19/04/2023 :: 7AM to 9.30AM (SpringBoot Mongodb)
20/04/2023 :: SpringAOP,Spring Mail
21/04/2023 :: Spring Mail,SpringSecurity
==
Weekend :: SpringRest(saturday and sunday) :: 9.00AM to 2.00PM

Microservices
=============
24th to 30th(Full Microservices)

Eclipse Debugging
=================
Debugging is a process of getting step by step by execution in an application to find out problem/bugs and fix them as needed.
Terminologies associated with Debugging
a. Bugs => Problem/defect/mistake in the app/program/project[some abnormality in the application execution]

Debug = > De + bug [Rectifiying the bug by identifying and Analyzing the bug]
The tools or programm or software that can be used for debugging is called "Debugger"/Debugging tool

To debug java code we have two tools
a. JDB(Supplied by jdk tools with jdk installation)
=> available inside <java_home>\bin directory as jdb.exe
=> it is CUI tool and not so popular(Not industry standard)

b. IDE supplied Debugger
=> Eclipse/Intelij/EclipseLink and etc ... IDE supplied Debuggers
=> These are GUI and they are user friendly
=> These are industry standard

Where Debugging is required in the company?
a. To find,analyze and fix logic problems/errors
b. if unit testing is failed, then also we use debugging to fix the problems[Run Junit in Debugging mode]
c. After joining in the company that to existing project then the programmer performs debugging process to know flow in the project.
d. If the project is in production then the bugs received will be fixed only after seeing log files[log messages] and after
performing debugging.

refer:: Debuggingimage_01

Debugging in eclipse
====================
=> Run the apps in Debug mode
=> Change eclipse mode to debug perspective
=> use various debugging operations/shortcuts to see the code flow to find out the problem and fix the problem.

breakpoint:: It is the point in the App/Project/code/program from where the debugging process would start.
Till the breakpoint code executes normally.
From the breakpoint control comes to programmer to see the code flow as needed.

Two types of Breakpoint

=======================
1. Method Breakpoint :: Give at method name of method defnition
2. Line Breakpoint   :: Given inside the method defnitions where logics are available.

DemoApp.java
============

```java
package in.ineuron.test;
import java.util.Scanner;
public class DemoApp {

	public void sayHello(String name) {
		System.out.println("DemoApp.sayHello()");
		for (int i = 1; i <= 10; i++)
			System.out.println(name + " ");

		System.out.println("End of sayHello()");
		Scanner scanner = new Scanner(System.in);
		System.out.print("Enter the message:: ");
		String msg = scanner.next();

		displayMsg(msg);

	}

	public void displayMsg(String msg) {
		System.out.println("DemoApp.displayMsg() :: "+msg);
		System.out.println("DemoApp.displayMsg() :: "+msg);
		printNumber(10);
	}

	public void printNumber(int n) {
		System.out.println("DemoApp.printNumber()");
		for(int i = 1;i<=n;i++)
			System.out.println(i);
		System.out.println("end");
	}

	public int add(int a, int b) {
		System.out.println("DemoApp.add()");
		return a + b;
	}

	public int sub(int a, int b) {
		System.out.println("DemoApp.sub()");
		return a - b;
	}

	public static void main(String[] args) {
		DemoApp app = new DemoApp();
		app.sayHello("sachin");
		app.add(10, 20);
		app.add(30, 20);

	}
}
```

eclipse shortcut
================
  ctrl+shift+B :: To toggle break(To enable or disable method/line break point)
      F6 :: Step Over(line by line execution)
      F5 :: Step into(executing the method body)
     F7 :: Get the control back to caller
     F8 :: Getting the control to move into another breakpoint(if it is the last break point then program would be terminated)
   drop to frame :: re-run back to the program[Takes the control to be the begining of the method defnition whethere breakpoint is
                    available or not.]
    Ctrl+R:: Take the cursor to that line [keep the breakpoint==> select any line then cursor will go that line]
    Ctrl+F5:: To step into selection[skipping the certain logic then moving into specific one]

Working with StepFilter
======================
  => It allows certain packages not to participate in debugging, It is useful especially if you are not interested in getting into details of
     predefined classes and the methods like System.out.println().

Step1: Add the packages which should not participate in debugging process
       window => prefereces =>java ====> Debugging => stepfiltering =>Add packages(which should not participate)

Step2: Make sure step filter is in active mode
Step3: Debug the java application and press f5 button for clases and methods of the above packages ,but it will not step into method defnition

Note:: By keeping mouse over variable or parameter of the debugging mode method defnition we can watch value changes that are taking the
       place in the application execution.
=> we can use variable window,expression window to see the value changes.
=> expression window is useful to given current inputs for another formula of buisness logic to check the results.
=> Breakpoints window give list of breakpoints.
=> Show we can enable or disable them as needed.
=> we can add new breakpoints in the debugging mode of the application(dynamically during the execution).

Standalone layered application components
===
controller =====> service =======> DAO =====> Database

=> It is possible to add break points not only programmer developed user defined classes.
=> It is possible in predefined classes that are internally used by user defined classes
   like HttpServlet,GenericServlet,System,DispatcherServlet,.....
=> While working with maven/gradle build tools we get built in java decompiler, there is no need of arranging
   decompilers seperately.
              To see source code :: F3
           To open details of current class/interface/enum including hierarchy :: F4
=> In the middle of debugging also we can add/remove new break points dynamically

```
                  eg:: Hibernate-DebuggingMode
                        IOC-RealTimeDependancyInjection

Debugging on WebApplication(Server level debugging)
===================================================
 Instead of Run as Server,we need to choose Debug as server

code insde servlet
=================
 public void doxXXXX(request,response){
      PrintWriter out = response.getWriter();
      out.println("<b>Today date is :: "+new java.util.Date()+"</b>");
      for(int i =0;i<=10;i++)
            out.println("<b>sachin&nbsp&nbsp"+i+"</b>");
      out.println("end of doxXXXX()");
 }
```