Agenda

- Revision
- Exception Handling
    - Exceptions
    - Errors
    - Custom Exceptions
    - Exception Chaning
- Date/LocalDate/Calender

class vs abstract class vs interface

- class
    - it consists of static and non static fields
    - it consists of non abstract methods.
    - It extends the class and also can be inherited into subclasses
    - class can extend only one class
    - class can implement multiple interfaces
    - we can create object of the class.
- abstract class
    - it consists of static and non satic fields
    - it consists of abstarct as well as non abstract methods
    - It extends the class and also can be inherited into subclasses
    - It can extend only one class
    - It can implement multiple interfaces
    - we cannot create object of the abstract class but we can create the reference.
- interface
    - if field is defined in interface it is by default public static final
    - all the methods declared as by defualt abstract.
    - It is inherited/implemented into sub interfaces / subclasses
    - interface can extend multiple interfaces, they cannot extend the class
    - It cannot implement another interface
    - we cannot cretae object of the interface class but we can create the reference.

Exception Handling (Dmeo01 to Demo03)

- Exception are runtinme problems
- When exceptions/problems occuer then generally they are required to be handled.
- If not handled , the program can terminate abruptly.
- In java, Exception Handling is done using below keywords
    - try
    - catch
    - throw

    - throws
    - finally
- In java the operators/ API's throws exception if some runtime wrong inputs are provided to them.
- to check for the exceptions if the statements are generating, such statements must be kept inside try block
- whatever exception are thrown by the statements from the try block must be handled inside matching catch block.
- a try block should have atleast
    - a catch block
    - a finally block
    - try-with-resource
- Throwable is the superclass of all the errors and exceptions in java
- java have divided the exceptions in two categories
    1. Error
        - It is the problem that occurs due to runtime environment
        - problems like issues in memroy allocations in RAM/JVM
        - problems like crashing of HDD, etc..
        - such errors should not be handled
        - we can write a try catch for such errors but it is highly recommended not to handle such errors
    2. Exception
        - problems that occur at runtime due to the wrong inputs
        - in java it is recommended to handle the exceptions

```
- Throwable
  - Error
    - IOError
    - VirtualMachineError
      - OutOfMemoryError
      - StackOverflowError

  - Exception (Checked Exception)
    - CloneNotSupportedException
    - IOException

    - RuntimeException (Unchecked Exception)
      - ArithmeticException
      - ClassCastException
      - DateTimeException
      - IndexOutOfBoundsException
        - ArrayIndexOutOfBoundsException
        - StringIndexOutOfBoundsException
      - NegativeArraySizeException
      - NullPointerException
      - NoSuchElementException
        - InputMismatchException
```

- Exception class and all its sub classes except RuntimeException class are checked exceptions

- these exceptions are checked at compile time and hence we need to compulsary handle these exceptions
- RuntimeException class and all its subclasses are unchecked exceptions
- these exceptions are not checked at compile time and hence optional to handle them.
- if you want to handle all the checked and unchecked exceptions in single catch block then handle it using the Exception class reference.
- such catch block is called as generic catch block

throw & throws Keyword

- throw keyword is used to generate the exception
- we can use the throw keyword to genereate checked as well as unchecked exceptions
- if we throw an exception of type checked from any method then , we have to use 'throws' keyword to route that exception towards the calling method.
- if we throw an exception if type unchecked from any methods then using 'throws' keyword to route the exception is optional

Exception Handling Keywords

- try
    - It is used to check for the exceptions that are generated by the statements
    - try should have atleast one of the below:
        - one catch block
        - one finally block
        - try-with-resource
- catch
    - it is used to handle the exceptions that are thrown from try block
    - for every exception from try block their should be a matching catch block.
    - a single try block can have multiple catch blocks
    - we can handle all exceptions in a single catch block using superclass of all the Exceptions called as Generic Catch Block
- throw
    - It is used to generate an exception.
    - we can generate checked as well as unchecked Exceptions
- throws
    - It is used to route exception from the method to its calling method.
    - using throws is optional for unchecked exceptions
    - using throws is compulsary for checked exceptions
    - we should no route the exceptions from main towards the JVM. it is a bad way of programming
    - all the exceptions that are not handled should be handled inside main
- finally

- It is block used to close the resourses
- this block can be use directly with the try block or can also be used with the catch blocks.
- the finally block should be mentioned at the last after all catch blocks.
- It gets executed every time irrespective of their is exception or no any exception

Custom Exception Class (Demo04)

- We can create custom Exception classes.
- the custom exception classes can be of type checked or unchecked.
- to make the custom exception class of type checked extend it from Exception class and to make it as unchecked extend it from RuntimeException class.
- we can also wrap one Exception type Object inside its super type Object and can throw the Exception.
- this is called as Exception Chaining

Date/ LocalDate/ Calender (Demo05)

- Date and Caleneter are the classes from java.util package
- Date class should not be used as the methods are depreceated and it is recommended to use calender class.
- To get the object of Calender class we have to call the method Calender.getInstance()
- Calender is a mutable class.

```
Calender calender = Calender.getInstance();
```

Lab Work

- Slide Reading max 20 minutes
- If required solve the demos.
- Solve the assignment.
- Revise the pending topics