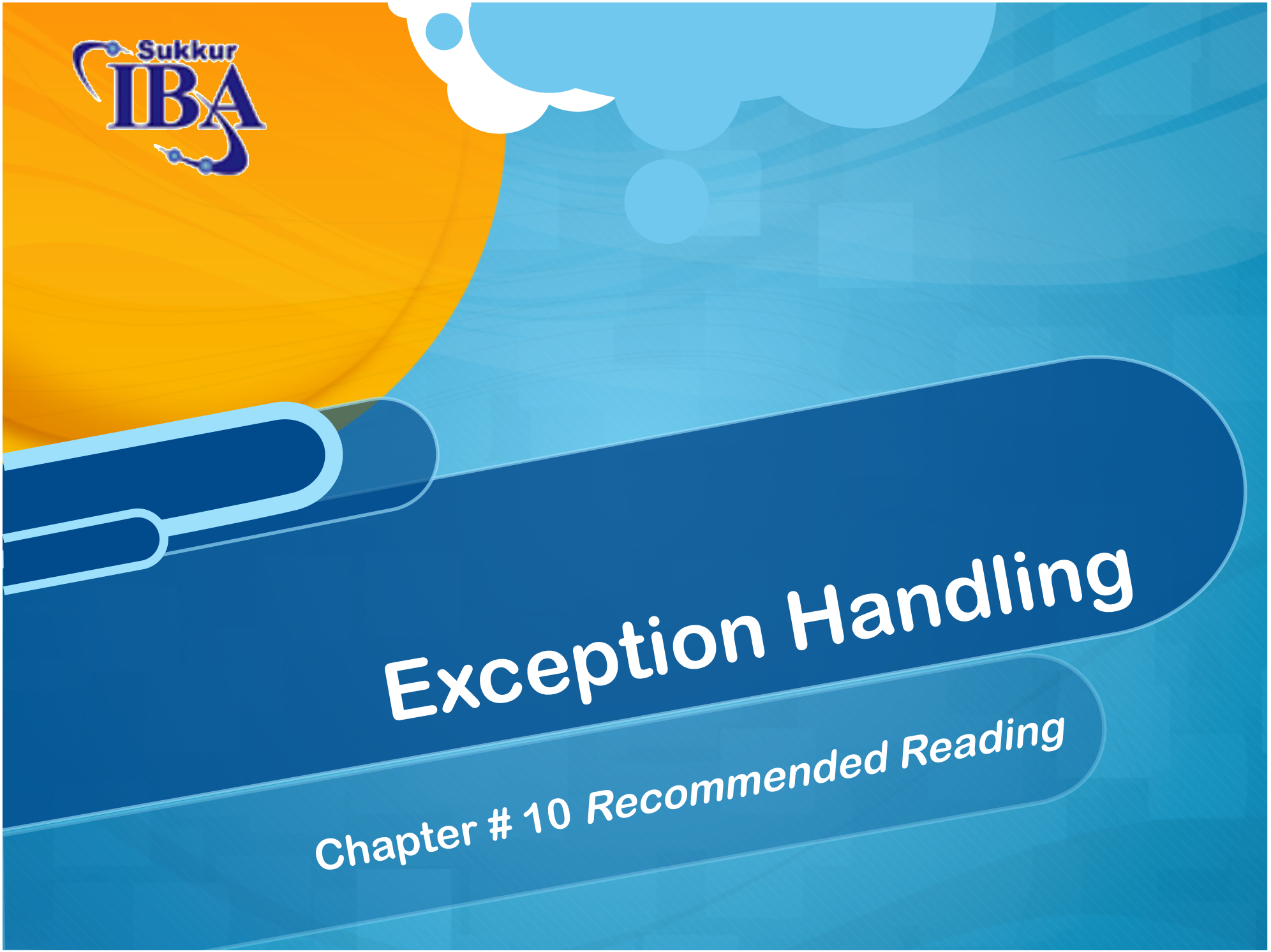
Fahim Akhtar Rajputt

Object Oriented Programming (JAVA)

Lecture 31 and 32


Sukkur
IBA
Exception Handling
Chapter # 10 Recommended Reading

Contents

- Exception
- Exception Handling
- Errors
- Try, catch and finally blocks
- Things to remember
- Exception propagation
- Throws (Way 1 and 2)
- Re-throw
- Creating own Exceptions
- Checked and Unchecked Exceptions

Exception

- Exception is an abnormal condition or an event that happens during execution of code, which interrupts the normal flow of code.
- Exception can occur for many reasons
  - User has entered invalid data
  - File that need to be opened couldn't found
  - A network connection has been lost in the middle of communication
  - Suddenly JVM has ran out of memory etc…

Example

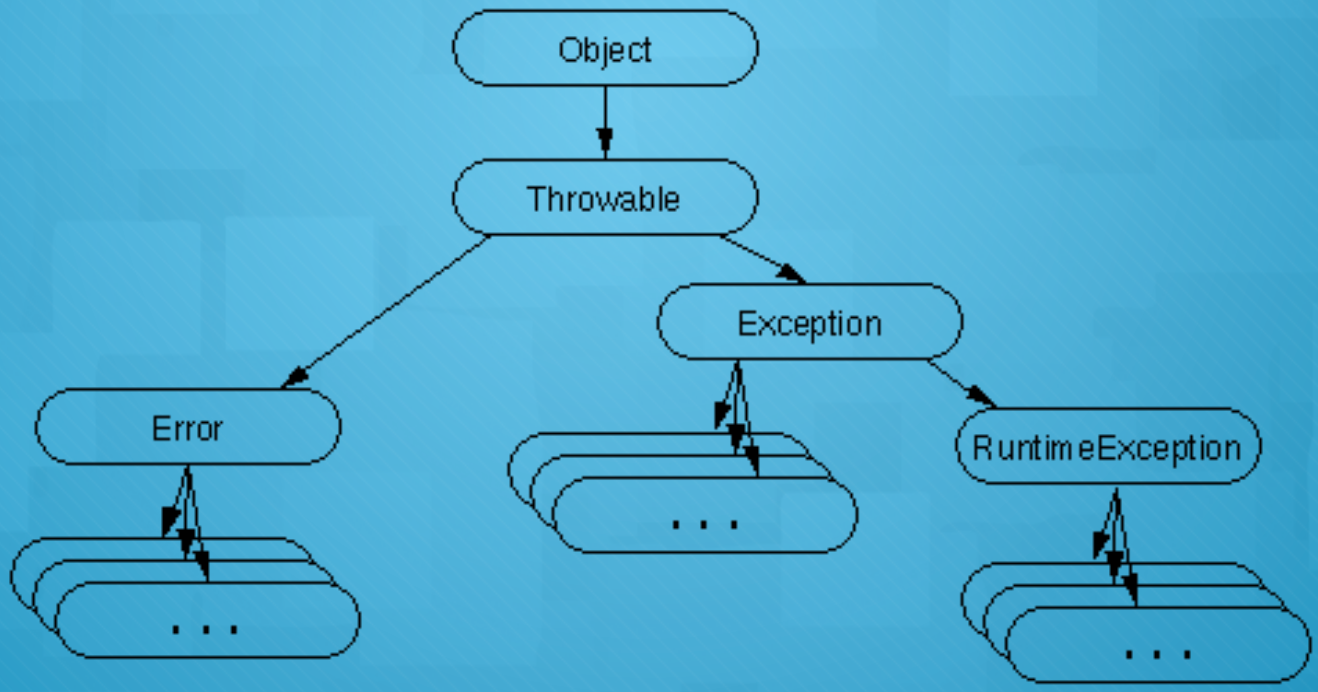
```
double division (double a, double b) {
        double c = a/b;
        return c;
}
```

Java Provides a way to handle such exceptions.
Called it as Exception Handling

Exception Handling

- Java's exception handling enables your Java applications to handle exceptions sensibly.
- When an exception occurs in a Java program it usually results in an exception being thrown.
- Exception occurs only in run time during program execution
- Some keywords:
  - try, catch, finally, throws and throw

Exception Hierarchy



Errors

- Errors are irrecoverable situations that occurs during program execution
- Once occurs the application can't be recovered from and come to halt.
- Errors need not to be handled
- Examples:
  - java.lang.OutOfMemoryError
  - java.lang.StackOverFlow

Example – Error

```
public class Test{
	public void method1(){
		this.method2();
	}
	public void method2(){
		this.method1();
	}

	public static void main (String args[]){
		Test errorDemo = new Test ();
		ed.method1();
	}
}
```

```
Exception in thread "main" java.lang.StackOverflowError
at Test.method2(Test.java:6)
at Test.method1(Test.java:3)
at Test.method2(Test.java:6)
at Test.method1(Test.java:3)
at Test.method2(Test.java:6)
at Test.method1(Test.java:3)
at Test.method2(Test.java:6)
at Test.method1(Test.java:3)
.
.
.
at Test.method1(Test.java:3)
```

Exceptions (try, catch and finally)

Unlike the errors, exceptions can be handled

```
try{
        //Some code, might throw an exception
}
catch (<ExceptionType> <name>) {
        //Control comes here
}
finaly{
        // release some resources
}
```

try, catch and finally – Example

```
import java.io.*;
public class ExcEg{
        public void FileIOOperation(){
                try{
                        FileReader fr = new FileReader("MyFile.txt");
                }
                catch(FileNotFoundException e){
                        System.out.println(e.getMessage());
                }
                finally{
                        System.out.println("I will execute always");
                }
        }
        public static void main (String args []){
                ExcEg eeg = new ExcEg ();
                eeg. FileIOOperation();
        }
}
```

OUTPUT

MyFile.txt (No such file or directory)
I will execute always

Things to remember [1/2]

- At least, catch or finally block should be there accompanying with try block, both catch and finally can also be there
- No statement is allowed in between try, catch and finally.
- There can be multiple catch blocks
- One finally block for a try block – and its optional
- Catch block will execute only when exception is thrown

Things to remember [2/2]

- When Child and Parent relation is there in exceptions, the catch block of parent always comes after child. (Narrow exceptions comes first and broader exceptions in last)
- Finally always runs even when there's no exception.
- In a try block after occurrence of exception, no remaining LOCs will execute.

Questions?

References

- http://docs.oracle.com/javase/tutorial/essential/exceptions/


Sukkur
IBA
Thanks…