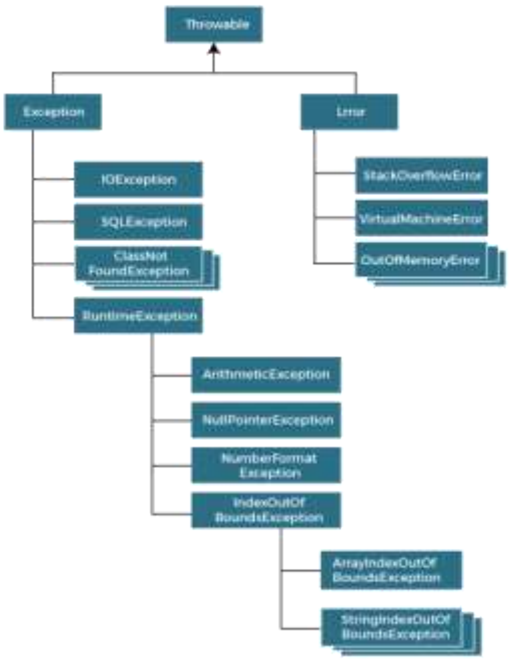
UNIT 5
Exception Handling, Applets

Exception Handling

- The Exception Handling in Java is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained.
- Exception is an abnormal condition.
- An exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

Hierarchy of Java Exception classes

The java.lang.Throwable class is the root class of Java Exception hierarchy inherited by two subclasses: Exception and Error. The hierarchy of Java Exception classes is given below:



Types of Java Exceptions

There are mainly two types of exceptions: checked and unchecked. An error is considered as the unchecked exception. However, according to Oracle, there are three types of exceptions namely:

- Checked Exception
- Unchecked Exception
- Error

Checked Exception

- The classes that directly inherit the Throwable class except RuntimeException and Error are known as checked exceptions. For example, IOException, SQLException, etc. Checked exceptions are checked at compile-time.

Unchecked Exception

- The classes that inherit the RuntimeException are known as unchecked exceptions. For example, ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException, etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

Error : Error is irrecoverable.

some scenarios where unchecked exceptions may occur

ArithmeticException

If we divide any number by zero, there occurs an ArithmeticException.

Ex : int a=50/0;//ArithmeticException

NullPointerException

If we have a null value in any variable, performing any operation on the variable throws a NullPointerException

Ex : String s=null;

System.out.println(s.length());//NullPointerException

NumberFormatException

If the formatting of any variable or number is mismatched, it may result into NumberFormatException. Suppose we have a string variable that has characters; converting this variable into digit will cause NumberFormatException.

Ex : String s="abc";

int i=Integer.parseInt(s);//NumberFormatException

ArrayIndexOutOfBoundsException

When an array exceeds to it's size, the ArrayIndexOutOfBoundsException occurs. there may be other reasons to occur ArrayIndexOutOfBoundsException. Consider the following statements.

Ex : int a[]=new int[5];

a[10]=50; //ArrayIndexOutOfBoundsException

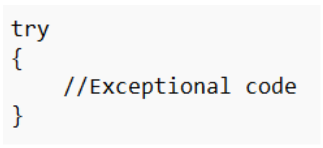
Java Exception Keywords

Java provides five keywords that are used to handle the exception.

- **Try**

The "try" keyword is used to specify a block where we should place
an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally.

Syntax:

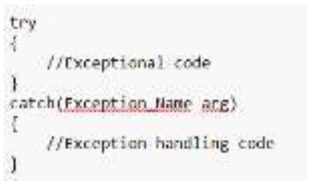


```
try
{
    //Exceptional code
}
```

- **catch**

The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later. We can use multiple catch blocks with a single try block if multiple exceptions raised.

Syntax:

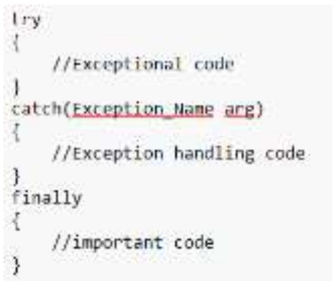


```
try
{
    //Exceptional code
}
catch(Exception_Name arg)
{
    //Exception handling code
}
```

- **finally**

The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not.

Syntax:



```
try
{
    //Exceptional code
}
catch(Exception_Name arg)
{
    //Exception handling code
}
finally
{
    //important code
}
```

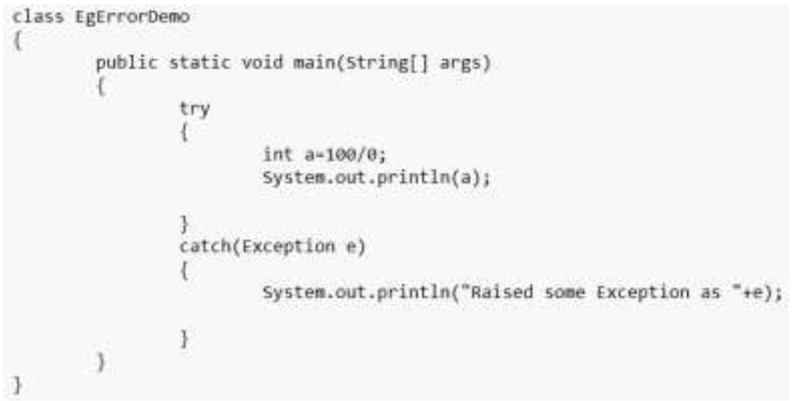
- **throw**

The "throw" keyword is used to throw an exception.

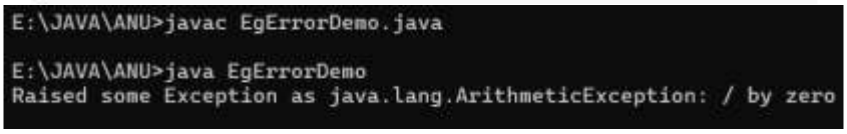
- **throws**

The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signature.

Java program to illustrate Exception Handling

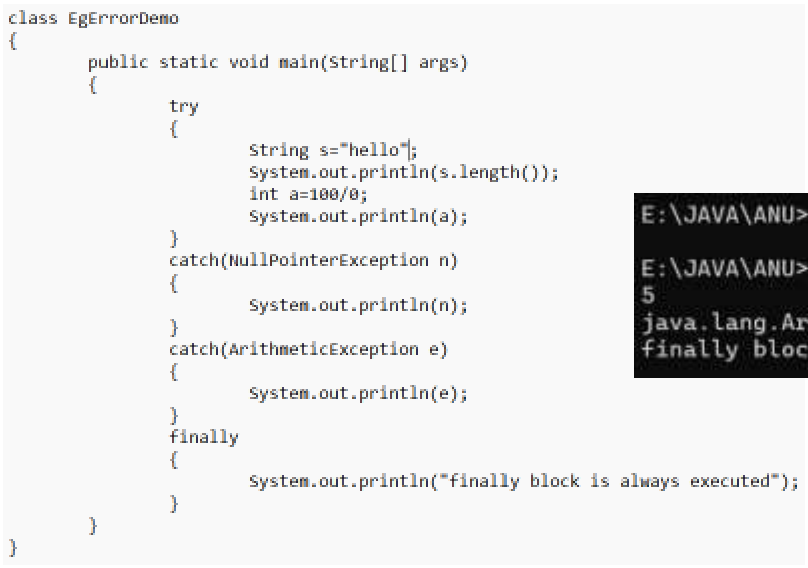


```
class EgErrorDemo
{
	public static void main(String[] args)
	{
		try
		{
			int a=100/0;
			System.out.println(a);

		}
		catch(Exception e)
		{
			System.out.println("Raised some Exception as "+e);

		}
	}
}
```

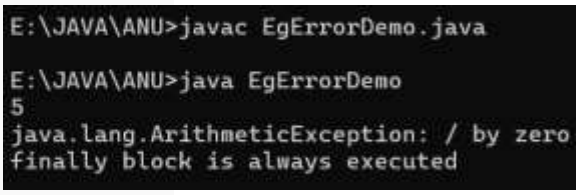


```
E:\JAVA\ANU>javac EgErrorDemo.java

E:\JAVA\ANU>java EgErrorDemo
Raised some Exception as java.lang.ArithmeticException: / by zero
```

Java program to illustrate Exception Handling using multiple catch blocks and finally block

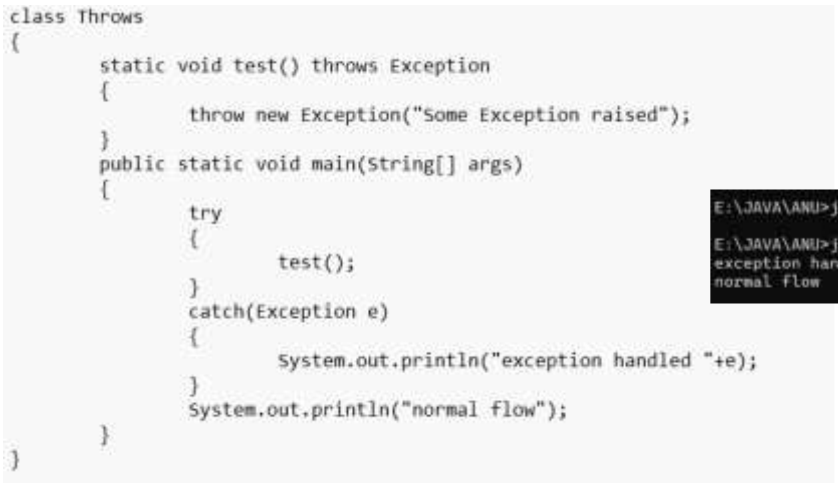


```
class EgErrorDemo
{
	public static void main(String[] args)
	{
		try
		{
			String s="hello";
			System.out.println(s.length());
			int a=100/0;
			System.out.println(a);
		}
		catch(NullPointerException n)
		{
			System.out.println(n);
		}
		catch(ArithmeticException e)
		{
			System.out.println(e);
		}
		finally
		{
			System.out.println("finally block is always executed");
		}
	}
}
```

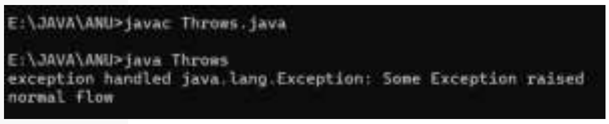


```
E:\JAVA\ANU>javac EgErrorDemo.java

E:\JAVA\ANU>java EgErrorDemo
5
java.lang.ArithmeticException: / by zero
finally block is always executed
```

Java program on handling Exceptions that are raised from methods using throws keyword



```
class Throws
{
	static void test() throws Exception
	{
		throw new Exception("Some Exception raised");
	}
	public static void main(String[] args)
	{
		try
		{
			test();
		}
		catch(Exception e)
		{
			System.out.println("exception handled "+e);
		}
		System.out.println("normal flow");
	}
}
```



```
E:\JAVA\ANU>javac Throws.java

E:\JAVA\ANU>java Throws
exception handled java.lang.Exception: Some Exception raised
normal flow
```

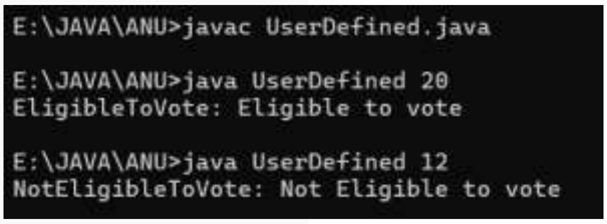
Difference between throw and throws in Java

throw	throws
Java throw keyword is used throw an exception explicitly in the code, inside the function or the block of code.	Java throws keyword is used in the method signature to declare an exception which might be thrown by the function while the execution of the code.
Type of exception Using throw keyword, we can only propagate unchecked exception i.e., the checked exception cannot be propagated using throw only.	Using throws keyword, we can declare both checked and unchecked exceptions. However, the throws keyword can be used to propagate checked exceptions only.
The throw keyword is followed by an instance of Exception to be thrown.	The throws keyword is followed by class names of Exceptions to be thrown.
throw is used within the method.	throws is used with the method signature.
We are allowed to throw only one exception at a time i.e. we cannot throw multiple exceptions.	We can declare multiple exceptions using throws keyword that can be thrown by the method. For example, main() throws IOException, SQLException.

User-defined Exceptions

Java provides us the facility to create our own exceptions which are basically derived classes of Exception. Creating our own Exception is known as a custom exception or user-defined exception. Basically, Java custom exceptions are used to customize the exception according to user needs. In simple words, we can say that a User-Defined Exception or custom exception is creating your own exception class and throwing that exception using the ‘throw’ keyword.

```
class EligibleToVote extends Exception
{
	EligibleToVote(String msg)
	{
		super(msg);
	}
}
class NotEligibleToVote extends Exception
{
	NotEligibleToVote(String msg)
	{
		super(msg);
	}
}
class UserDefined
{
	public static void main(String[] args)
	{
		try
		{
			int age=Integer.parseInt(args[0]);
			if(age<18)
			{
				throw new NotEligibleToVote("Not Eligible to vote");
			}
			else
			{
				throw new EligibleToVote("Eligible to vote");
			}

		}
		catch(Exception e)
		{
			System.out.println(e);
		}
	}
}
```



```
E:\JAVA\ANU>javac UserDefined.java

E:\JAVA\ANU>java UserDefined 20
EligibleToVote: Eligible to vote

E:\JAVA\ANU>java UserDefined 12
NotEligibleToVote: Not Eligible to vote
```

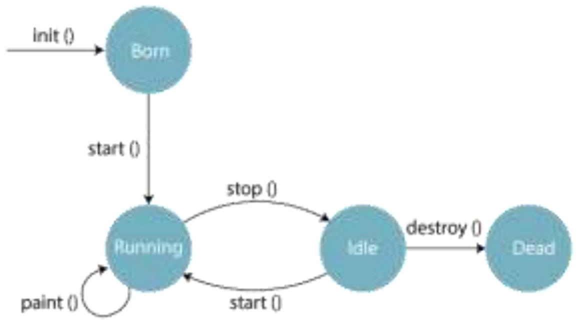
Applets

- An applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. An applet is embedded in an HTML page using the APPLET or OBJECT tag and hosted on a web server.
- Applets are used to make the website more dynamic and entertaining.

Important points :

- ✓ All applets are sub-classes (either directly or indirectly) of java.applet.Applet class.
- ✓ Applets are not stand-alone programs. Instead, they run within either a web browser or an applet viewer. JDK provides a standard applet viewer tool called applet viewer.
- ✓ In general, execution of an applet does not begin at main() method.
- ✓ Output of an applet window is not performed by System.out.println(). Rather it is handled with various AWT methods, such as drawString().

Applet Lifecycle



The life cycle of a Java applet is the series of events that occur when a web browser loads and executes an applet. The life cycle of a Java applet has four stages, which are triggered by the browser or applet viewer

1. **init( )** : The init( ) method is the first method to be called. This is where you should initialize variables. This method is called only once during the run time of your applet.
2. **start( )** : The start( ) method is called after init( ). It is also called to restart an applet after it has been stopped. Note that init( ) is called once i.e. when the first time an applet is loaded whereas start( ) is called each time an applet's HTML document is displayed onscreen. So, if a user leaves a web page and comes back, the applet resumes execution at start( ).
3. **paint( )** : The paint( ) method is called each time an AWT-based applet's output must be redrawn. This situation can occur for several reasons. For example, the window in which the applet is running may be overwritten by another window and then uncovered. Or the applet window may be minimized and then restored.
   paint( ) is also called when the applet begins execution. Whatever the cause, whenever the applet must redraw its output, paint( ) is called.
   The paint( ) method has one parameter of type Graphics. This parameter will contain the graphics context, which describes the graphics environment in which the applet is running. This context is used whenever output to the applet is required.
4. **stop( )** : The stop( ) method is called when a web browser leaves the HTML document containing the applet—when it goes to another page, for example. When stop( ) is called, the applet is probably running. You should use stop( ) to suspend threads that don't need to run when the applet is not visible. You can restart them when start( ) is called if the user returns to the page.

5. **destroy( ) :** The destroy( ) method is called when the environment determines that your applet needs to be removed completely from memory. At this point, you should free up any resources the applet may be using. The stop( ) method is always called before destroy( ).

Different Ways to Run Applet in Java

There are two standard methods for running an applet:

- ✓ By using Web Browser
- ✓ By using Applet Viewer

Executing the applet within a Java-compatible Web browser:

Suppose we have a **DemoApplet.java** file in which we have our java code.

```
import java.awt.*;
import java.applet.*;
class DemoApplet extends Applet
{
        public void init()
        {
                setBackground(Color.blue);
        }
        public void paint(Graphics g)
        {
                g.drawString("welcome to Applet",100,100);
        }
}
```

Compiling: javac DemoApplet.java

Create an Html file and embed the Applet tag in the HTML file.

To run an applet in a web browser, we must create an HTML text file with a tag that loads the applet. For this, we may use the APPLET or OBJECT tags.

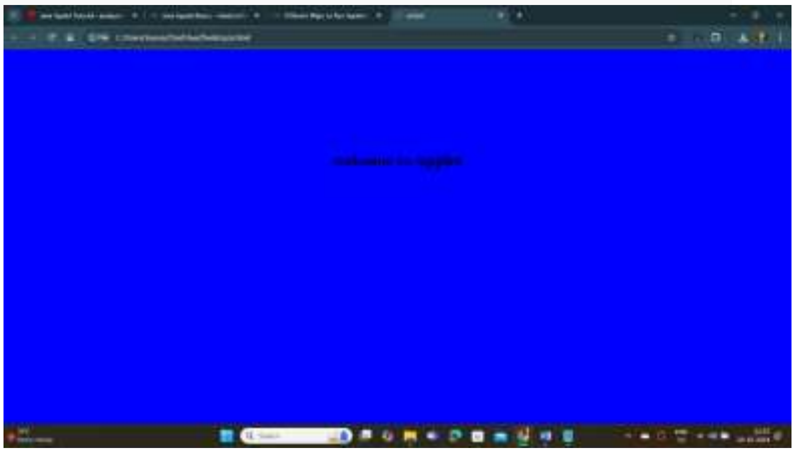
Attributes in applet tag:

- ✓ Code: It specifies the name of the applet class to load into the browser.
- ✓ Width: It specifies the width of an applet.
- ✓ Height: It sets the height of an applet.

Applet.html

```
<html>
    <body>
        <applet code="GfgApplet.class" width=300 height=100></applet>
    </body>
</html>
```

Now, double click on this applet.html file it will redirectd to browser and displayed as follows



welcome to Applet

Using an Applet Viewer to run the applet:

It is a java application that allows you to view applets. It’s similar to a mini-browser that will enable you to see how an applet might appear in a browser. It recognizes the APPLET tag and uses it during the creation process. The APPLET tag should be written in the source code file, with comments around it.

Write HTML APPLET tag in comments in the java source file.

Compile the applet source code using javac.

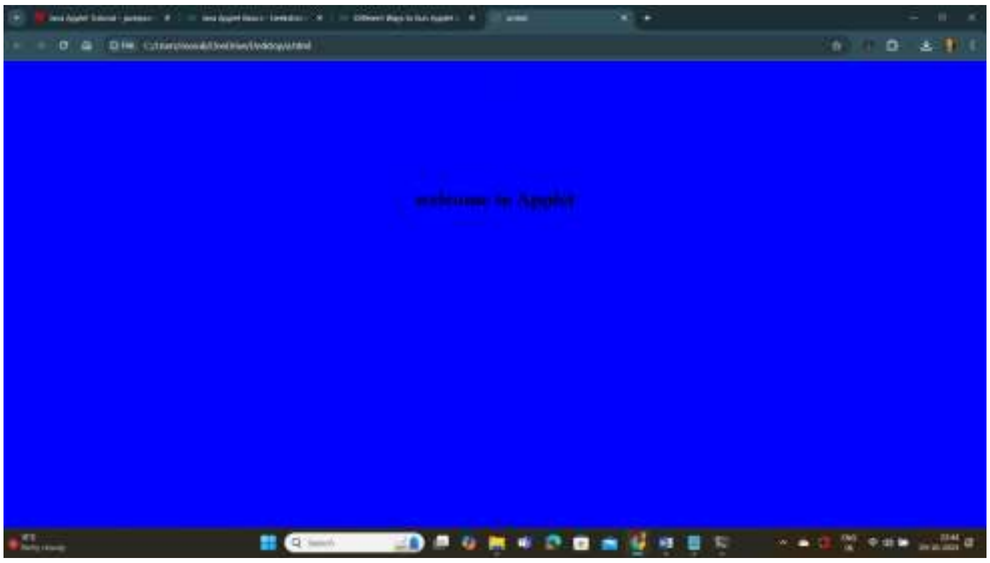
Use applet viewer ClassName.class to view the applet

```
import java.awt.*;
import java.applet.*;
/*
     <applet code="DemoApplet.java" width=400,height=600>
     </applet>
*/
class DemoApplet extends Applet
{
        public void init()
        {
                setBackground(Color.blue);
        }
        public void paint(Graphics g)
        {
                g.drawString("welcome to Applet",100,100);
        }
}
```

To use the applet viewer utility to run the applet, type the following at the command prompt:

```
E:\JAVA\ANU>javac DemoApplet.java

E:\JAVA\ANU>appletviewer DemoApplet.java
```

Output:



```
welcome to Applet
```

HTML Applet tag: **(read from text book)**