Selenium Class 15 - Exception Handling in Java

Exception Handling

> An exception is an event that occurs during execution of a program when normal execution of the program is interrupted.

> Exception handling is a mechanism to handle run-time errors

In Automated Testing Selenium with Java we may face two types' exceptions

1) Global Exceptions (Ex: ArithmeticException)

2) Local Exceptions (Ex: NoSuchElementException)

Common Scenarios where Exception may occur

1) Scenario where Arithmetic Exception may occur

If we divide any number by zero then ArithmeticException occurs

Ex:

```
int a=10/0;
```

2) Scenario where NumberFormat Exception occurs

The wrong format of any value

```
String s1="abc";
String s2="123";

int a= Integer.parseInt(s1);
int a= Integer.parseInt(s2);

System.out.println(a); //NumberFormatException
```

3) Scenario where NullPointerException occurs

```
String s=null;
System.out.println(s.length()); //NullPointerException
```

4) Scenario where ArrayIndexOutOfBoundsException occurs

If we assign any value in the wrong index of an Array

Ex:

```
int [] a= new int[3];
.
.
.
int a[7]=70;
```

System.out.println(a[7]); //ArrayIndexOutOfBoundsException

Program with Exception (Run-time Error):

```
int a=10;
int b=0;

int result=a/b;
System.out.println(result);
System.out.println("Hello Java");
System.out.println("Hello Selenium");
```

Use try catch block

Syntax:

```
try {
Statements
.............
.............
.............
catch (ExceptionName name){
exception Handling code
}
```

Java Program with Exception handling code

```
int a=10;
int b=0;

try {
int result=a/b;
System.out.println(result);
}
catch (ArithmeticException e1){
System.out.println("Divided by Zero Error");
}
System.out.println("Hello Java");
System.out.println("Hello Selenium");
```

Handle multiple exceptions

```
int a=10;
int b=0;
int result;
int [] array1= new int[4];
try {
result=a/b;
System.out.println(result);
}
catch (ArithmeticException e1){
System.out.println("Divided by Zero Error");
```

```
}
System.out.println("Hello Selenium");
try {
array1[1]=100;
System.out.println(array1[10]);
}
catch (ArrayIndexOutOfBoundsException e2){
System.out.println("Array Out of Bound Error");
}
System.out.println("Hello Java World");
```

Types of Output in Java

The Output in Computer Programming is basically four types

1) Value based Result/Output (Fixed)
2) Boolean / Logical Result (true/false)
3) Constant based Result
4) Dynamic Result

1) Value based Result/Output (Fixed)

Arithmetic Operators and some predefined methods return this type of result

Example:

```
System.out.println(10+3);//13
System.out.println(7*4);//28
```

System.out.println(Math.max(10, 11));//11

2) Boolean / Logical Result (true/false)

Relational or Comparison Operators return Boolean Results, and some predefined methods also return this type of result

Example:

System.out.println(100 > 90);//true

System.out.println(100 < 90);//false

System.out.println("ABC".equals("abc"));//false

System.out.println("JAVA".equals("JAVA"));//true

3) Constant Based Result

Java Returns Constant based Result for 3-way comparison of numbers

Result Criteria:

if number1 = number2 then 0

if number1 > number2 then 1

if number1 < number2 then -1

Example:

int a=7;

String b="Abc";

```
Integer x=a;


System.out.println(x.compareTo(7));//0
System.out.println(x.compareTo(9)); //-1
System.out.println(x.compareTo(6));//1
```

4) Dynamic Result

i) Generating Random Numbers

ii) Return System Data Etc...

Example:

```
//Generate Random Numbers
System.out.println(Math.random());

//Return System Date
Date myDate = new Date();
System.out.println(myDate);
```
