Course Name: A Level (2nd Sem) **Subject: JAVA**
Topic :try-catch Examples(contd) **Date: 09-04-20**

Try-catch example-2:

```
class TryCatch2 {
 public static void main(String[] args) {
    try
    {
    int data=20/0;                    // exception   prone code
    }

    catch(Exception e)
    {
       System.out.println(e);
    }
    System.out.println("this portion will always be printed");
  }

}
```

Explaination: Parent class exception(Exception) can also be used in place of specific exception,but not any other exception can be used

Try-catch example-3:

```
class TryCatch3 {
public static void main(String[] args) {
    try
    {
    int a=20/0; // exception prone code
 System.out.println("other statement"); //---in case of exception this line will not be executed
    }
        catch(ArithmeticException e)
    {
       System.out.println(e);
    }

  }

}
```

Explaination: when exception occurs the other statement will not be executed and the relevant exception will be thrown.

Try-catch example-4:

```
class TryCatch4 {
        public static void main(String[] args) {
                try
            {
            int a=20/0; // exception
          }
                  catch(Exception e)
            {
               int b=10/0; // exception
          }
           System.out.println("other statement");
           }
       }
```

Explaination: In this example we have seen that exception can also be enclosed in catch block as well, but will eventually terminate because it does not contain exception handling code.It is also not a good practice to put exception prone code in catch block.

Try-catch example-5:

```
class TryCatch5 {
        public static void main(String[] args) {
            try
            {
            int a=20/2; // here no exception
         System.out.println("other statement"); //---so this line will  be executed
            }
                catch(ArithmeticException e)
            {
               System.out.println(e);
            }
                }
           }
```

Explaination: In this example, since there is no exception, so the try block will be executed normally and flow will not go to catch block and come outside try-catch block.The catch block is executed only when exception occurs.

multiple catch blocks:

If we have to perform various tasks at the occurrence of various exceptions, we can use the multi-catch block. A try block can be followed by one or more catch blocks. Each catch block must contain a different exception handler.

A generic catch block can handle all the exceptions. Whether it is ArrayIndexOutOfBoundsException or ArithmeticException or NullPointerException or any other type of exception, this handles all of them.

Example 1 of multiple catch blocks:

```
class Multicatch1{
  public static void main(String args[]){
    try{
        int a[]=new int[5];
        a[4]=20/0;
        System.out.println("First  statement in try block");
    }
    catch(ArithmeticException e){
      System.out.println(" ArithmeticException occurs");
    }
    catch(ArrayIndexOutOfBoundsException e){
      System.out.println("ArrayIndexOutOfBoundsException occurs");
    }
    catch(Exception e){
      System.out.println(" Some Other exception occurs");
    }
  System.out.println("Out of try-catch block");
 }
}
```

Note:

1. At a time only one exception occurs and at a time only one catch block is executed.
2. All catch blocks must be ordered from most specific to most general, i.e. catch for ArithmeticException must come before catch for Exception.

Exercise:

1. What is the benefit of multiple catch blocks.
2. What will happen if we declare generic exception in first catch block?