VIT-AP UNIVERSITY

Course code : **CSE2005**

Course title : **Object Oriented Programming**

Exception Handling – Try- Catch Block

Objectives

This session will give the knowledge about

- Try-Catch Block
- Multiple catch block
- Nested try catch block

Try-Catch Block

- Any part of the code that can generate an error should be put in the try block
- Any error should be handled in the catch block defined by the catch clause
- This block also called the catch block, or the exception handler
- The corrective action to handle the exception should be put in the catch block

Try-Catch Block

```
public class Main {
        public static void main(String ar[]) {
                try {
                        int x = 30 / 0;
                }
                catch (ArithmeticException e) {
                        System.out.println("Division by zero.");
                }
                System.out.println("After catch statement.");
        }
}
```

Quiz

What will be the result, if we try to compile and execute the following code as javac Main welcome to exception

```
public class Main {
        public static void main(String ar[]) {
                for(int i=0;i<=ar.length;i++)
                        System.out.println(ar[i]);
        }
}
```

Multiple Catch Block

- A single block of code can raise more than one exception
- You can specify two or more **catch** clauses, each catching a different type of execution
- When an exception is thrown, each **catch** statement is inspected in order, and the first one whose type matches that of the exception is executed
- After one **catch** statement executes, the others are bypassed, and execution continues after the **try/catch** block

Multiple Catch Block

```
public class Main {
        public static void main(String ar[]) {
                try{
                        int len=ar.length;
                        System.out.println(34/len);
                        System.out.println(ar[5]);      }
                catch(ArithmeticException e){
                        System.out.println("/ by zero");
                }
                catch(ArrayIndexOutOfBoundsException e){
                        System.out.println("array index");
                } } }
```

Quiz

What will be the result, if we try to compile and execute the following code as java Main 23

```
public class Main {
        public static void main(String ar[]) {
                try {
                        int i= Integer.parseInt(ar[0]);
                        System.out.println(i);
                }
                System.out.println("this is catch block");
                catch(NumberFormatException e){
                        System.out.println(e);
                } } }
```

Multiple Catch Block

- When you use multiple catch statements, it is important to remember that exception subclasses must come before any of their exception super classes

- This is because a catch statement that uses a superclass will catch exceptions of that type as well as exceptions of its subclasses

- Thus, a subclass exception would never be reached if it came after its superclass that manifests as an **unreachable code error**

Quiz

```
public class Main {
        public static void main(String args[]) {
                try {
                        int i = Integer.parseInt(args[0]);
                        System.out.println(i);
                } catch (RuntimeException e) {
                        System.out.println(e);
                } catch (NumberFormatException e) {
                        System.out.println(e);
                }
        }
}
```

Nested try Statements

- The **try** statement can be nested
- If an inner **try** statement does not have a **catch** handler for a particular exception, the outer block's catch handler will handle the exception
- This continues until one of the **catch** statement succeeds, or until all of the nested **try** statements are exhausted
- If no catch statement matches, then the Java runtime system will handle the exception

Summary

We have discussed about

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