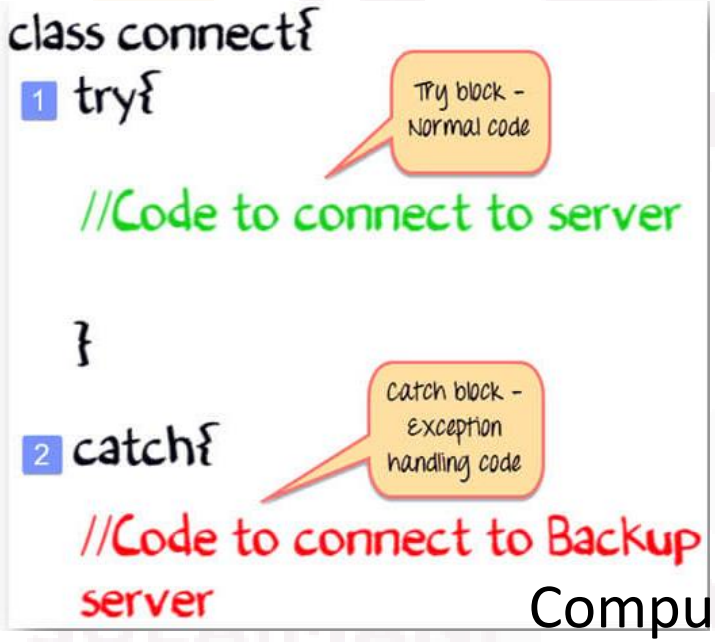
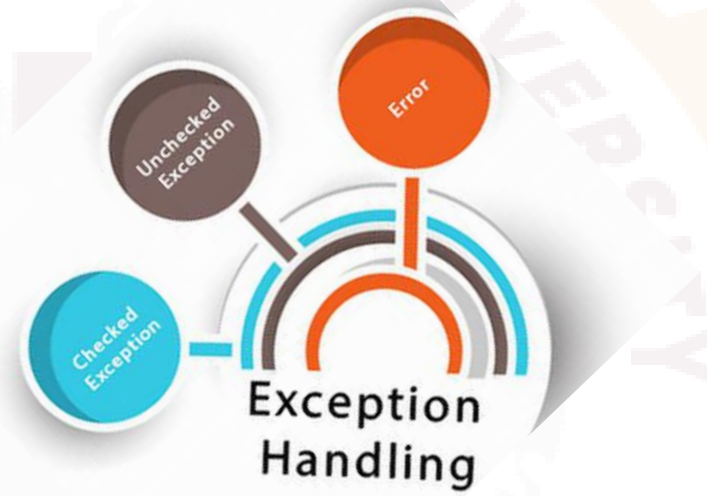
Java & Exception Handling





Computer Engineering

Yusramohammed@tiu.edu.iq

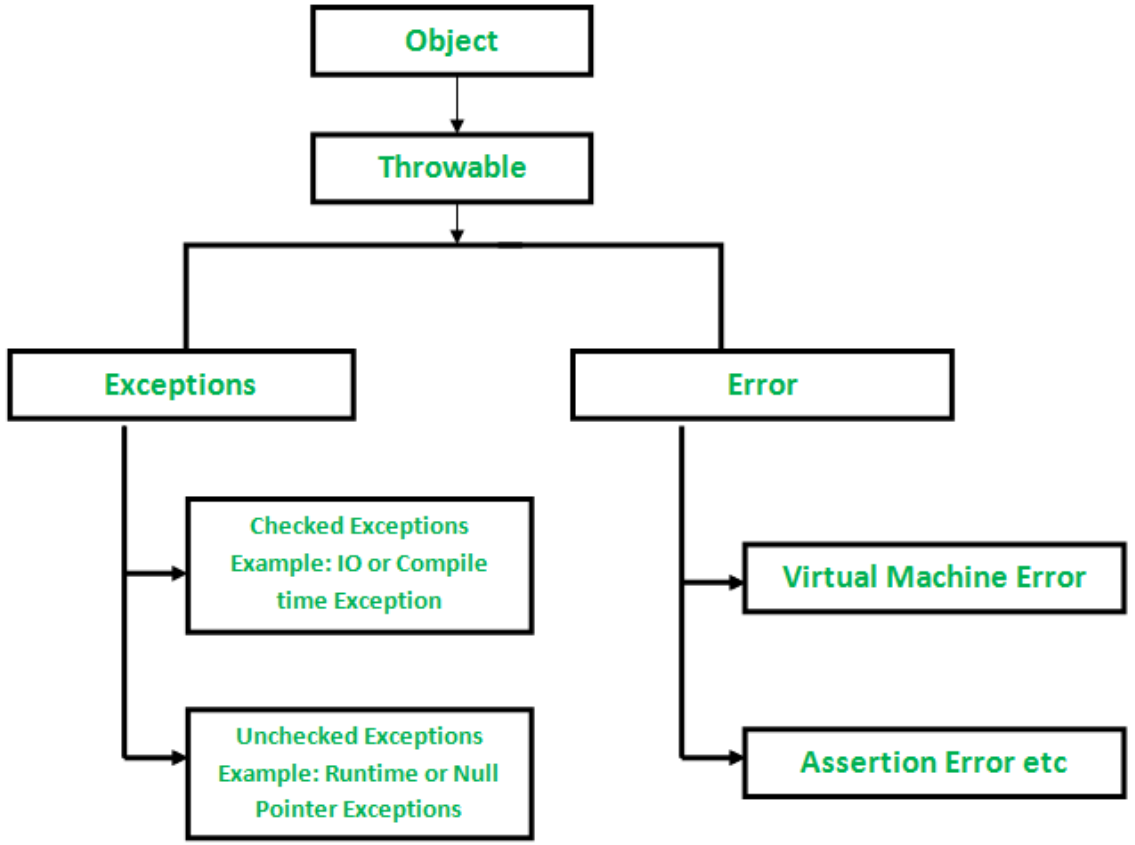
2022 - 2023

What is Exception?

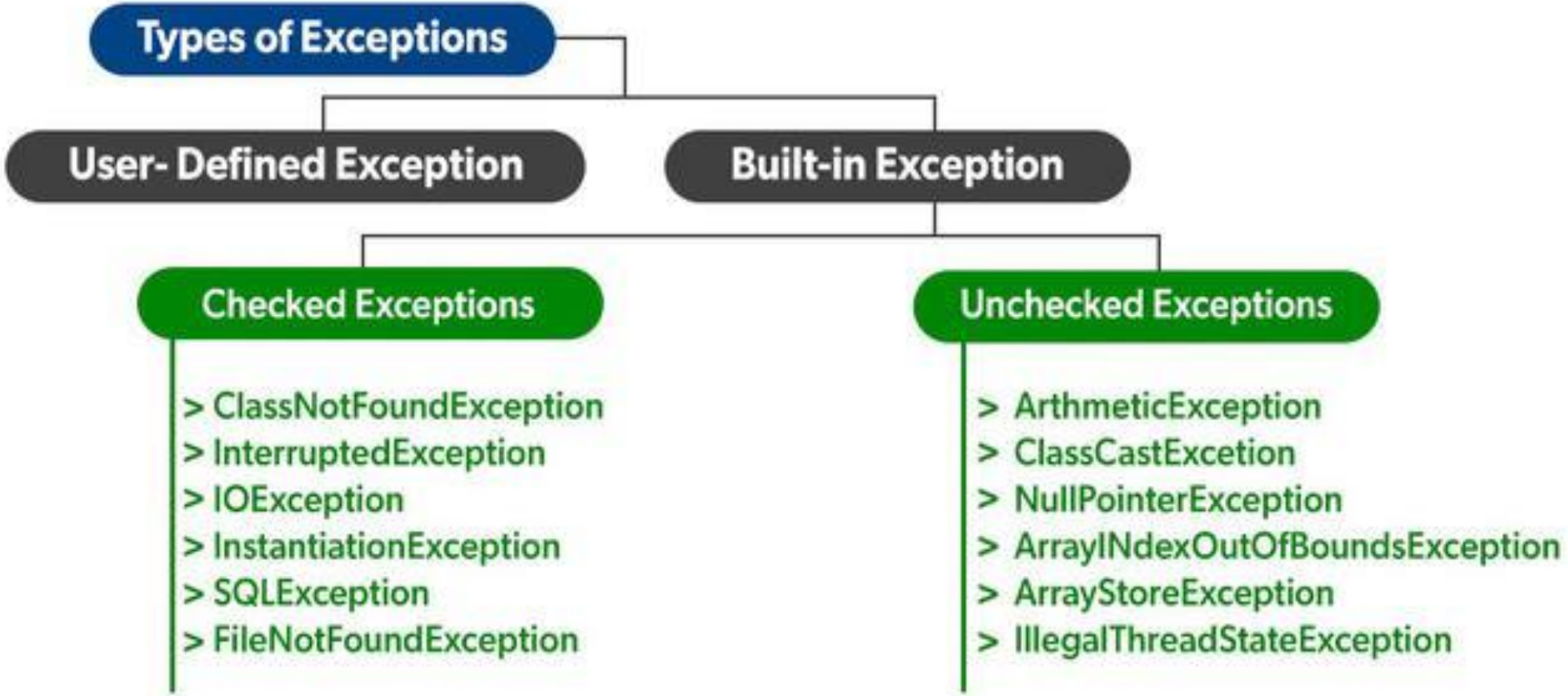
- When executing Java code, different errors can occur: coding errors made by the programmer, errors due to wrong input, or other unforeseeable things.
- When an error occurs, Java will normally stop and generate an error message. The technical term for this is: Java will throw an **exception** (throw an error).
- ***An exception*** (or exceptional event) is a problem that arises during the execution of a program.
- An exception can occur for many different reasons. Following are some scenarios where an exception occurs.
  - Invalid user input
  - Device failure
  - Loss of network connection
  - Physical limitations (out of disk memory)
  - Code errors
  - Opening an unavailable file
- Some of these exceptions are caused by user error, others by programmer error, and others by physical resources that have failed in some manner.

JVM Error vs. Exception

- **Error:** An Error indicates a serious problem that a reasonable application should not try to catch. conditions such as Java virtual machine (JVM) running out of memory, memory leaks, stack overflow errors, library incompatibility, infinite recursion, etc.
- **Exception:** Exception indicates conditions that a reasonable application might try to catch.



Types of Exceptions



What is Exception Handling?

- Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.
- *Arethmatic Exception Example:*

```
run:
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at TryCatch.ExceptionHandlingExample.main(ExceptionHandlingExample.java:15)
Java Result: 1
```

- *IOException Example:*

```
run:
Enter num1:bd
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Scanner.java:864)
        at java.util.Scanner.next(Scanner.java:1485)
        at java.util.Scanner.nextInt(Scanner.java:2117)
        at java.util.Scanner.nextInt(Scanner.java:2076)
        at TryCatch.ExceptionHandlingExample.main(ExceptionHandlingExample.java:22)
Java Result: 1
```

Catching Exceptions

- A method catches an exception using a combination of the **try** and **catch** keywords.
- Code within a try/catch block is referred to as protected code, and the syntax for using try/catch looks like →

Syntax

```
try {
  //  Block of code to try
}
catch(Exception e) {
  //  Block of code to handle errors
}
```

- The code which is seems to have exceptions is placed in the try block. When an exception occurs, that exception occurred is handled by catch block associated with it.
- Every try block should be immediately followed either by a catch block or finally block.
- A catch statement involves declaring the type of exception you are trying to catch.

Catching Exceptions

Syntax

```
try {
  //  Block of code to try
}
catch(Exception e) {
  //  Block of code to handle errors
}
```

```
public class ExceptionHandlingExample {
    public static void main(String[] args) {
        int x=0,y=1;
        try{
            int z=y/x;
            System.out.println("z:"+z);
        }
        catch(Exception e){
            System.out.println(e);
        }

    }}
```

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

Example without try-catch

- Without try catch if the program got an error at run time. It will stop running and shows the error for example:

```
public class ExceptionHandlingExample {
    public static void main(String[] args) {
        Scanner input=new Scanner (System.in);
        int array[]=new int [5];

            array[6]=3;
            System.out.println(array[6]);

        System.out.println("Enter number: ");
        int number =input.nextInt();
        array[0]=number;
        System.out.println("value of index: "+array[0]);
    }
}
```

- We have an array of 5 indices and we want to give a value of index 6

- The user enters a number and we give it to the index 0.

Example using try-catch

```
public class ExceptionHandlingExample {
    public static void main(String[] args) {
        Scanner input=new Scanner (System.in);
        int array[]=new int [5];
        try{
            array[6]=3;
            System.out.println(array[6]);
         }
        catch(Exception e){
            System.out.println(e);}
        try{
        System.out.println("Enter number: ");
        int number =input.nextInt();
        array[0]=number;
        System.out.println("value of index: "+array[0]);}
        catch(Exception e){
            System.out.println(e);}
    }
}
```

- Using a try-catch

- Using a try-catch

Output and see the differences

- Without using try-catch the program will see the error and stops running, the run of the program will be:

```
run:
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 6
        at TryCatch.ExceptionHandlingExample.main(ExceptionHandlingExample.java:19)
Java Result: 1
```

- Using try-catch: The program catch the error, shows the error in the run, and continue running. It will not stop.

```
run:
java.lang.ArrayIndexOutOfBoundsException: 6
Enter number:
5
value of index: 5
```

Common Scenario of Java Exceptions

- ***ArithmeticException:*** If we divide any number by zero, there occurs an ArithmeticException.

```
public class ExceptionHandlingExample {
    public static void main(String[] args) {
        int num=8;
        try{
            System.out.println(num/0);
         }
        catch(Exception e){
            System.out.println(e);
        }
    }
}
```

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

Common Scenario of Java Exceptions

- ***NumberFormatException:*** The wrong formatting of any value may occur NumberFormatException. Suppose I have a string variable that has characters, converting this variable into digit will occur NumberFormatException.

```
public class JavaApplication33 {
 public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.println("Enter something");
       String name=input.next();
       try{
           int b=Integer.parseInt(name);
           System.out.println(b);
       }
       catch(Exception e)
       {
           System.out.println(e);
       }
 }
```

Run

```
Enter something
yusra
java.lang.NumberFormatException: For input string: "yusra"
```

Common Scenario of Java Exceptions

- ***InputMismatchException:*** For example we want the user to enter an integer, and the user enters a different type of input:

```
public class ExceptionHandlingExample {
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        try{
        System.out.println("Enter num1:");
        int num=input.nextInt();}
        catch(Exception e){
            System.out.println(e);
        }
    }
}
```

```
Enter num1:
e
java.util.InputMismatchException
```

Sample Development

```
public class JavaApplication33 {
 public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        int cont=1;
        do{
            try{
                System.out.print("Enter first number: ");
                int num1=input.nextInt();
                System.out.print("Enter Second number: ");
                int num2=input.nextInt();
                int result=num1/num2;
                System.out.printf("Result= %d\n",result);
                System.out.println("press 1 to continue");
                cont=input.nextInt();
            }
            catch(Exception e)
            {
                System.out.println("Cant divide number by 0");
            }
        }while(cont==1);
 }
}
```

Sample Development (Run)

```
run:
Enter first number: 2
Enter Second number: 0
Cant divide number by 0
Enter first number: 32
Enter Second number: 0
Cant divide number by 0
Enter first number: 32
Enter Second number: 2
Result= 16
press 1 to continue
1
Enter first number: 2
Enter Second number: 1
Result= 2
press 1 to continue
2
BUILD SUCCESSFUL (total time: 34 seconds)
```

- Now remove the try-catch and run the program to see the difference.