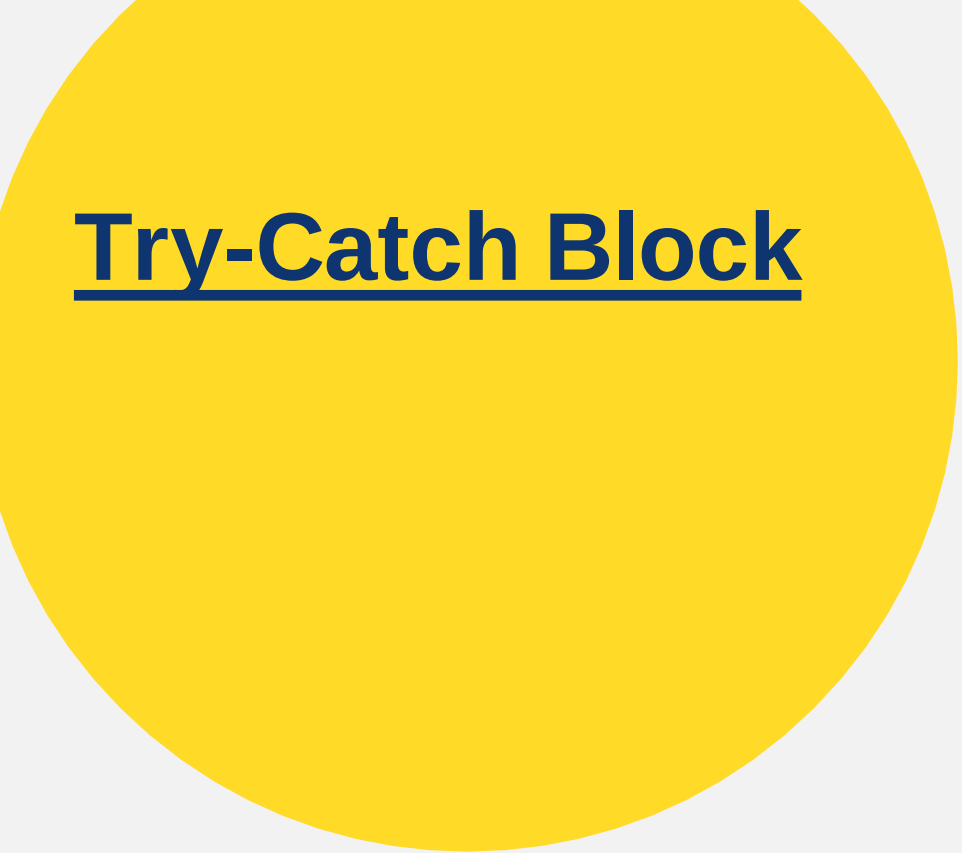
Exception Handling – Try-Catch Block

Agenda

1. Try-Catch Block
2. Multiple Catch Block
3. Nested Try Block

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 c**atch** block defined by the **catch** clause
- This block is also called the **catch block**, or the **exception handler**
- The corrective action to handle the exception should be put in the **catch** block

How to Handle exceptions?

```
class ExceptDemo{
  public static void main(String args[]){
    int x, a;
    try{
        x = 0;
        a = 22 / x;
        System.out.println("This will be bypassed.");
    }
    catch (ArithmeticException e){
        System.out.println("Division by zero.");
    }
    System.out.println("After catch statement.");
  }
}
```

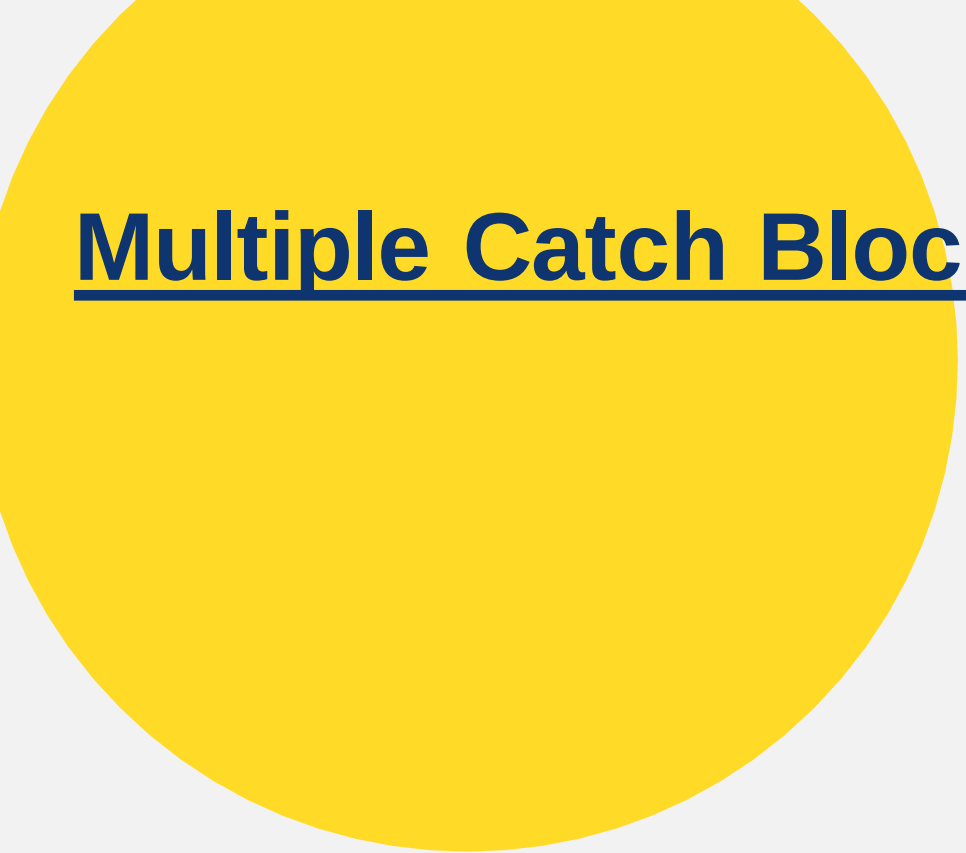
Quiz

- What will be the result, if we try to compile and execute the following code as java Ex1 Wipro Bangalore

```
Class Ex1 {
    public static void main(String[] xyz){
        for(int i=0;i<=args.length;i++)
          System.out.println(args[i]);
    }
  }
```

It will compile successfully but will throw exception during runtime!
Why this exception is thrown?

Multiple Catch Block



Multiple Catch Statements

- 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 Statements (Contd.).

```
class MultiCatch{
  public static void main(String args[]){
    try{
        int l = args.length;
        System.out.println("l = " +l);
        int b = 42 / l;
        int arr[] = { 1 };
        arr[22] = 99;
    }
    catch(ArithmeticException e){
        System.out.println("Divide by 0: "+ e);
    }
```

Multiple Catch Statements (Contd.).

```
    catch(ArrayIndexOutOfBoundsException e){
        System.out.println("Array index oob: "+e);
    }
    System.out.println("After try/catch blocks.");
  }
}
```

Quiz

- What will be the result, if we try to compile and execute the following code as java Ex2 100

```
class Ex2 {
    public static void main(String[] args) {
      try {
        int i= Integer.parseInt(args[0]);
        System.out.println(i);
      }
      System.out.println("Wipro");
      catch(NumberFormatException e) {
        System.out.println(e);
      }
   }
}
```

It will throw compilation Error

Multiple Catch Statements involving Exception Superclasses & Subclasses

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

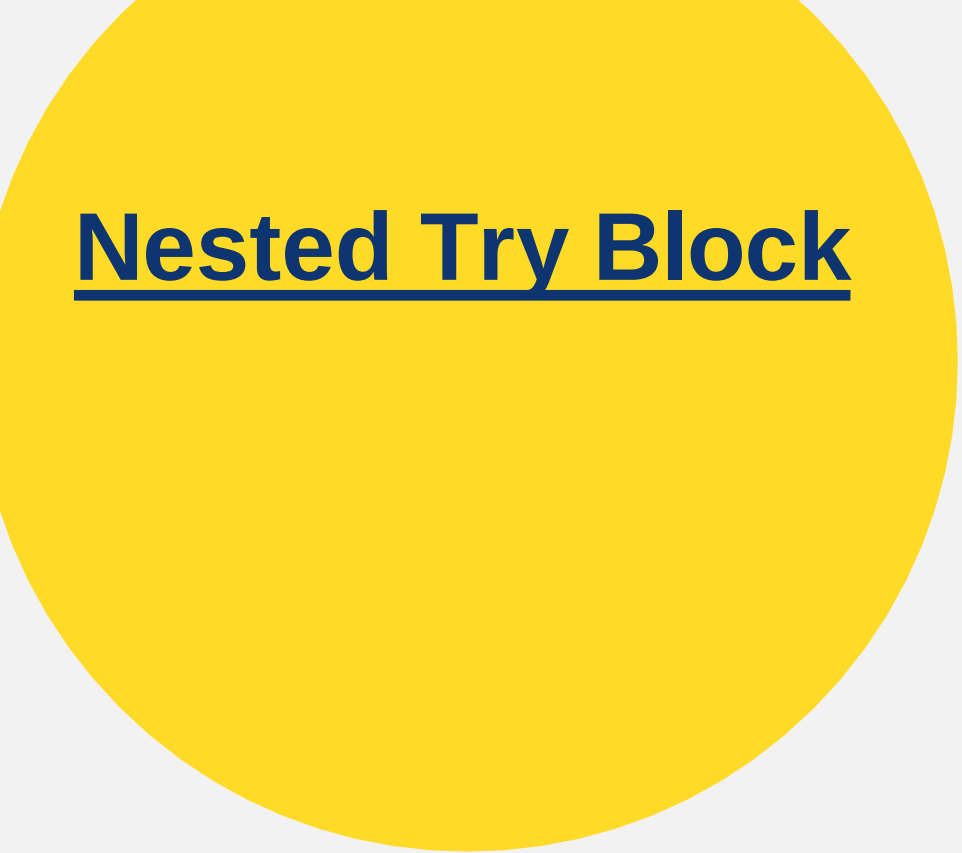
Quiz

- What will be the result, if we try to compile and execute the following code as java Ex2 100

```
class Ex2 {
    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);}
   }
}
```

It will throw compilation Error

Nested Try Block



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

Syntax

```
try
{
     statement 1;
     statement 2;
     try
     {
          statement 1;
          statement 2;
     }
     catch(Exception e)
     {
     }
}
catch(Exception e)
{
}
```

Example for nested try

```
class Nested_Try{
 public static void main(String args[]){
  try{
          try{
                System.out.println("Arithmetic Division");
             int b =39/0;
          } catch(ArithmeticException e){
                System.out.println(e);
           }
        try{
         int a[]=new int[5];
        System.out.println("Accessing Array Elements");
         a[5]=4;
      } catch(ArrayIndexOutOfBoundsException e)    {
           System.out.println(e);
      }
      System.out.println ("Inside Parent try");
   } catch(Exception e) {
                              System.out.println("Exception caught");
      }
System.out.println("Outside Parent try");
 }
 }
```

Quiz

1. Debug the code

```
public class Tester {
public static void main(String[] args) {
try{
     try{System.out.println(12/0); }
}
catch(Exception e){
}
}}
```

2. Debug the code

```
public class Tester {
public static void main(String[] args) {
     try {
          System.out.println("A");
     }
     catch (Exception e)
          {System.out.println("B");          }
     catch (ArithmeticException a)
     {System.out.println("C"); }
}}
```

Summary

In this session, you were able to :

- Learn about try-catch block
- Learn about multiple catch block
- Learn about nested try block

Assignment

Thank You