Learning Goals

Click to Continue

After completing this chapter you will be able to

- What is an exception?
- Advantages of exception handling framework.
- How to handle exception.
- Create User Defined Exception.

What is an Exception?

Click to Continue

"**Exception** " refers to any abnormality or an error that occurs during run time.

- An exception in Java is a signal that indicates the occurrence of a unexpected condition during execution of a program at runtime.
- Exception causes normal program flow to be disrupted.

Examples :

- int num=5/0 – Divide by Zero Error –Arithmetic Exception
- Trying to open a file that has been deleted – File not found Exception.

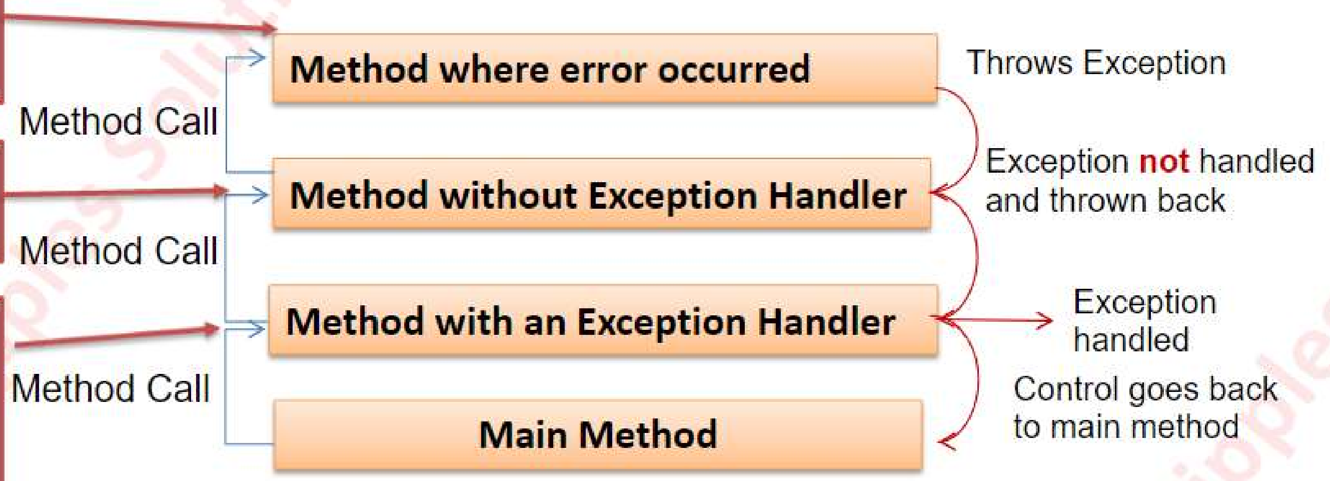
Exception flow in a program?

Click to Continue

This is the final method where the error occurs.

This in turn invokes another method.

Here main method invokes a method in another class which has a exception handler.


Method where error occurred
Throws Exception
Method Call
Method without Exception Handler
Exception **not** handled and thrown back
Method Call
Method with an Exception Handler
Exception handled
Method Call
Main Method
Control goes back to main method

If confusing don't worry! We will see about how to handle the exceptions in the next slides.

Try it Out - Exception

Click to Continue

Try to run this program in your IDE

```
public class DivByZero {
    public static void main(String args[]) {
        System.out.println(3/0);
        System.out.println("Pls. print me");
        }
}
```

You can notice the following exception thrown by the run time system,

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at com.cognizant.academy.handson.DivByZero.main(DivByZero.java:3)
```

Exception Benefits

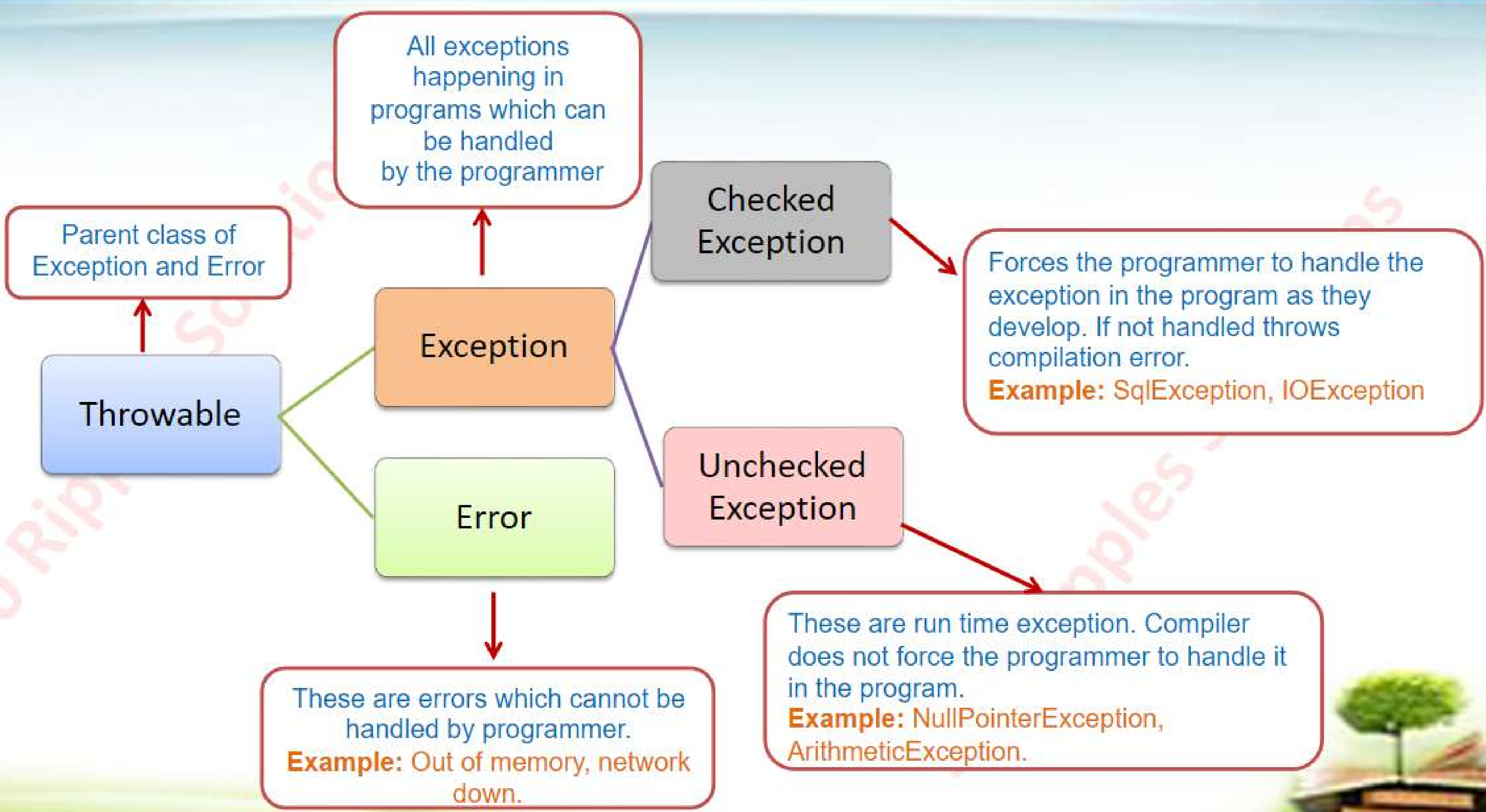
Click to Continue

Benefits of Java Exception Handling Framework:

- It separates Error-Handling code from “regular” business logic code.
- It can propagate errors up the call stack till a handler handles the exception.
- It can group and categorize the exception types.

Exception Hierarchy

Click to Continue


All exceptions happening in programs which can be handled by the programmer
Parent class of Exception and Error
Checked Exception
Exception
Throwable
Error
Unchecked Exception

Forces the programmer to handle the exception in the program as they develop. If not handled throws compilation error.
Example: SqlException, IOException

These are run time exception. Compiler does not force the programmer to handle it in the program.
Example: NullPointerException, ArithmeticException.

These are errors which cannot be handled by programmer.
Example: Out of memory, network down.

Checked Vs Unchecked Exception

Click to Continue

Checked Exception	Unchecked Exception
At compile time, the java compiler automatically checks that a program contains handlers for checked exceptions	The compiler doesn't force them to be declared in the throws clause.
Checked exceptions must be explicitly caught or propagated using try-catch-finally blocks	Unchecked exceptions do not have this requirement. They don't have to be caught or declared thrown.
Checked exceptions in Java extend the java.lang.Exception class	Unchecked exceptions extend the java.lang.RuntimeException.
Exception handling is mandated by JVM for these exceptions	It is not advisable to catch these exceptions since it might make the code unstable.
Example: IOException	**Example:** NullPointerException

Examples of Checked Exception

Click to Continue

Checked Exception	Description
ClassCastException	This Exception occurs when Java run-time system fail to find the specified class mentioned in the program.
InstantiationException	This Exception occurs when you create an object of an abstract class and interface.
IllegalAccessException	This Exception occurs when you create an object of an abstract class and interface
NoSuchMethodException	This Exception occurs when the method you call does not exist in class.

Examples of Unchecked Exception

Click to Continue

Unchecked Exception	Description
ArithmeticException	These Exception occurs, when you divide a number by zero causes an Arithmetic Exception
ClassCastException	These Exception occurs, when you try to assign a reference variable of a class to an incompatible reference variable of another class.
NullPointerException	These Exception occurs, when you try to invoke a method on a object without instantiating it.
ArrayIndexOutOf Bounds Exception	These Exception occurs, when you assign an array which is not compatible with the data type of that array.