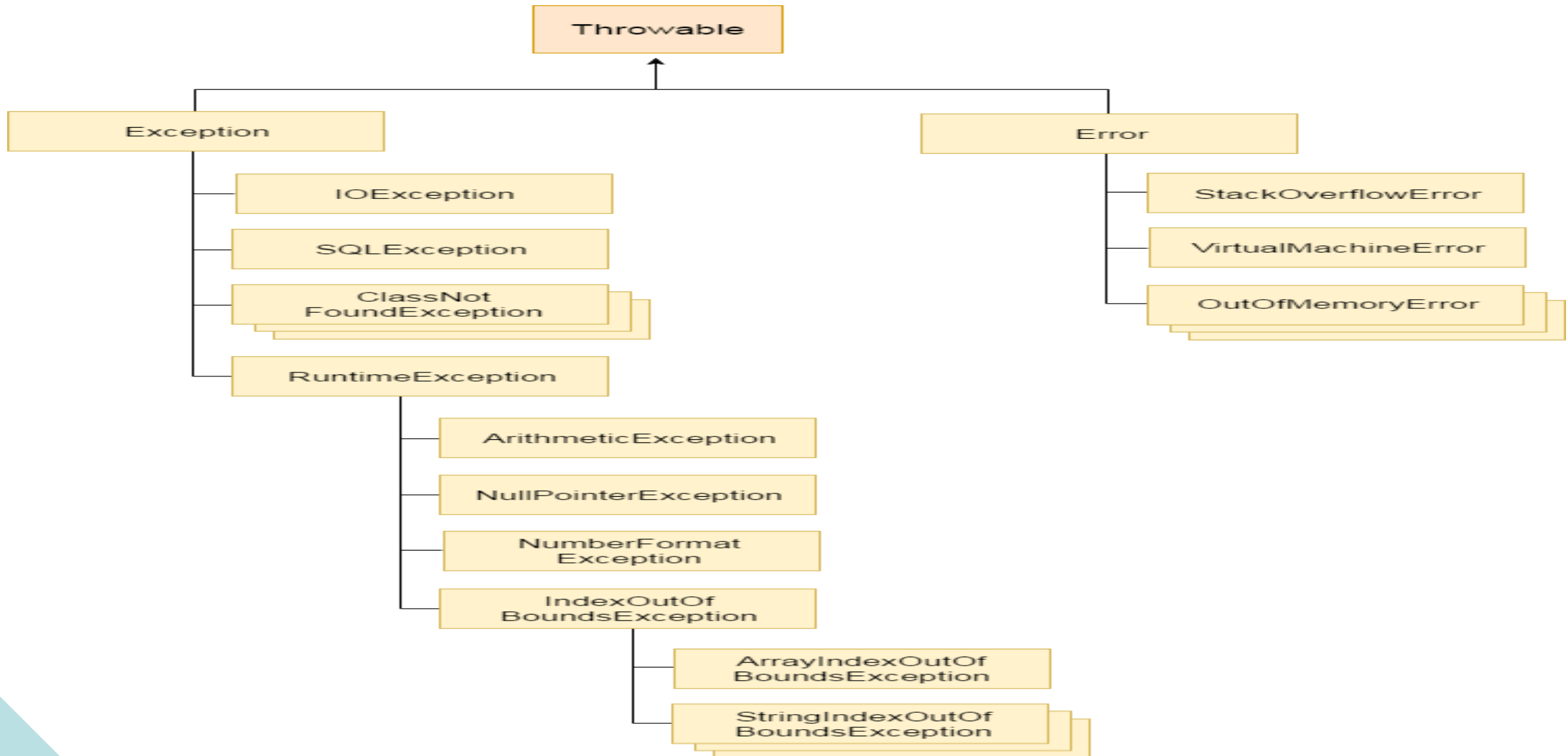
Keyword	Description
Finally	The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not.
Throw	The "throw" keyword is used to throw an exception.
Throws	The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signature.



# The Throwable class hierarchy

ሐምሌ, 2013 ዓ.ም

- ❖ The **class** at the **top** of the **exception** class hierarchy is the **Throwable class**, which is a direct subclass of the **Object class**. Throwable has two direct subclasses - **Exception** and **Error**.
- ❖ The **diagram** below shows the standard **exception** and **error** classes defined in Java, organized in the Java exceptions hierarchy:



# Types of Java Exceptions

ሐምሌ, 2013 ዓ.ም

- ❖ There are mainly **two** types of **exceptions**: **checked** and **unchecked**. Here, an **error** is considered as the unchecked exception.
- ❖ According to Oracle, there are three types of exceptions:
  1. **Checked** Exception
  2. **Unchecked** Exception
  3. **Error**

## 1. Checked Exception

The **classes** which directly inherit **Throwable** class except RuntimeException and Error are known as checked exceptions

**Example:**

- ✓ IOException
- ✓ SQLException etc.

**Checked** exceptions are checked at compile-time.

## Example:

```
Private static checkedExceptionWith Throws() throws FileNotFoundException  
{  
    File file = new File("not_existing_file.txt");  
    FileInputStream stream = new FileInputStream(file);  
}
```

## 2. Unchecked Exception

- ❖ The **classes** which inherit RuntimeException are known as **unchecked** exceptions
- ❖ **Example:** ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc.
- ❖ **Unchecked exceptions** are not checked at compile-time, but they are checked at runtime.

## Example:

```
private static void divideByZero() {  
    int numerator = 1;  
    int denominator = 0;  
    int result = numerator / denominator;  
}
```

## 3. Error

Error is irrecoverable

Example

- ✓ OutOfMemory Error
- ✓ VirtualMachine Error
- ✓ Assertion Error etc.



Example:

```
import java.util.*;

public class Heap {

    public static void main(String args[]) throws Exception{

        Integer[] array = new Integer[10000 * 10000];

    }

}
```

- ❖ Both **throw** and **throws** are concepts of **exception** handling in Java.
- ❖ The **throws** keyword is used to declare which exceptions can be thrown from a method, while the **throw** keyword is used to explicitly throw an exception within a method or block of code.

- ❖ The **throws** keyword in Java is used to **declare exceptions** that can occur during the execution of a program.
- ❖ For any **method** that can throw exceptions, it is mandatory to use the **throws** keyword to list the exceptions that can be thrown.

- ❖ The **throws** keyword provides **information** about the exceptions to the programmer as well as to the caller of the method that **throws** the exceptions

- ❖ The `throw` keyword in Java is used for **explicitly throwing** a single exception.
- ❖ This can be from within a method or any block of code. Both checked and unchecked exceptions can be thrown using the `throw` keyword.

Syntax:

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

## Example

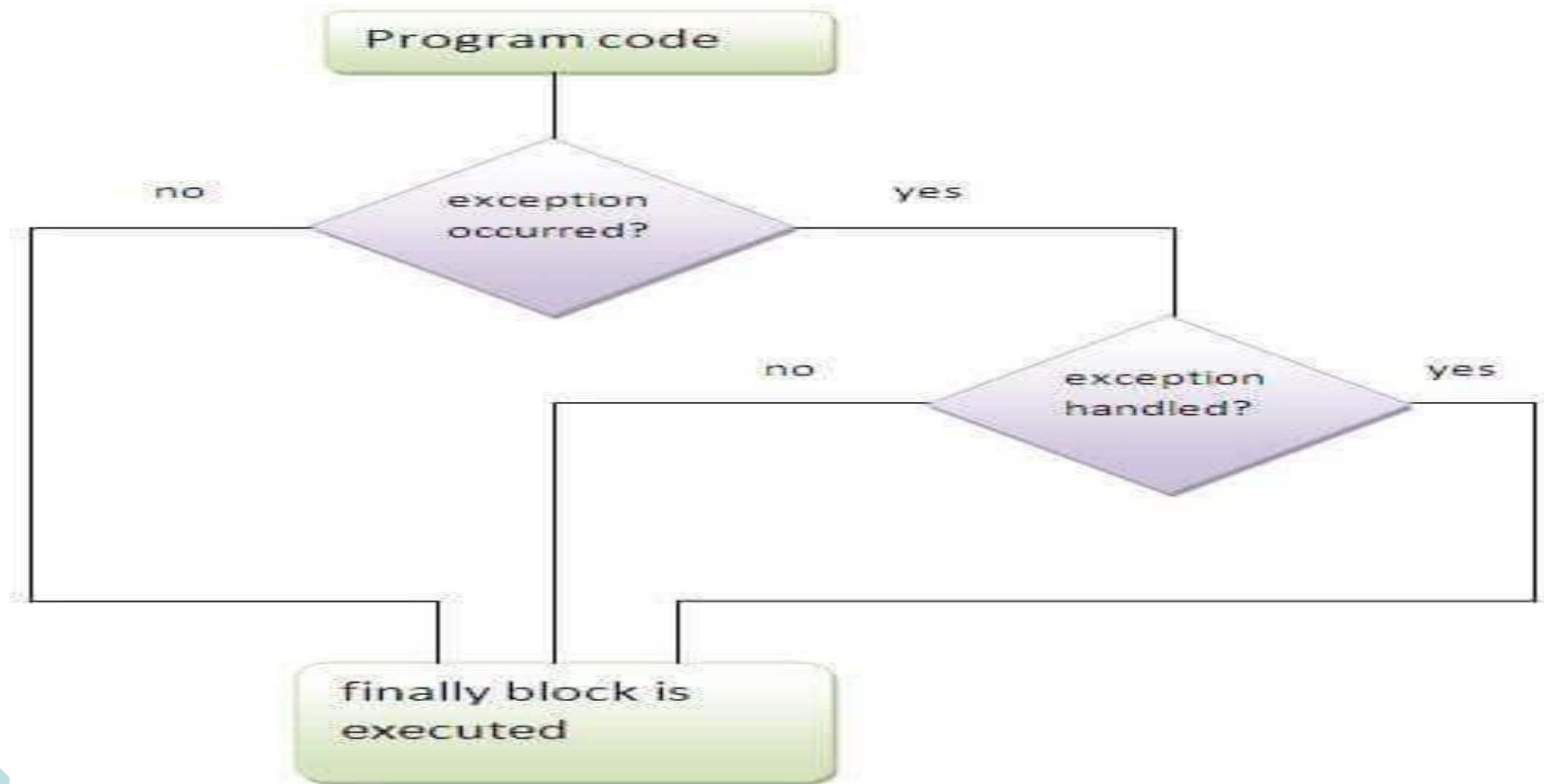
```
class ThrowsExecp{
    static void fun() throws IllegalAccessException{
        System.out.println("Inside fun(). ");
        throw new IllegalAccessException("demo");
    }
    public static void main(String args[]){
        try{
            fun();
        }
        catch(IllegalAccessException e){
            System.out.println("caught in main.");
        }
    }
}
```

# The Finally Clause

ሐምሌ, 2013 ዓ.ም

- ❖ The finally block in java is used to put important codes such as clean up code e.g. **closing** the file or **closing** the connection.
- ❖ The **finally** block executes whether exception rise or not and whether exception handled or not.
- ❖ A **finally** contains all the crucial statements regardless of the exception occurs or not.





## Example

```
import java.io.*;
class Newclass {
    public static void main(String[] args)
    {
        try {
            System.out.println("inside try block");
            System.out.println(34 / 0);
        }
        catch (ArithmeticException e) {

            System.out.println("catch : exception handled.");
        }
        finally {
            System.out.println("finally : i execute always.");
        }
    }
}
```

- ❖ If you are creating your own Exception that is known as custom exception or user-defined exception.
- ❖ Java custom exceptions are used to customize the exception according to user need.

## Cont.....

ሐምሌ, 2013 ዓ.ም

```
class MyException extends Exception{
}
// A Class that uses above MyException
public class setText{
    public static void main(String args[]){
        try{
            // Throw an object of user defined exception
            throw new MyException();
        }
        catch (MyException ex){
            System.out.println("Caught");
            System.out.println(ex.getMessage());
        }
    }
}
```

ሐምሌ, 2013 ዓ.ም

# THANK YOU

---

**Any questions?**  
Feel free !