[2.38.05]

Exceptions Handling

Aspect	Error	Exception
Type	These are critical, usually system-related issues.	Application-level problems that can be handled.
Hierarchy	Subclass of "Throwable"	Subclass of "Exception" ~~"Throwable"~~
Recovera-bility	generally unrecoverable	"Throwable". Can be catch and handled using try-catch block.
Checking	No	These are checked and handled at Compile-time
Examples	OutOfMemoryError, StackOverflowError, VMError.	IOException, NullPointerException, SQLException.
Common Sources	Hardware failures, JVM limitations	Invalid operations, incorrect logic, faulty Input/output.

Types of Exceptions :-

(i) Checked Exceptions :-

→ Also known as "Compile-time Exception"

→ This can be handled using "try-catch" blocks or by declaring them with the "throws" clause.

Eg :- IOException, SQLException

(ii) Unchecked Exceptions :-

→ Also known as "Run-time Exception".

→ Caused by Programming error.

Eg :- NullPointerException, ArithmaticException.

(a) Compile Time :-

→ The Phase where the Program is translated into machine code (bytecode).

→ during this phase the Code is checked for Syntactical Correctness and the Compiler identifies any errors in the Code.

→ If there are errors then the Program will not enter into run-time for execution.

Eg :- Syntax Errors / type Errors, Missing class or Method, Unchecked Exception.

b) Runtime :-

→ Phase after the Compilation of a program. [Program passes compilation]

→ Errors may occur here based on the actual data and input that the program is processing.

→ Errors occur due to incorrect logic, invalid User input or unexpected conditions.

Eg :-

NullPointerException,
ArithmeticException,
ArrayIndexOutOfBoundsException
IllegalArgumentException.

Throw

The "throw" keyword is used to explicitly throw an exception from within a block of a code or within a method.

Eg :-

```
if (age < 18) {
    throw new ArithmeticException
        ("Age must be 18 or above."); }
else {
    S.OP ("Welcome to the club!");
}
```

Throws

→ Used in method declarations to indicate that the method may throw 1 or more Exceptions.

Eg:-

```
Static void check (int age) throws UserDefined,
    ArithmeticException {
    if (age < 18) {
    throw new UserDefined ("Age invalid");
    }
  else {
    throw new ArithmeticException("Age valid");
  }
}
```

Output:- will be based on the input in the try block :-

i) If age < 18

↓

Age invalid

ii) If age > 18

↓

Age valid

Throwable

→ This is a Superclass of all exceptions and errors in java.

→ A Part of the java language's exception hierarchy.

[3.08.14]

Predefined Classes

i] toString => Converts a datatype into String

Eg:- Int to String

Eg:-

```
String s = Integer.toString(55);
S.O.P(s);
```

Output :-

55

ii] toBinaryString => Converts a given Int into it's binary form.

Eg:-

```
String s = Integer.toString(15);
S.O.P(s);
```

Output :-

1111

11]

Value of , ParseInt

Converts a datatype into Int.

```
int y= Integer.parseInt(x);
S.O.P(y);
```

Output;

1111

ValueOf => returns an object of the Integer Class.

ParseInt => returns Int (Primitive datatype)

File Reading

1:53]

→ Used only Inside a 'try' 'Catch' Block

```
import java.util.Scanner;
import java.io.File;

Class DT{
    Public static void main(String[] args){
        try{
            File f = new File("Example.txt");
            Scanner reader = new Scanner(f);
            while(reader.hasNextLine()){
                S.O.P(reader.nextLine());
            }
    }
```

```
    Catch (Exception E){
        S.o.p(E);
    }
}
}
```

Output :-

i) If file found.

=> Content in the file is displayed.

ii) If file is not found.

=> "File Not Found Error" will be displayed

Absolute path :- Starts from "C:\Localdisk"...

Relative path :- Path other than absolute path

File Writing

-> Import FileWriter

-> Create an object for FileWriter

```
//
FileWriter fw = new FileWriter(f);
fw.write("Hello");
//
fw.close();
```

OTHERS

① Why Java is not a pure object Oriented Programming language?

i) Use of Primitive Data types:-

⇒ int, byte, short, long
float, double, char, boolean

ii) Wrapper classes:-

allows primitive types to be treated as objects

⇒ Integer, Double, Boolean

eg:-

```
Integer obj = Integer.valueOf(5);
```

iii) Static Methods:-

S.O.P, Math.max(), Math.min()...

↓

this belongs to a class but not tied up to an object's instance.

eg:-

```
int result = Math.max(10,20);
// static method call
```

iv) Absence of Multiple Inheritance.

2] Pure object-Oriented :

Treats everything as an object, including primitive types, functions and methods.

3] Java features :

- → Write Once, Run Anywhere (WORA) [Platform independent]
- → Object Oriented
- → Uses "Garbage Collectors" for memory management.
- → Simple
- → Secure [No direct pointer access as in C/C++]
- → Robust, Portable
- → Multithreading [multiple programs can be run simultaneously]
- → High Performance [Just-In Time (JIT) compiler] => Compiles bytecode to native machine code at runtime.
- → Distributed
- → Dynamic
- → Architecture - Neutral

4] JVM [Java Virtual Machine]

⇒ The Compiled bytecode is converted into machine-specific code using this.

(.java files) is Compiled by the Java Compiler into bytecode (.class files).

JRE [Java Runtime Environment]

JRE = JVM + Libraries + other Tools
(Core Java APIs)

Used by users who want to run Java applications.

JDK [Java Development Kit]

JDK = JRE + Development tools
(Compiler, debugger)

Used by developers for writing and Compiling Java applications.

5] Constructor Overloading :-

Allows us to define multiple Constructors with different parameter lists within the same class.

Eg:-

```
Class Car {
    String model;
    int year;

    // Constructor 1 : No arguments
    Car() {
        model = "Unknown";
        year = 0;
    }
    // Constructor 2 : 1 argument
    Car(String model) {
        this.model = model;
    }
}
Public class Main{
    Public static void main(String[] args){
        Car car1 = new Car();
        Car c2 = new Car("Toyota");
    }
}
```

Constructor Overridding :-

=> Java does not support Constructor Overridding because constructors are not inherited by subclasses.

=> If a Subclass want's to inherit a Constructor then we need to use "Super" keyword, this changes the Context to inheritance.

6] Operator Overloading :-

A special feature is some programming languages like C++ where the same operator can be give different meanings based on the type of its operands.

Note! :-

Java does not Support operator Overloading to keep the language Simple and avoid Potential misuse misuse.

Eg :- To add an Complex number we can used the add() method in Java instead, we need to Create a add() method.