CREATING OWN EXCEPTION SUB CLASSES

BY
M. BABY ANUSHA,
ASST.PROF IN CSE DEPT.,
RGUKT,NUZVID

INTRODUCTION :

- Java provides rich set of built-in exception classes like: ArithmeticException, IOException, NullPointerException etc. all are available in the java.lang package and used in exception handling.
- Apart from these classes, Java allows us to create our own exception class to provide own exception implementation.
- These type of exceptions are called user-defined exceptions or **custom** exceptions.

INTRODUCTION :

- In Java, we can create our own exceptions that are derived classes of the Exception class.
- Creating our own Exception is known as custom exception or user-defined exception.
- Basically, Java custom exceptions are used to customize the exception according to user need.

WHY USE CUSTOM EXCEPTIONS?

- Java exceptions cover almost all the general type of exceptions that may occur in the programming. However, we sometimes need to create custom exceptions.

  Following are few of the reasons to use custom exceptions:

1. To catch and provide specific treatment to a subset of existing Java exceptions.
2. Business logic exceptions: These are the exceptions related to business logic and workflow. It is useful for the application users or the developers to understand the exact problem.

- In order to create custom exception, we need to extend Exception class that belongs to java.lang package.

WHY USE CUSTOM EXCEPTIONS?

- Consider the following example, where we create a custom exception named WrongFileNameException:

```
public class WrongFileNameException extends Exception
{
   public WrongFileNameException(String errorMessage)
   {
        super(errorMessage);
   }
}
```

JAVA CUSTOM EXCEPTION :EXAMPLE 1

- Let's see a simple example of Java custom exception. In the following code, constructor of InvalidAgeException takes a string as an argument.
- This string is passed to constructor of parent class Exception using the super() method. Also the constructor of Exception class can be called without using a parameter and calling super() method is not mandatory.

JAVA CUSTOM EXCEPTION :

- Consider the example 1 in which InvalidAgeException class extends the Exception class.
- Using the custom exception, we can have your own exception and message.
- Here, we have passed a string to the constructor of superclass i.e.
- Exception class that can be obtained using getMessage() method on the object we have created.

JAVA CUSTOM EXCEPTION :EXAMPLE 1

```
 class InvalidAgeException  extends Exception
{
    public InvalidAgeException (String str)
    {
        // calling the constructor of parent Exceptio
  n
        super(str);
    }
}
```

JAVA CUSTOM EXCEPTION :EXAMPLE 1

```
public class  CustomExceptionDemo
{
      static void validate (int age) throws InvalidAgeException{
       if(age < 18)
   {
  throw new InvalidAgeException("age is not valid to vote");
    }
       else
       {
        System.out.println("welcome to vote");
        }
     }
```

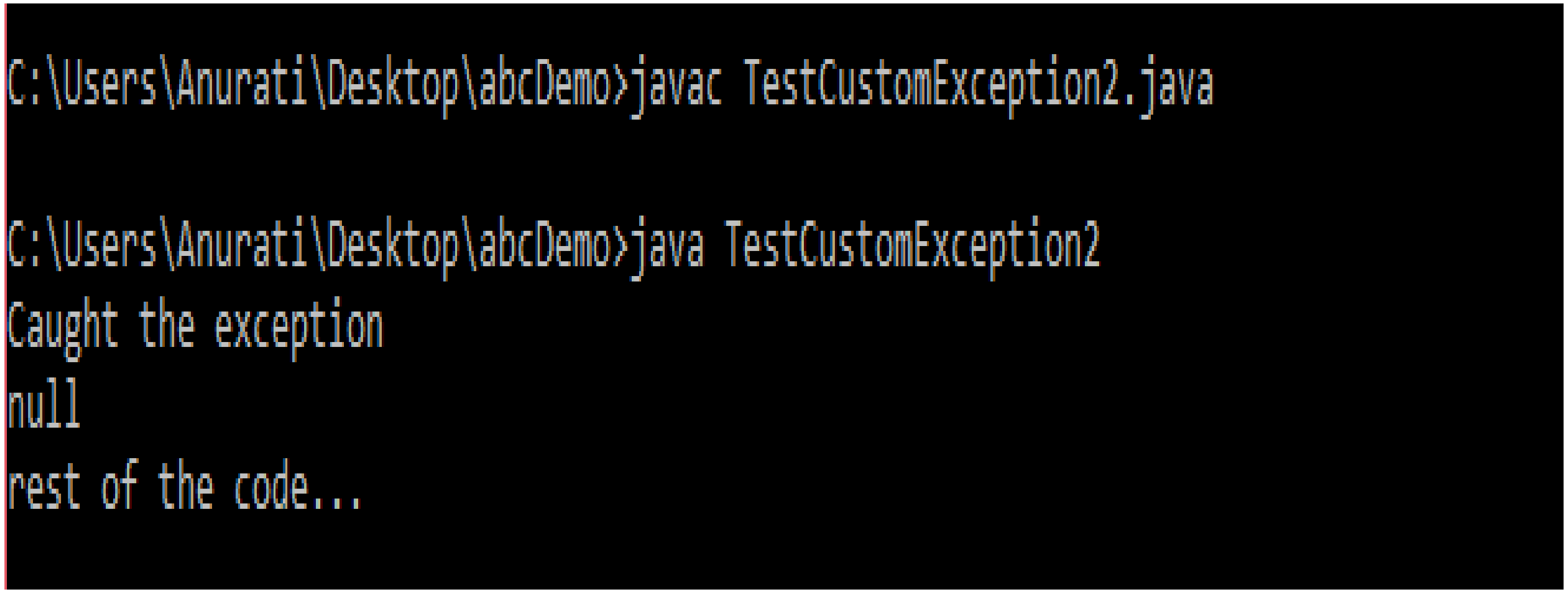
JAVA CUSTOM EXCEPTION :EXAMPLE 1

```
    public static void main(String args[])
    {
        try
        {
                validate(13);
        }
        catch (InvalidAgeException ex)
        {
            System.out.println("Caught the exception");

            System.out.println("Exception occured: " + ex);
        }

        System.out.println("rest of the code...");
    }
}
```

JAVA CUSTOM EXCEPTION :EXAMPLE 1



```
C:\Users\Anurati\Desktop\abcDemo>javac TestCustomException2.java

C:\Users\Anurati\Desktop\abcDemo>java TestCustomException2
Caught the exception
null
rest of the code...
```

Thank you!
shutterstock.com · 567687052