Throwable Class in Java with Examples

Last Updated : 24 Sep, 2021

Classes and Objects are basic concepts of Object-Oriented Programming which revolve around the real-life entities. A class is a user-defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. In this article, we will discuss the **Throwable** class, its constructors and different methods available in this class.

The **Throwable** class is the superclass of every error and exception in the Java language. Only objects that are one of the subclasses this class are thrown by any "Java Virtual Machine" or may be thrown by the Java throw statement. For the motives of checking of exceptions during compile-time, **Throwable** and any subclass of Throwable which is not also a subclass of either Error or RuntimeException are considered as checked exceptions.

Throwable class is the root class of Java Exception Hierarchy and is inherited by two subclasses:
1.Exception
2.Error
The throwable class implements Serializable Interface and the direct known classes to Throwable are Error and Exception.
Throwable contains a snapshot of the execution stack of its thread at the time it was created. It can also contain a message string that gives more information about the error. It can also suppress other throwables from being propagated.

If a user wants to create his own, custom throwable, then he/she can extend Throwable class.

Example:

```
Class MyThrowable extends Throwable{
//Here the user can create his own custom throwable
}
Class GFG{
Public void test() throws MyThrowable{
// the custom throwable created can be used as follows
throw new MyThrowable();
}
}
```

The class declaration for **java.lang.Throwable** class is as follows:

```
public class Throwable
    extends Object
    implements Serializable
```

Constructors: Any class can have any one of the three or all the three types of constructors. They are default, parameterized and non-parameterized constructors. This class primarily has the following constructors defined:

Public Constructors

1. **Throwable():** It is a non-parameterized constructor which constructs a new Throwable with null as its detailed message.
2. **Throwable(String message):** It is a parameterized constructor which constructs a new Throwable with the specific detailed message.
3. **Throwable(String message, Throwable cause):** It is a parameterized constructor which constructs a new Throwable with the specific detailed message and a cause.
4. **Throwable(Throwable cause):** It is a parameterized constructor which constructs a new Throwable with the specific cause and a detailed message of the cause by converting the case to the String using toString() method.

Protected constructors

1. **Throwable(String message, Throwable cause, boolean enableSuppression, boolean writableStackTrace)**:It Constructs a new throwable with the specified detail message, cause, suppression enabled or disabled, and writable stack trace enabled or disabled.
The parameters are:-
message – the detail message.
cause – the cause. (A null value is permitted, and indicates that the cause is nonexistent or unknown.)

enableSuppression – whether or not suppression is enabled or disabled
writableStackTrace – whether or not the stack trace should be writable.

Methods: Apart from the above mentioned constructors, there are also many predefined methods available in the throwable class. They are:

1. **addSuppressed(Throwable exception)**: This method appends the specified exception to the exceptions that were suppressed in order to deliver this exception.
Syntax:
Public final void addSuppressed(Throwable exception)
Returns: This method does not returns anything.
2. **fillInStackTrace()**:Fills in the execution stack trace. This method records information about the current state of the stack frames for the current thread within the current Throwable object.
Syntax:
public Throwable fillInStackTrace ()
Returns: a reference to the current Throwable instance.
3. **getCause()**: It returns the cause that was supplied via one of the constructors requiring a Throwable , or that was set after creation with the initCause() method.
Syntax:
public Throwable getCause ()
Returns: the cause of current Throwable. If the cause is nonexistent or unknown, it returns null.
4. **getLocalizedMessage()**: This method creates a localized description of current Throwable.
Syntax:
public String getLocalizedMessage ()
Returns: The localized description of current Throwable
5. **getMessage()**:Returns the detail message string of current throwable.
Syntax:
public String getMessage ()
Returns: the detailed message string of current Throwable instance(may also return null)
6. **getStackTrace()**: This method provides programmatic access to the stack trace information printed by printStackTrace(). It returns an array of stack trace elements, each representing one stack frame. The zeroth element of the array (assume that the array's length is non-zero) is the last method invocation in the sequence. It also represents as the top of the stack and is the point at which this throwable was created and thrown. The last element of

the array (assuming the array's length is non-zero) is the first method invocation in the sequence and it represents the bottom of the stack.

Syntax:

public StackTraceElement[] getStackTrace ()

Returns: an array of stack trace elements representing the stack trace related to current Throwable.

7. **getSuppressed()**:Returns an array containing all of the exceptions that were suppressed, in order to deliver this exception. If no exceptions were suppressed or suppression is disabled, an empty array is returned.

Syntax: public final Throwable[] getSuppressed ()

Returns: an array containing all of the exceptions that were suppressed to deliver this exception.

8. **initCause(Throwable cause)**:Initializes the cause of current Throwable to the specified value. This method can be called at most once. It is generally called from within the constructor, or immediately after creating the throwable.

Syntax: public Throwable initCause (Throwable cause)

Parameters:

Throwable cause- the cause of current Throwable.

Throws:

1.IllegalArgumentException: This exception is thrown if cause is the current throwable, because a throwable cannot be its own cause.

2. IllegalStateException: It occurs if this method has already been called on current throwable.

Returns: a reference to current Throwable instance.

9. **printStackTrace()**:Prints the current throwable and its backtrace to the standard error stream.

Syntax: public void printStackTrace ()

Returns: This method returns nothing.

10. **printStackTrace(PrintWriter s)**:Prints current throwable and its backtrace to the specified print writer.

Syntax: public void printStackTrace (PrintWriter s)

Parameters: PrintWriter- It is the PrintWriter to use for output

Returns: This method returns nothing.

11. **printStackTrace(PrintStream s)**:Prints current throwable and its backtrace to the specified print stream.

Syntax: public void printStackTrace (PrintStream s)

Parameters: PrintStream- It is the PrintStream to use for output

Returns: This method returns nothing.

12. **setStackTrace(StackTraceElement[] stackTrace)**:This method sets the stack trace elements that will be returned by getStackTrace() and printed by printStackTrace() and related methods.

Syntax: public void setStackTrace (StackTraceElement[] stackTrace)

Parameter: StackTraceElement- These are the stack trace elements to be associated with current Throwable.

Throws:

NullPointerException- if stackTrace is null or if any of the elements of stackTrace are null

Returns:

This method returns nothing.

13. **toString()**: This method returns a short description of current throwable.

Syntax: public String toString ()

Returns: a string representation of current throwable.

Below program demonstrates the **toString()** method of Throwable class:

```java
// Java program to demonstrate
// the toString() Method.

import java.io.*;

class GFG {

    // Main Method
    public static void main(String[] args)
        throws Exception
    {

        try {

            testException();
        }

        catch (Throwable e) {

            // Print using tostring()
            System.out.println("Exception: "
                               + e.toString());
        }
    }

    // Method which throws Exception
    public static void testException()
        throws Exception
    {

        throw new Exception(
            "New Exception Thrown");
    }
}
```

Output:

```
Exception:
 java.lang.Exception:
 New Exception Thrown
```

Below program demonstrate the **getMessage()** method of java.lang.Throwable Class:

```
// Java program to demonstrate
// the getMessage() Method.

import java.io.*;

class GFG {

    // Main Method
    public static void main(String[] args)
        throws Exception
    {

        try {

            // Divide the numbers
            divide(2, 0);
        }

        catch (ArithmeticException e) {

            System.out.println(
                    "Message String = "
                    + e.getMessage());
        }
    }

    // Method which divides two numbers
    public static void divide(int a, int b)
        throws ArithmeticException
    {

        int c = a / b;
        System.out.println("Result:" + c);
    }
}
```

Output:

```
Message String = / by zero
```

Reference: https://docs.oracle.com/javase/9/docs/api/java/lang/Throwable.html/