Exception Handling

1. Exception VS Error

Exception	Error
1. Subclasses of Exception	Subclasses of Error
2. It can classified in two categories i.e. checked & unchecked	All errors in java are unchecked
3. It occur atcompile or runtime	It occur at runtime
4. It is mainly caused by the application itself.	It is mostly caused by the environment in which the application is running.
5. Eg - Checked - IOException, SQLException Unchecked - ArrayIndexOutOfBoundException, NullPointerException, ArithmeticException	Eg. Java.lang.StackOverFlow, java.lang.OutOfMemoryError

2. Checked / Uncheck (All eg.)

Checked	Uncheck
1. Checked at compile-time	Checked at runtime not compile time.
1. Must be handled or declared by the programmer.	Optional handling, not required to be caught or declared
2. Compiler enforces handling (try-catch or throws clause)	No compile-time enforcement, left to the programmer's choice
3. Subclasses of Exception	Subclasses of RuntimeException
4. **Eg-** IOException, SQLException, FileNotFoundException, ParseException	**Eg-** ArrayIndexOutOfBoundException, NullPointerException, ArithmeticException, IllegalArgumentException

3. Try / Catch / Finally

Try	**Catch**	**Finally**
1. The try statement allows you to define a block of code to be tested for errors while it is being executed.	The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.	The finally statement lets you execute code, after try…catch, regardless of the result
1. Code within ‘try’ is monitored for exceptions	Executed if an exception occurs in try	Always executed, even if there’s an exception
2. Can have multiple ‘catch’ blocks to handle various types of exceptions.	Multiple ‘catch’ blocks can be used to handle different exception.	Usually not used for handling exceptions, but only one ‘finally’ block.

4. Final / Finally

Final	**Finally**
1. Used to declare a variable, method, or class as constant, unchangeable, or to prevent inheritance.	Used in a try-catch block structure to specify code that must be executed regardless of whether an exception is thrown or not.
2. Used in variable, method, or class declarations.	Used within the context of a try-catch block.
3. Can be used for variables, methods, or classes.	Typically used within a try-catch block and appears once at the end of the block.

5. Throw vs Throws

Throw	Throws
1. Used to explicitly throw an exception within a block of code.	Used in method declarations to indicate that the method may throw certain exceptions.
2. Used inside methods, constructors or blocks to raise an exception manually.	Used in method declarations.
3. Can be used multiple times within a block.	Lists multiple exceptions a method may throw.
4. Syntax - `throw new ExceptionType("Message");`	Syntax - `void methodName() throws ExceptionType { ... }`

6. Define Exception handling

Exception handling is a programming paradigm that manages and addresses errors during program execution. It is like having a backup plan in programming. It helps the code deal with unexpected issues, so the program doesn't just crash. It's like catching problems as they happen and figuring out how to fix or handle them, making sure the program keeps running smoothly. Developers use constructs like try, catch, and finally to gracefully manage unexpected events, enhancing code robustness and maintaining program stability.

7. Can we use catch without try

No, we cannot use catch without try. The catch block is meant to handle exceptions that occur within the corresponding try block. It provides a way to respond to and manage errors in a controlled manner.

8. WAP to show runtime exception & display (Eg. 7/0)

```
public class RuntimeEg {
    public static void main(String[] args) {
        try {
            // Attempting to divide 7 by 0
            int result = 7 / 0;
            System.out.println("Result: " + result);
```

```
          } catch (ArithmeticException e) {
              System.out.println("Exception caught: " + e.getMessage());
          }
      }
  }
```

9. ArrayIndexOutOfBoundException program

```
public class ArrayIndexOutOfBoundsExceptionEg {
    public static void main(String[] args) {
        try {
            int[] n = {1, 2, 3};

            // Attempting to access an index that is out of bounds
            int value = n[5];

            System.out.println("Value: " + value); // This line won't be reached
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Exception caught: " + e.getMessage());
        }
    }
}
```

By - Divya Parihar