

"throw" Keyword

⇒ By default, methods create the exception object in which exception has occurred.
⇒ But sometimes, programmer needs to create exception object manually.
⇒ In that case we use "throw" keyword.

⇒ "throw" keyword is used to create an exception object manually (by programmer)

⇒ Syntax :-
 throw new ExceptionClassName(----);

⇒ Program :-

```
public class MainApp1
{
    void divide(int no1, int no2)
    {
        if(no2 > 0)
        {
            int res = no1 / no2;
            System.out.println("Result : "+res);
        }
        else if(no2 < 0)
        {
            System.err.println("no2 should be greater than 0");
        }
        else
        {
            //System.out.println("You cannot divide by zero");
            throw new ArithmeticException("You cannot divide by zero");
        }
    }

    public static void main(String[] args)
    {
        MainApp1 obj = new MainApp1();
        obj.divide(100, 0);
    }
}
```

⇒ Points to remember :-
1. throw keyword is mostly used for Custom Exceptions.
2. It is mostly used for "Unchecked Exceptions"
3. throw keyword should be the last statement in the block
4. throw keyword does not handle the exception, it only creates the exception object manually, but if we want to handle the exception, then we have to use try-catch bloc

"throws" Keyword

⇒ throws keyword transfers the exception object to the caller method
⇒ throws keyword is used for checked exceptions.

⇒ Syntax :-

```
void m2() throws Exception
{
    //exception occurred
}
```

⇒ Program

```
import java.io.FileInputStream;

public class MainApp3
{
    void m1() throws Exception
    {
        readFile(filePath: "d://aaa.txt");
    }

    void readFile(String filePath) throws Exception
    {
        FileInputStream fis = new FileInputStream(filePath);
        fis.read();
    }

    public static void main(String[] args) throws Exception
    {
        System.out.println("----- App Started -----");

        MainApp3 obj = new MainApp3();
        obj.m1();

        System.out.println("----- App Finished Successfully -----");
    }
}
```

⇒ throws keyword does not handle the exception, we have to use try-catch block for exception handling

Difference between throw & throws

⇒ throw keyword is used to create exception object manually
 throws keyword is used to transfer the exception object to its caller method

⇒ throw keyword is used within the method
 throws keyword is used with method signature

⇒ throw keyword creates a new instance
 throws keywords does not create any new instance

⇒ throw keyword can be used only for 1 exception class
 throws keyword can be used for multiple exception classes

⇒ throw keyword can be used as break statement because we cannot write any statement after throw statement
 throws keyword cannot be used as break statement

Create Custom Exception Class

⇒ Programmer can create any user-defined exception class
⇒ If programmer wants to create :
 → Checked Exception class, then we have to inherit the Exception class
 → Unchecked Exception class, then we have to inherit the RuntimeException class
⇒ Its recommend to create user-defined exception class as unchecked exception

⇒ Steps to create user-defined exception class :-
1. Create new class but it should have "Exception" in the class name
 For eg. InvalidAgeException
2. Extends Exception or RuntimeException class
3. Create public default and parameterized constructor
4. Use this custom exception class with "throw" keyword

Interview Questions

- ⇒ Java Interview Questions - throw Keyword
- What is the purpose of the throw keyword in Java?
 - What types of objects can be thrown using throw?
 - What is the difference between throw and throws?
 - Can we write multiple throw statements in a single method?
 - What happens if we throw a checked exception without handling it?
 - Can we throw an error using throw? Should we?
 - Can we rethrow an exception in Java? How?
 - What is the difference between throwing a new exception and rethrowing an existing one?
- ⇒ Java Interview Questions - throws Keyword
- What is the purpose of the throws keyword in Java?
 - Can we declare multiple exceptions using throws in a method?
 - What is the difference between declaring throws Exception and throws Throwable?
 - Can we declare unchecked exceptions in the throws clause?
 - What happens if we declare an exception in throws but never actually throw it?
 - Can a constructor declare a throws clause?
 - What is the difference between overriding a method with respect to the throws clause?
 - Can an overridden method throw broader exceptions than the parent method? Why or why not?
- ⇒ Java Interview Questions - Custom Exception Class
- What is a custom exception in Java? Why do we need it?
 - How do we create a custom checked exception in Java?
 - How do we create a custom unchecked exception in Java?
 - Should custom exceptions extend Exception or RuntimeException? Why?
 - Can we create a custom exception without extending Throwable?
 - What best practices should we follow when creating custom exceptions?
 - What is the difference between application-specific custom exceptions and built-in exceptions?
 - Can we create a custom error class in Java? Why is it not recommended?
 - What is the advantage of adding extra fields/methods in a custom exception class?
 - How do we chain exceptions in custom exception classes?
 - What is the difference between checked and unchecked custom exceptions in terms of usage?