==========
What is Logging ?
==========

=> Storing application execution details to a log file is called as Logging.

=> Logging is used to understand runtime behaviour of the application.

=> We can understand exceptions occuring in the application using log msgs.

Note: If we don't implement then understanding application behaviour and understand root cause of the problems occuring in application is very difficult.

=> In realtime, every project will have logging implementation.

==========
Logging Architecture
==========

1) Logger : It is used to generate log msgs in our application

2) Layout : It represents log msg pattern

3) Appender : To write log msg to destination (console / file)

==========
Logging Levels
==========

-> Log level represents what type of log messages we have to generate in log file.

1) TRACE

2) DEBUG

3) INFO (default)

4) WARN

5) ERROR

=> We can change log level for our application.

=> From the configured log level , it will print all higher level log msgs also

INFO : INFO + WARN + ERROR

WARN : WARN + ERROR

ERROR : ERROR

DEBUG : DEBUG + INFO + WARN + ERROR

==========
Logging in Spring Boot
==========

=> Spring Boot using Logback framework to perform logging

=> Default log level in spring boot is info

=> The default Layout is "Pattern Layout"

```
        Ex: {date} {time} {level} {class-name} : {msg}

=> Default appender is console.

============================================================
changing logging properties in application.properties file
============================================================

logging.level.root=ERROR
logging.file.name=myapp.log

============================
RestController with Logging
============================

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class MsgRestController {

        Logger logger = LoggerFactory.getLogger(MsgRestController.class);

        @GetMapping("/welcome")
        public String getWelcomeMsg() {

                logger.trace("this is log - trace msg");
                logger.debug("this is log - debug msg");
                logger.info("this is log - info msg");
                logger.warn("this is log - warn msg");
                logger.error("this log - error msg");

                String msg = "Welcome to Ashok IT..!!";

                return msg;
        }

        @GetMapping("/greet")
        public String getGreetMsg() {

                String msg = "Good Morning..!!";

                return msg;
        }
}


=======================
Rolling File Appenders
=======================

ConsoleAppender : Log msgs will be printed on console

FileAppender : Log msgs will be stored in a file

Note: If we use single log file to store all log msgs then file size will be increased and lot of
msgs will be stored on daily basis. It will become difficult to manage and monitor log msgs of our
application.

-> To avoid these problems, we will use RollingFileAppender concept to roll log files.


Size Based Rolling : After storing 1 GB data, create new log file.
```

Time Based Rolling : Everyday create new log file to store log msgs

==========
Logging Frameworks
==========

1) Log4J

2) Log4J2

3) Logstash

4) Logback (default in spring boot)

==========
What is SLF4j ?
==========

-> Simple Logging Facade for Java

-> It provides abstraction with logging frameworks

-> By using SLF4J, we can plug-in any logging framework in runtime

-> If we implement logging using SLF4J, then our application will be loosely coupled with logging framework. We can change logging framework at the time of deployment.

==========
configuring logback.xml
==========

-> logback.xml file is used to configure logging information

-> We will keep logback.xml file in src/main/resources folder

-> Below is the sample logback.xml file

```
<configuration>
        <appender name="Console" class="ch.qos.logback.core.ConsoleAppender">
                <encoder>
                        <pattern>%d [%thread] %-5level %-50logger{40} - %msg%n</pattern>
                </encoder>
        </appender>
        <appender name="RollingFile"
                class="ch.qos.logback.core.rolling.RollingFileAppender">
                <file>MyApp.log</file>
                <encoder>
                        <pattern>%d [%thread] %-5level %-50logger{40} - %msg%n</pattern>
                </encoder>
                <rollingPolicy
                        class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
                        <fileNamePattern>MyApp-%d{yyyy-MM-dd}.%i.log</fileNamePattern>
                        <maxFileSize>1MB</maxFileSize>
                        <maxHistory>30</maxHistory>
                        <totalSizeCap>10MB</totalSizeCap>
                </rollingPolicy>
        </appender>
        <root level="INFO">
                <appender-ref ref="Console" />
                <appender-ref ref="RollingFile" />
        </root>
</configuration>
```

```
==========================
What is Logs Monitoring
==========================

-> Application will run in remote server (Ex: AWS Cloud machine)

-> Application log files will be created in Remote Server

-> Log Monitoring tools are used to get application logs from remote machines

-> We have several log monitoring tools in the market

                Ex:  MobaXterm, Putty, WinScp, ELK, Splunk...


*JRTP - Assignment*

Task-1 : Create Free Tier account in AWS cloud

        Video : https://youtu.be/xi-JDeceLeI?si=PupxskKrJdC2ICRS

Task-2 : Deploy SpringBoot App in AWS cloud

        Video : https://youtu.be/cRQPgbwOWq0?si=2IjWn8pv0eeKZY1k


====================================
Deploy Spring Boot App in AWS Cloud
====================================

1) Develop Spring Boot REST API with Logging

2) Package SpringBoot application as jar file using Maven Goal

3) Create Account in AWS Cloud

4) Create Linux VM (Amazon Linux) in AWS cloud using EC2 service

5) Connect with Linux VM using MobaXterm s/w

6) Install Java software in linux vm

        $ java -version
        $ sudo yum install java
        $ java -version

7) Upload our spring-boot project jar file into linux vm

        $ ls -l

8) Execute jar file

        $ java -jar <jar-file> (terminal will be blocked to see log file)

Note: To exit from terminal we need to press 'Ctrl + C' it will stop our application.

-> To avoid this process we can use 'nohup' command to run our jar file

        $ nohup java -jar sb-rest-api.jar &

        $ ls -l

9) Enable 8080 port number in security group to access our application outside
```

10) Access our application Urls

        URL-1 : http://public-ip:8080/welcome
        URL-2 : http://public-ip:8080/greet
        URL-3 : http://public-ip:8080/product

==========
How to monitor logs using mobaxterm
==========

cat <log-file> : It will print content of log file

head <log-file> : Displays top 10 lines of content in file (top to bottom)

tail <log-file> : Display last 10 lines of content in file (bottom to top)

tail -n 50 <log-file> : Display last 50 lines of content

tail -f <log-file> : To print on-growing logs (realtime logs)

grep -i 'Exception' MyApp.log : Display lines which contains exception keyword

Note: Using MobaXterm GUI option, we can download log file also.

==========
ELK Stack
==========

=> ELK stack is used for log aggregration

E -> Elastic search (it is used to store the logs)

L -> Log Stash (It will take logs from log files and store into Elastic search)

K -> Kibana (It will provide User Interface to get logs)

-> ELK products are open source.

ELK Demo : https://youtu.be/8MLcbbfEL1U?si=__A9gteIP09NVThm

==========
Splunk
==========

-> It is very advanced log monitoring software

-> Splunk is licensed software

Note: In realtime most of the companies will prefer Splunk software to monitor logs of the application.

Splunk Demo Video : https://youtu.be/BUfOuWLgnMU?si=K0vWWUPdyU8BYrx5

==========
Summary
==========

1) What is Logging

2) Why Logging

3) Logging Architecture
- Logger
- Layout
- Appender

4) Logging Levels

5) How to change log level

6) How to implement Logging in Spring Boot project

7) Logging Frameworks

8) SLF4J vs Log4J

9) Rolling File Appenders

- Time Based Rolling
- Size Based Rolling

10) What is logback.xml

11) What is Log Monitoring

12) Tools available to monitor logs of application