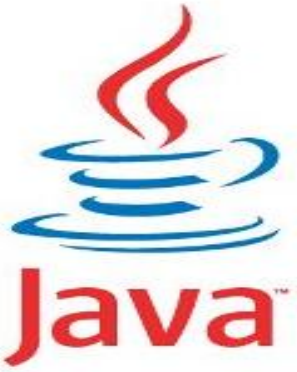

JAVA SE

(CORE JAVA)

LECTURE-29

Today's Agenda

- Creating programmer defined exception\Custom exception
- Using the keyword "finally"
- Multi catch feature of Java.

Programmer defined\ Customized Exceptions

- Many times, in some situations a programmer might not find any predefined exception class to be used with throw. For example, in case of a banking application the method withdraw has some minimum limit like 500 else an exception will generate. In this case there is no predefined java's exception class.

- So, Java advises us to design our own exception classes. Such classes are called customized exception class.

- To create an exception class we need to follow some steps, which are as follows :-

Programmer defined\ Customized Exceptions

1. Inherit or extend any of the predefined exception class in our own exception class.

2. Provide a parameterized constructor so that exception message can be set and passed on to parent class's constructor.

Example


```
import java.util.*;
class InvalidNumeratorException extends Exception
{
 public InvalidNumeratorException(String msg)
 {
 super(msg);
 }
}
class Test
{
 public static void main(String [] args)
 {
 Scanner kb=new Scanner(System.in);
 System.out.println("Enter two numbers");
```

Example


```
try
{
int a=kb.nextInt();
int b=kb.nextInt();
if(a<=0)
{
throw new InvalidNumeratorException("Numerator should be
    positive");
}
int c=a/b;
System.out.println("Division is "+c);
}
```

Example


```
catch(ArithmeticException ex)
{
System.out.println(ex.getMessage());
}
catch(InvalidNumeratorException ex)
{
System.out.println(ex.getMessage());
}
catch(InputMismatchException ex)
{
System.out.println("Please input digits only");
}
}
}
```

Programmer defined\ Customized Exceptions

- The customized exceptions become checked exception if we inherit the base class Exception.

- Since, only RuntimeException and its child classes are unchecked in nature, so programmer has to specifically inherit anyone of those to create and unchecked excception class.

Using the keyword "finally"

- There are certain statements in our program whose execution is so crucial that before our program gets terminated, these statements must be executed.
- Example, if we have opened a file or any database connection and before the program completes its execution the file or database connection should be closed.
- In such cases java suggests us to write such statements in a block whose execution is guaranteed by java and such blocks are created using the keyword finally.

Using the keyword "finally"

- If no exception occurs finally block is executed.
- If an exception occurs inside the try block and its catch has been defined, in such case also finally is executed.
- Even if no catch block is used then also the finally block is executed.
- Moreover, a try block is required for an finally block. If an exception occurs outside try block in that case finally block is not executed and also when the method System.exit() is used.

Syntax


```
try
{
---
---
}
catch( ---)
{
----
}
finally
{
----
----
}
```

```
try
{
---
---
}
finally
{
----
----
}
```

Example


```
import java.util.*;
class TestFinally
{
 public static void main(String [] args)
 {
 Scanner kb=new Scanner(System.in);
 System.out.println("Enter two numbers");
 try
 {
 int a=kb.nextInt();
 int b=kb.nextInt();
 int c=a/b;
 System.out.println("Division is "+c);
 }
```

Example


```
catch(ArithmeticException ex)
{
System.out.println("Denominator should not be 0");
}
finally
{
System.out.println("Thank you! Have a good day");
}
}
```

Multi catch feature

- We can use a single catch to handle multiple exceptions.
- This feature was introduced in java from 7th version.
- Syntax :-

```
catch(ArithmeticException | InvalidNumeratorException ex)
{
System.out.println(ex.getMessage);
}
```

End Of Lecture 29

Thank You

For any queries mail us @: scalive4u@gmail.com
Call us @ : 0755-4271659, 7879165533

Agenda for Next Lecture:

1. **String Handling**