Course code : **CSE2005**

Course title : **Object Oriented Programming**

Exception Handling

Objectives

This session will give the knowledge about

- Introduction to Exception Handling
- Exception Handling Techniques
- Exception Handling Keywords
- Types of Exceptions

Exception Handling

When we write programs as part of an application, we may have to visualize the challenges that can disrupt the normal flow of execution of the code.

Once we know what are the different situations that can disrupt the flow of execution, we can take preventive measures to overcome these disruptions.

In java, this mechanism comes in the form of Exception Handling.

What is an Exception?

In procedural programming, it is the responsibility of the programmer to ensure that the programs are error- free in all aspects

Errors have to be checked and handled manually by using some error codes

But this kind of programming was very cumbersome and led to spaghetti code. Java provides an excellent mechanism to handle the exception at runtime.

An exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions.

What is an Exception?

The ability of a program to intercept run-time errors, take corrective measures and continue execution is referred to as exception handling.

There are various situations when an exception occur:

- Attempting to access a file that does not exist
- Inserting an element into an array at a position that is not in its bounds
- Performing some mathematical operation that is not permitted
- Declaring an array using negative values

Uncaught Exceptions

```
public class Main {
        public static void main(String[] ar){
                int x=34;
                int y=x/0;
        }
}
```

Although this program will compile, but when you execute it, the Java run-time-system will generate an exception and displays the following output:

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at vit.demo.Main.main(Main.java:8)
```

Exception Handling Techniques

There are several built-in exception classes that are used to handle the very fundamental errors that may occur in your programs

You can create your own exceptions also by extending the Exception class

These are called user-defined exceptions, and will be used in situations that are unique to your applications

Handling Runtime Exceptions

Whenever an exception occurs in a program, an object representing that exception is created and thrown in the method in which the exception occurred

Either you can handle the exception, or ignore it

In the latter case, the exception is handled by the Java run-time-system and the program terminates

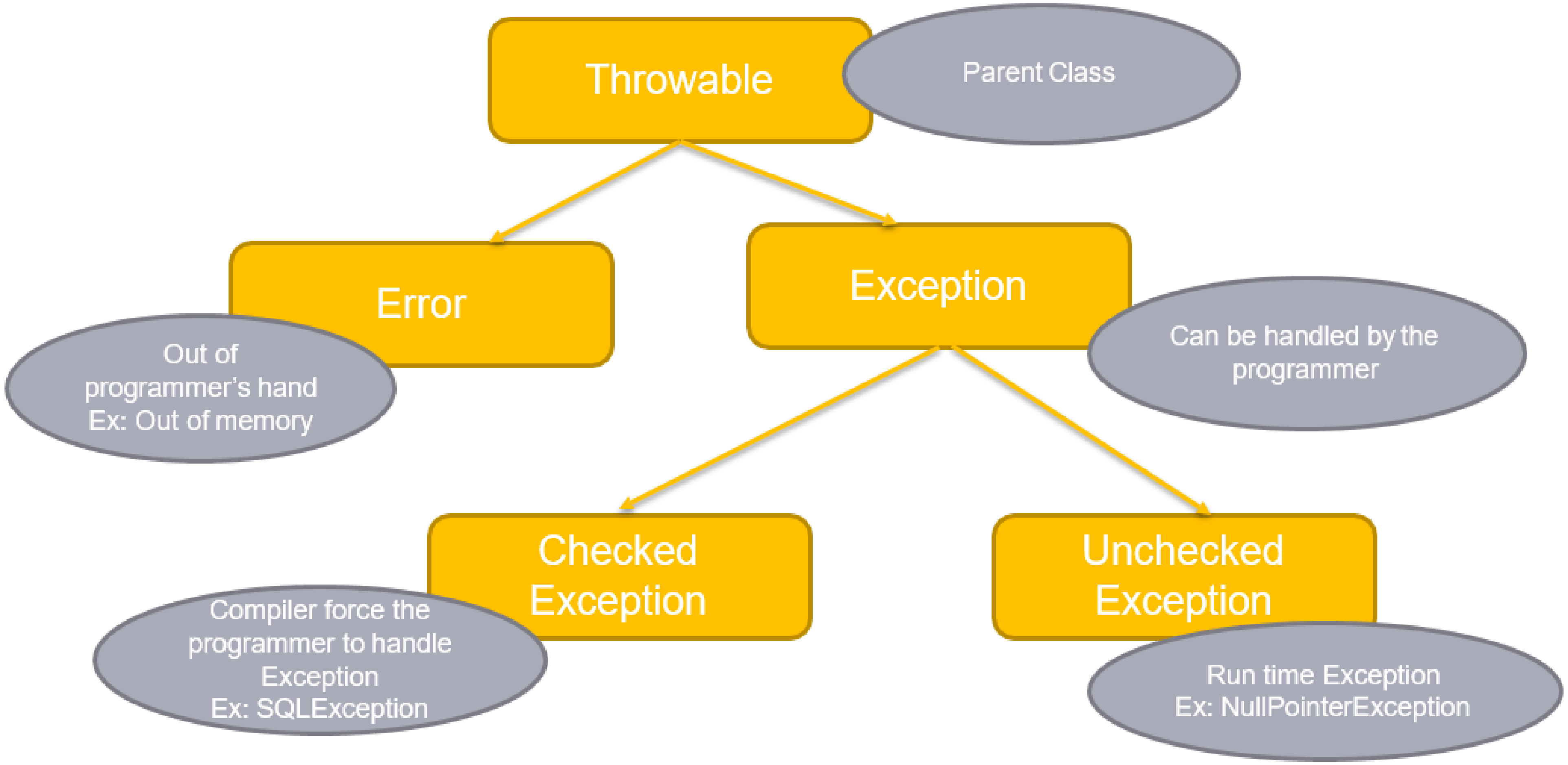
However, handling the exceptions will allow you to fix it, and prevent the program from terminating abnormally

Exception Handling Keywords

Java’s exception handling is managed using the following keywords: try, catch, throw, throws and finally.

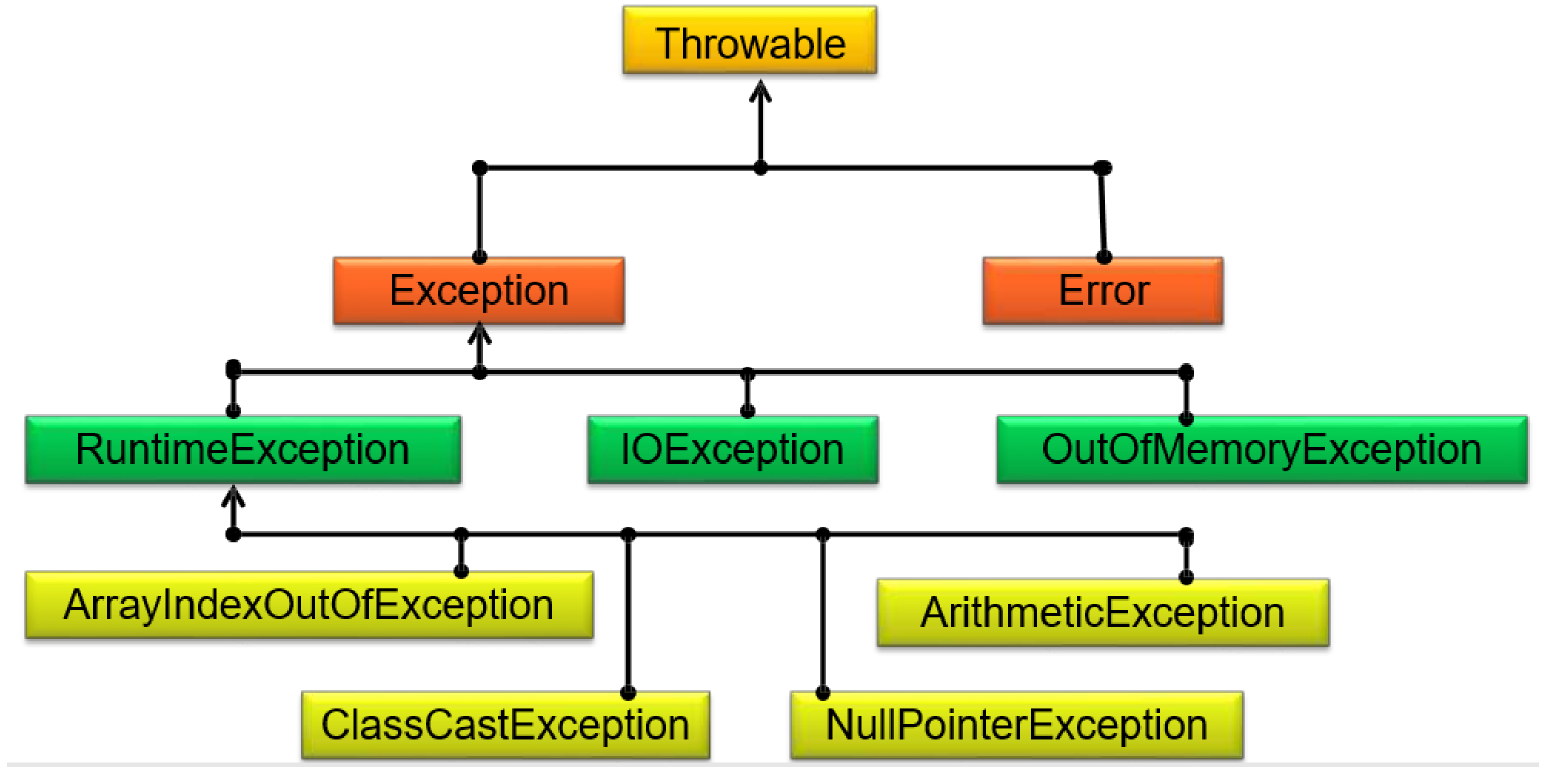
```
public static void main(String[] ar){
        try{
        }
        catch(TypeOfException obj){
        }
        finally{
        }
}
```

Types of Exception

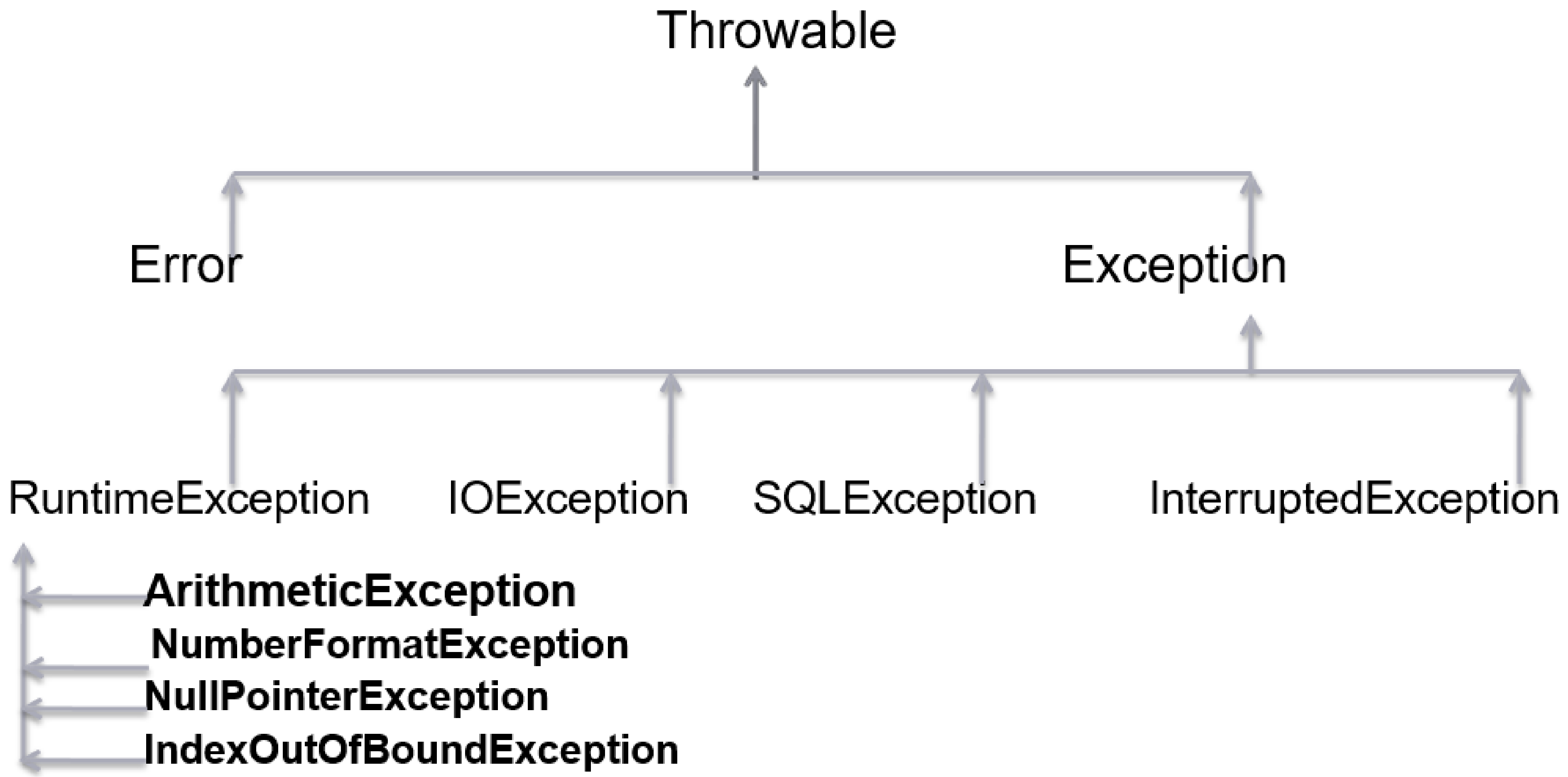

Throwable
Parent Class
Error
Out of programmer's hand Ex: Out of memory
Exception
Can be handled by the programmer
Checked Exception
Compiler force the programmer to handle Exception Ex: SQLException
Unchecked Exception
Run time Exception Ex: NullPointerException

Types of Exception

Exceptions are implemented in Java through a number of classes. The exception hierarchy is as follows:



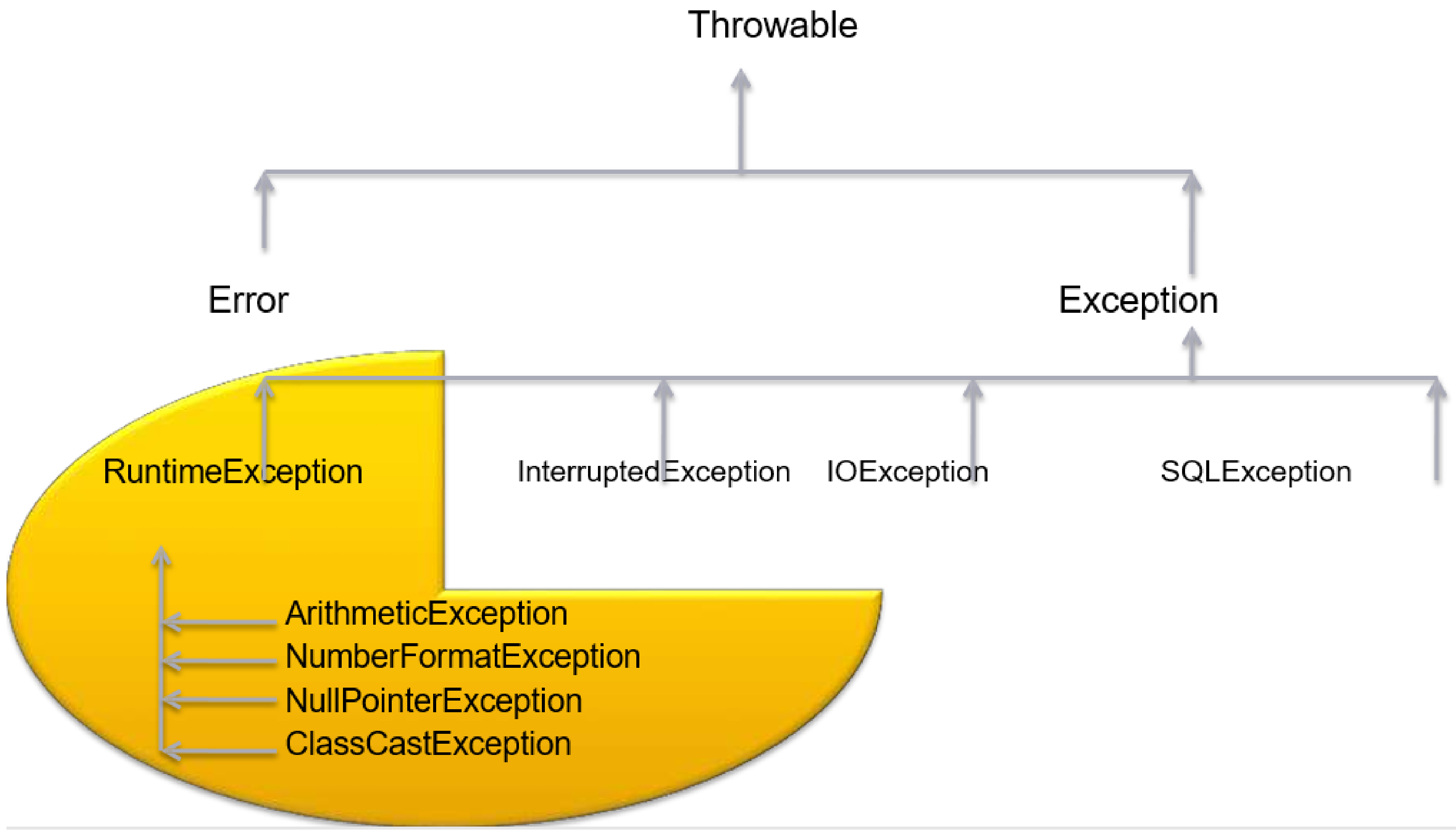
Checked and Unchecked Exceptions



Checked Exception (Contd.).

- A checked exception is an exception that usually happens due to user error or it is an error situation that cannot be foreseen by the programmer
- A checked exception must be handled using a try or catch or at least declared to be thrown using throws clause. Non compliance of this rule results in a compilation error
- Ex: FileNotFoundException If you try to open a file using FileInputStream fx = new FileInputStream(“A1.txt”);
- During execution, the system will throw a FileNotFoundException, if the file A1.txt is not located, which may be beyond the control of a programmer

Unchecked Exception



Unchecked Exceptions (Contd.).

- An unchecked exception is an exception, which could have been avoided by the programmer
- The class RuntimeException and all its subclasses are categorized as Unchecked Exceptions
- If there is any chance of an unchecked exception occurring in the code, it is ignored during compilation

Error

- Error is not considered as an Exception
- Errors are problems that arise beyond the control of the programmer or the user
- A programmer can rarely do anything about an Error that occurs during the execution of a program
- This is the precise reason Errors are typically ignored in the code
- Errors are also ignored by the compiler.   Ex : Stack Overflow

Error

```
public class Main {
        static void main(){
                main();
        }
        public static void main(String[] ar){
                main();
        }
}
```

```
Exception in thread "main" java.lang.StackOverflowError
        at vit.demo.Main.main(Main.java:7)
```

Checked VS Unchecked Exceptions

Checked Exception

The classes which directly inherit Throwable class except RuntimeException and Error are known as checked exceptions e.g. IOException, SQLException etc. Checked exceptions are checked at compile-time.

Unchecked Exception

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

Quiz

Listed below are some of the built in exception classes. List them in the table below as per their classification, whether they are checked exceptions or unchecked exceptions :

1)NullPointerException 2) ClassNotFoundException 3) IOException
4) InterruptedException 5)ArrayIndexOutOfBoundsException
6) NumberFormatException 7) ClassCastException 8)SQLException
9) IllegalAccessException 10) NegativeArraySizeException

Checked Exceptions	Unchecked Exceptions

Unchecked Exceptions

Name	Description
NullPointerException	Thrown when attempting to access an object with a reference variable whose current value is null
ArrayIndexOutOfBound	Thrown when attempting to access an array with an invalid index value
IllegalArgumentException.	Thrown when a method receives an argument formatted differently than the method expects.
IllegalStateException	Thrown when the state of the environment doesn’t match. e.g., using a Scanner that’s been closed.
NumberFormatException	Thrown when a method that converts a String to a number receives a String that it cannot convert.
ArithmaticException	Arithmetic error, such as divide-by-zero.

Checked Exceptions

Name	Description
IOException	While using file input/output stream related exception
SQLException.	While executing queries on database related to SQL syntax
DataAccessException	Exception related to accessing data/database
ClassNotFoundException	Thrown when the JVM can’t find a class it needs, because of a command-line error, a classpath issue, or a missing .class file
InstantiationException	Attempt to create an object of an abstract class or interface.

Example : A simple example

```
import java.util.*;
public class Main {
        public static void test(){
                try{
                        Scanner sin=new Scanner(System.in);
                        int x=sin.nextInt();

                        }
                        catch(InputMismatchException obj){
                                System.out.println("enter only integer");
                                test();
                        }
```

Example : A simple example

```
        }

        public static void main(String[] ar) {
                test();
        }
}
```

Summary

We have discussed about

- Introduction to Exception Handling
- Exception Handling Techniques
- Exception Handling Keywords
- Types of Exceptions